How Tokenization Limits Phonological Knowledge Representation in Language Models and How to Improve Them

Disen Liao[1,2] **Freda Shi**[1,2]
[1]University of Waterloo [2] Vector Institute
{d7liao,fhs}@uwaterloo.ca

Abstract

Tokenization is the first step in every language model (LM), yet it never takes the sounds of words into account. We investigate how tokenization influences text-only LMs' ability to represent phonological knowledge. Through a series of probing experiments, we show that subword-based tokenization systematically weakens the encoding of both local (e.g., rhyme) and global (e.g., syllabification) phonological features. To quantify this effect, we introduce the syllabification-tokenization alignment distance (STAD), a metric that measures the misalignment between a model's tokenization and the natural syllable boundaries of words, and find that higher misalignment correlates with poorer phonological representations, providing a simple diagnostic for phonology-aware tokenization. To address these limitations, we propose a lightweight IPA-based fine-tuning method that infuses phonological awareness into LMs, leading to consistent improvements across three phonology-related tasks while largely preserving math and general reasoning ability, with 1.1% and 0.9% drops on GSM8K and MMLU, respectively.[1]

1 Introduction

Language models (LMs) have achieved remarkable success across diverse tasks, yet their understanding of sound structure, i.e., how words are pronounced, rhymed, or syllabified, remains poorly understood. Surprisingly, even text-only LMs such as GPT-4o exhibit awareness of word pronunciation, enabling creative and educational applications like poetry generation (Zhang and Eger, 2024; Yu et al., 2024) and language learning assistants (Hamaniuk, 2021; Bonner et al., 2023). This raises a fundamental question: how do LMs represent the phonological properties of words, despite never being trained on speech? We focus on text-only models as they must infer phonology from orthography alone, which lets us isolate the role of text tokenization in that emergence.

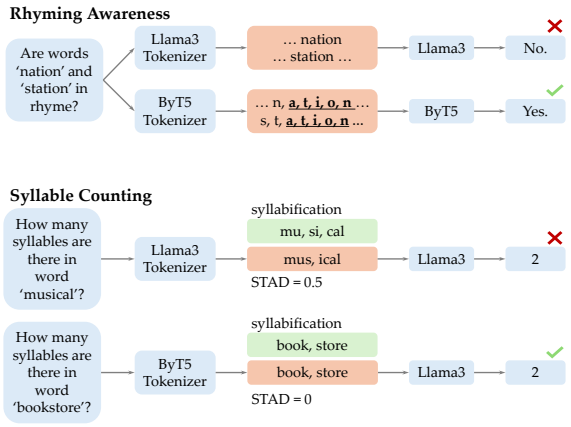


Figure 1: Illustrations of two key issues in LMs' phonological understanding caused by tokenization, using rhyming awareness and syllable counting as examples. The **top** shows that subword tokenization (e.g., BPE) may be too coarse to capture local phonological information, whereas the **bottom** shows that tokenization may misalign with natural syllable boundaries, leading to difficulties in learning prosodic structures.

We take a systematic approach to examine how tokenization, the first stage of text processing in every LM, shapes the model's capacity to represent phonological structure. We focus on two complementary aspects of phonological knowledge.

- **Local phonological features**, which capture whether words share similar sounds in part, for example, whether *nation* and *station* rhyme.
- **Prosodic structure**, which concerns the overall rhythmic organization of words, such as how *musical* has three syllables and how graphemes map to phonemes in pronunciation.

To probe these aspects, we first consider three representative tasks—rhyming awareness, grapheme-to-phoneme (G2P) conversion, and syllable counting—and use linear probes to quantify how well hidden representations at different model layers encode relevant phonological features.

[1]Our code is available at https://github.com/liaodisen/Tokenization-Phonology

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13921–13938

Our findings reveal that tokenization plays a pivotal role in shaping phonological competence. Subword tokenization methods such as Byte Pair Encoding (Gage, 1994; Sennrich et al., 2015) and SentencePiece (Kudo, 2018), while computationally efficient, are often (1) either too coarse to capture local phonological features or (2) misaligned with phonological boundaries (Figure 1). We further show that these issues systematically negatively affect models' ability to capture both rhyming relations and prosodic structure. We introduce the syllabification-tokenization alignment distance (STAD) to quantify the misalignment between a model's tokenization and the natural syllable boundaries of words, and show that higher misalignment leads to degraded phonological representations. This turns a broad intuition about tokenization into a measurable diagnostic that can guide tokenizer evaluation and design, while also clarifying how phonological structure can emerge in LMs despite the absence of speech.

To address these limitations, we propose a lightweight fine-tuning method that explicitly incorporates phonological information, offering a practical retrofit for existing models whose tokenizers cannot be easily changed. By augmenting general-purpose QA data with phonological annotations, our approach enables LMs to reason more effectively about pronunciation while largely preserving their general reasoning ability, albeit with small trade-offs on broad benchmarks.

2 Related Work

Probing the linguistic competence of LMs. Probing (Ettinger et al., 2016) investigates the internal representations of language models by training lightweight classifiers on hidden states to predict specific attributes, such as truthfulness (Azaria and Mitchell, 2023), spatial understanding (Gurnee and Tegmark, 2023), sound perception (Ngo and Kim, 2024), and sound symbolism (Alper and Averbuch-Elor, 2024). Compared to performance-based evaluation methods like accuracy, probing reveals more nuanced latent knowledge (i.e., competence; Chomsky, 1965) embedded within internal activations (Burns et al., 2022)—even when a model produces incorrect predictions, it may still encode relevant information.

Following recent work that advocates for using simple linear models as probes (Ettinger et al., 2016; Alain and Bengio, 2018; Hewitt and Manning, 2019), in this work, we employ simple linear models to study how well the representations learned by LMs capture phonological features.

Phonology in LMs. Benchmarks for assessing the phonological capabilities of LMs are still in an early stage. Recently, Suvarna et al. (2024) introduced a benchmark specifically designed to evaluate LLMs' performance on phonological tasks. They proposed three tasks: rhyming generation, G2P, and syllable counting, to evaluate the phonology ability of LLMs, which motivates our choice of tasks in this work.

On the other hand, using LMs for phonological inference has become a promising direction, where some phoneme-based models have been tailored for lower-level phonological tasks. For instance, PhonemeBERT (Sundararaman et al., 2021) is fine-tuned on a dataset combining ASR transcripts and phonemes, while Mix-Phoneme BERT (Zhang et al., 2022) is pre-trained with phonemes and sub-phonemes as additional features to enhance text-to-speech performance. Furthermore, Qharabagh et al. (2024) demonstrate that LLMs could significantly improve G2P conversion, especially in low-resource languages, showing the potential of LLMs to advance phonological processing in linguistically underserved contexts. In contrast to this line of work, our study focuses on understanding how tokenization affects the phonological knowledge encoded in LMs' hidden states, aiming at offering fundamental insights that can inform future model design, as well as training, fine-tuning and inference strategies.

Tokenization Pitfalls. Subword-based tokenization algorithms such as BPE (Gage, 1994; Sennrich et al., 2015) are widely used in training modern LMs. Prior work has highlighted how tokenization can introduce artifacts that impact model performance, particularly in tasks involving phoneme and grapheme representations (Shin et al., 2020, *inter alia*). Additionally, tokenization consistency plays a crucial role in extractive QA tasks (Sun et al., 2023). Singh and Strouse (2024) further argued that for numeric reasoning tasks, LLMs perform better when numbers are tokenized from right to left. To mitigate tokenization-induced issues, we draw insights from Deng et al. (2023) and Bunzeck et al. (2024), and propose a fine-tuning strategy (§5) that incorporates phonological information to help LMs overcome tokenization pitfalls.

Meanwhile, character-level tokenization has

been explored as an alternative to subword-based methods to eliminate tokenization biases. For instance, BERT has been shown to exhibit sensitivity to misspellings due to its reliance on subword tokenization (Sun et al., 2020). To address the limitations of subword tokenization, character-level or sub-character level tokenization strategies have been explored, including pre-training variants such as CANINE (Clark et al., 2022) and ByT5 (Xue et al., 2022). In line with these existing efforts, our work highlights tokenizer pitfalls in phonological tasks, extending these findings toward better phonological understanding in LMs.

3 Experiment 1: Probing Phonological Knowledge in LM Hidden States

We begin by addressing the research question **RQ1**: *beyond performance-based evaluation* (Suvarna et al., 2024), *how well do LMs encode phonological information in their hidden states?*

To answer this question, we adopt the probing approach in Ettinger et al. (2016). Formally, given a language model $\mathcal{M}$, we prompt it with a text sequence s, tokenized into $|s|$ subwords $\{w_1, \ldots, w_{|s|}\}$. For each token w_i, at layer ℓ, the model produces a hidden state vector $\boldsymbol{h}_{i\ell} = \mathcal{M}(s, i, \ell) \in \mathbb{R}^d$, where d is the hidden dimension of the model. We take the hidden state at the last token at layer ℓ as the layer-wise representation of the entire prompt: $\boldsymbol{h}_\ell = \boldsymbol{h}_{|s|\ell}$. This follows a common starting point in decoder-only probing work: prior studies on truthfulness/factuality, sentiment, and lexical structure also extract the final-token representation and train a lightweight probe on top of it, sometimes later comparing it against alternative pooling schemes (Burns et al., 2022; Liu et al., 2024; Di Palma et al., 2025; Kaplan et al., 2025).

Given a dataset of n examples $\{s_1, \ldots, s_n\}$, which have corresponding labels $\boldsymbol{y} \in \mathbb{R}^n$ that represent certain phonological properties, we construct a probing feature set

$$\boldsymbol{X} = [\boldsymbol{h}_\ell^{(1)}, \ldots, \boldsymbol{h}_\ell^{(n)}]^\top \in \mathbb{R}^{n \times d}, \qquad (1)$$

where each $\boldsymbol{h}_\ell^{(\cdot)}$ denotes the representation of prompt s_j at layer ℓ. We then train a lightweight classifier or regressor (i.e., a probe) to predict $\boldsymbol{y}$ from $\boldsymbol{X}$. If the probe achieves performance above chance, it suggests that the LM's hidden states encode information relevant to the phonological property of interest (Hewitt and Liang, 2019). By applying this process across layers, we can analyze how phonological information evolves throughout the model hierarchy.

3.1 Task Formulation and Setups

Following Suvarna et al. (2024), we select three phonology-related tasks that collectively capture both local phonological coherence and higher-level prosodic structures. Each task is formulated as either a classification or regression problem, using input features $\boldsymbol{X}$ as defined in Equation (1).

Task 1 - Rhyming Awareness (classification): Given a pair of words, determine whether they form a perfect rhyme; that is, whether the words share the same stressed vowel and all subsequent sounds, regardless of spelling. For example, *night* (/naɪt/) and *kite* (/kaɪt/) form a perfect rhyme because the stressed vowel /aɪ/ and the following consonant /t/ coincide exactly, whereas *cough* (/kɔf/) and *tough* (/tʌf/) do not, despite their orthographic similarity. We fit a logistic regression that takes hidden states as the input to predict binary labels $y_i \in \{0, 1\}$.

To prevent models from exploiting superficial graphemic similarity, we restrict the dataset to rhyming pairs that differ in their final three orthographic characters, i.e., we will not include pairs such as *procrastination* and *verification*, which trivially rhyme due to shared suffixes. Specifically, we compile 200 positive examples of perfect rhymes satisfying this condition and 200 negative examples of non-rhyming pairs, yielding a balanced dataset for binary classification.

Task 2 - Grapheme-to-Phoneme (G2P) (regression): Convert a written word into its phonemic representation in ARPAbet (Shoup, 1980, e.g., *cat* $\rightarrow$ *K AE T*). We use the CMU Pronouncing Dictionary as the reference,[2] which provides phonemic transcriptions of English words in ARPAbet notations. In total, there are 39 possible phonemes used to describe English word pronunciations. Each phoneme is encoded as a unique index from 0 to 39, with index 0 reserved for padding. All phoneme sequences are padded to a fixed length of 8, the maximum number of phonemes per word in our dataset, yielding the output space $\mathbf{y}_i \in \mathbb{R}^8$. For each LM, we train a multi-label ridge regression model using the hidden states as input features and the phoneme indices as targets.[3] The dataset

[2] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[3] We acknowledge that this task naturally fits better with a sequence generation formulation involving discrete sequential targets. However, such approaches typically require a more complex probe that risks storing linguistic knowledge within

Rhyming Awareness Accuracy (%) by Layer Depth						G2P R^2 by Layer Depth						Syllable Counting R^2 by Layer Depth					
0%	20%	40%	60%	80%	100%	0%	20%	40%	60%	80%	100%	0%	20%	40%	60%	80%	100%
BERT (110M)																	
56.0	67.6	68.3	70.9	**71.0**	70.5	.001	**.073**	.003	.007	.031	.060	.020	**.265**	.023	.022	.110	.171
GPT-2 (124M)																	
63.4	64.7	66.1	**66.2**	66.0	61.6	.125	**.188**	.180	.159	.113	.145	.476	.624	**.629**	.621	.575	.540
GPT-neo-2.7B (2.7B)																	
68.2	68.6	**72.4**	69.6	69.7	67.0	.083	**.147**	.100	-.035	-.224	.128	.505	**.662**	.631	.514	.464	.492
Mistral-7B-Instruct-v3 (7B)																	
64.5	80.6	**80.8**	78.8	77.4	74.7	.004	.202	.245	**.315**	.276	-.092	.019	.470	.578	**.582**	.560	.312
Llama3-8B-Instruct (8B)																	
71.4	**80.7**	76.6	76.8	70.5	78.4	.041	**.263**	.234	.259	.245	.152	.144	**.661**	.660	.634	.563	.532
Llama3.1-8B-Instruct (8B)																	
72.5	**79.8**	79.0	77.9	77.3	74.9	.052	.330	.346	**.394**	.377	.189	.177	**.713**	.703	.710	.685	.616
Control Experiment: Random Embeddings																	
48.8	48.7	51.7	49.3	50.2	50.8	-.070	-.100	-.043	-.073	-.066	-.082	-.082	-.073	.001	-.115	-.097	-.022

Table 1: Probe performance (accuracy for classification, and R^2 for regression) of different LM layers on the probing tasks averaged over 10 runs, higher is better. We take Layer $\lfloor \text{depth} \times \text{percentage} \rfloor$ for each depth percentage (0%, ..., 100%), where 0% is the input embedding layer and 100% is the final output. For each task, we boldface the performance of the best layer.

consists of 2,000 words randomly sampled from the overlap between the CMU Pronouncing Dictionary and Google-10000-English, the most frequent 10,000 English words provided by Google.[4]

Task 3 - Syllable Counting (regression): Predict the number of syllables in a given word. We use the same dataset as in the G2P task, where each word is annotated with its syllable count using the same CMU-based procedure described by Suvarna et al. (2024). In this task, we fit a ridge regression model to predict the syllable count $y_i \in \mathbb{Z}_+$ from the hidden states.

3.2 Results and Discussion

We evaluate LMs with different architectures, sizes, and tokenization strategies, including BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019), GPT-neo-2.7B (Black et al., 2021), Llama3-8B, Llama3.1-8B (Grattafiori et al., 2024), Mistral-7B-v3 (Jiang et al., 2023). We adopt the cross-validation paradigm, repeating each experiment 10 times with different random seeds and reporting the average performance. In each experiment, we use 80% of the data for training and the remaining 20% for evaluation. More details on probe training and hyperparameters can be found in Appendices D and E. Following Hewitt and Liang (2019), we train probes with the same amount of random embedding examples as a control experiment to ensure that the probe is providing meaningful results. The random embedding features are of the same dimension as the LM hidden states, but are sampled from a standard normal distribution. Significantly outperforming random embedding–based probes indicates that there are indeed relevant features in the LM's hidden states.

According to Table 1, with very few exceptions in G2P (e.g., BERT at Layer 0% and GPT-neo-2.7B at Layers 60% and 80%), we find clear phonological signal in the hidden states of all tested LMs, as evidenced by their superior performance compared to the random embedding baseline across all tasks. While we observe a general trend of larger models exhibiting stronger phonological knowledge, it is not universally true nor is it consistent across all tasks (for example, while Mistral-7B-Instruct-v3 achieves the best G2P performance overall, it underperforms smaller models like GPT-2 in syllable counting).

We also observe that phonological information is most prominently encoded in the middle layers of the models, typically between 20% and 60% of the total depth, indicating that prompting-based evaluations (Suvarna et al., 2024) may underestimate the phonological knowledge present in LMs (Hu and

the probe itself, thereby confounding the interpretation (Voita and Titov, 2020). In addition, the limited size of our dataset makes training a complex model prone to overfitting. Therefore, we employ a simpler linear regression probe to both minimize potential information leakage and avoid overfitting.

[4] https://github.com/first20hours/google-10000-english

Levy, 2023). However, the overall performance remains far from perfect, suggesting that there is still considerable room for improvement.

4 Experiment 2: Explaining the Effects of Tokenization on Phonological Tasks

Having confirmed that LMs encode some imperfect phonological knowledge in their hidden states (§3), we now turn to answering **RQ2**: *do tokenization strategies affect LMs' ability to capture phonological information, and if so, how?*

4.1 Local Phonological Coherence: Exploiting Fine-Grained Tokenization for Rhymes

We hypothesize that subword-based tokenization may negatively impact an LM's ability to capture local phonological features, such as rhymes, due to its coarse granularity. Therefore, we test the hypothesis by comparing the performance of LMs with standard tokenization against a finer-grained tokenization approach that splits words into individual characters separated by delimiters (e.g., by inserting slashes, the tokenization of `boy` becomes `b / o / y` instead of `boy`). In the latter case, the model is forced to process each character separately, potentially allowing it to better capture phonological patterns like rhymes. We use this character-level splitting only as a controlled diagnostic of granularity, not as a claim that character-level tokenization is the final design choice; a syllable-aware tokenizer would be a more phonologically grounded long-term direction.

Results and Discussion. We fit logistic regression probes on the task of rhyming awareness and report accuracy with the same setup as in § 3. In addition to models assessed in Table 1, we also include ByT5-base and ByT5-small (Xue et al., 2022), which use finer-grained byte-level (i.e., sub-character level) tokenization strategies.

We find that inserting slashes to create a finer-grained tokenization improves the performance of all tested LMs across almost all layers, with statistical significance ($p < 0.05$) in most cases (Table 2). Extending the findings by Zheng et al. (2025) that non-canonical tokenizations can be handled by LMs without harm to performance across text understanding tasks, our results suggest that such tokenization strategies can even enhance the model's ability to capture phonological features.

Model Size	Format	Accuracy by Layer Depth 0%	20%	40%	60%	80%	100%
Subword Level Tokenization							
BERT							
110M	orig.	56.0	67.6	68.3	70.9	71.0	70.5
	slash	**68.6*****	**74.5****	**73.4****	**77.5****	**79.5****	**78.1****
GPT-2							
124M	orig.	63.4	64.7	66.1	66.2	66.0	61.6
	slash	**71.7*****	**76.9*****	**77.2*****	**79.1*****	**78.5*****	**77.5*****
GPT-neo-2.7B							
2.7B	orig.	68.2	68.6	72.4	69.6	69.7	67.0
	slash	**73.2**	**82.5*****	**83.9*****	**82.5*****	**81.6*****	**82.4*****
Mistral-7B-Instruct-v3							
7B	orig.	**64.5**	80.6	80.8	78.8	77.4	74.7
	slash	55.8	**81.1**	**82.7***	**79.5**	**79.0***	**77.6****
Llama3-8B-Instruct							
8B	orig.	**71.4**	80.7	76.6	76.8	70.5	**78.4**
	slash	56.3	**85.4***	**81.9****	**77.9**	**71.9**	76.1
Llama3.1-8B-Instruct							
8B	orig.	**72.5**	79.8	79.0	77.9	77.3	74.9
	slash	56.3	**85.1****	**84.0****	**80.0**	**78.9**	**79.5****
Sub-Character Level Tokenization							
ByT5-base							
580M	orig.	45.5	79.6	81.0	79.9	80.3	66.3
ByT5-small							
300M	orig.	45.5	75.5	80.1	77.6	72.7	71.8
Control Experiment: Random Embeddings							
-	-	48.8	48.7	51.7	49.3	50.2	50.8

Table 2: Accuracy (%) of rhyming awareness probing tasks across models and layers. For each model, we report results using both the original tokenization (orig.) and slash-delimited tokenization (slash). The best performance for each model at each layer is indicated in boldface. In addition, a paired t-test is conducted to assess the hypothesis that slash-delimited tokenization yields higher accuracy than the original tokenization, with significance levels indicated as * ($p < 0.05$), ** ($p < 0.01$), and *** ($p < 0.001$). Results with other delimiters (e.g., dot and comma) are with similar trends (Appendix G) and are omitted here for brevity.

4.2 Alignment between Tokens and Syllables Captures Prosodic Structure

Prosodic structure refers to the rhythmic and hierarchical organization of spoken language, including units such as syllables, stress patterns, and intonation. These features jointly shape how words are pronounced beyond their written form. To examine how tokenization may implicitly capture such prosodic regularities, we focus on two diagnostic tasks, G2P and syllable counting, as representative proxies for prosodic structure.

We hypothesize that closer alignment between tokenization and syllabification corresponds to

Model	STAD	G2P (R^2 by Layer Depth)						Syllable Counting (R^2 by Layer Depth)					
		0%	20%	40%	60%	80%	100%	0%	20%	40%	60%	80%	100%
BERT	.000 (A)	.004	.056	.001	.002	**.030**	.009	*****.999**	*****.952**	.009	.022	.129	.054
	.290 (M)	**.009**	**.085**	.001	.002	.024	**.010**	.404	.626	**.068**	**.074**	**.264**	**.143**
GPT-2	.000 (A)	*****.198**	*****.229**	*****.232**	*****.217**	*****.185**	*****.194**	.027	*****.980**	*****.980**	*****.969**	*****.952**	*****.929**
	.388 (M)	.081	.148	.146	.124	.080	.119	**.589**	.740	.732	.728	.714	.684
GPT-neo-2.7B	.000 (A)	*****.179**	****.219**	*****.211**	*****.148**	*****.078**	*****.004**	*****.945**	*****.953**	*****.967**	*****.942**	*****.930**	*****.914**
	.388 (M)	.072	.169	.111	.005	-.124	-.219	.555	.787	.758	.692	.634	.539
Mistral-7B-Instruct-v3	.000 (A)	**.001**	.212	**.301**	.314	**.297**	.239	.028	*****.800**	*****.913**	*****.911**	*****.854**	*****.816**
	.348 (M)	.000	**.214**	.283	**.317**	.282	**.261**	**.045**	.708	.804	.806	.789	.762
Llama3-8B-Instruct	.000 (A)	*****.034**	*****.349**	*****.356**	*****.370**	*****.366**	*****.325**	.152	*****.931**	*****.935**	*****.923**	*****.899**	*****.860**
	.372 (M)	.023	.295	.297	.333	.308	.276	**.165**	.769	.795	.769	.749	.717
Llama3.1-8B-Instruct	.000 (A)	**.033**	***.325**	****.321**	****.387**	****.357**	**.166**	.188	*****.936**	*****.939**	*****.921**	*****.898**	*****.859**
	.372 (M)	.029	.304	.285	.349	.317	.157	**.211**	.783	.789	.769	.754	.723
Control: Randomized Embedding		-.070	-.101	-.043	-.073	-.066	-.082	-.082	-.073	.001	-.115	-.097	-.022

Table 3: R^2 for G2P and syllable counting, comparing words with aligned (A; STAD = 0) vs. misaligned (M; STAD > 0.25) tokens and syllables. For each model and layer, the best R^2 is presented in boldface. A one-sided t-test is performed between the A and M splits at each layer to evaluate whether aligned words yield significantly better performance, with significance levels denoted as * ($p < 0.05$), ** ($p < 0.01$), and *** ($p < 0.001$). Results for more models can be found in the Appendix.

better performance on phonological tasks. To evaluate this hypothesis, we introduce a metric, the *syllabification–tokenization alignment distance* (STAD), which quantifies the degree of mismatch between tokenization and syllabification. We then compare the performance of language models (LMs) on words with high versus low STAD values.

STAD. Tokenizers segment words using frequency-based heuristics (Gage, 1994; Kudo and Richardson, 2018), which often diverge from syllable boundaries because phonological principles are not explicitly encoded in the process. The STAD metric measures this divergence, with higher values indicating greater misalignment. Consider a word with $n+1$ characters $w = a_1 a_2 \ldots a_{n+1}$, where there are n possible positions to insert a split. We represent tokenization and syllabification as two vectors, $\boldsymbol{v}_{\text{tok}} = [b_1, b_2, \ldots, b_n]$ and $\boldsymbol{v}_{\text{syl}} = [c_1, c_2, \ldots, c_n]$, where each $b_i, c_i \in \{0, 1\}$ indicates if a split occurs after the i-th character (1) or not (0). The STAD is defined as the normalized Hamming distance between $\boldsymbol{v}_{\text{tok}}$ and $\boldsymbol{v}_{\text{syl}}$:

$$\text{STAD}(\boldsymbol{v}_{\text{tok}}, \boldsymbol{v}_{\text{syl}}; w) = \frac{\sum_{i=1}^{n} |b_i - c_i|}{n}.$$

Taking the word `musical` as an example: it is syllabified as `['mu', 'si', 'cal']`, represented by $\boldsymbol{v}_{\text{syl}} = [0, 1, 0, 1, 0, 0]$. Meanwhile, if a tokenizer splits it as `['mus', 'ical']`, yielding $\boldsymbol{v}_{\text{tok}} = [0, 0, 1, 0, 0, 0]$, the STAD becomes

$$\frac{0+1+1+1+0+0}{6} = 0.5.$$

We obtain reference syllable boundaries using `syllabify`[5], a CMU Pronouncing Dictionary–based English syllabification toolkit consistent with the CMU-based preprocessing used in Suvarna et al. (2024). For each model, token boundaries are obtained by running that model's own tokenizer on the word and projecting the resulting splits back to character positions.

Data and experimental setup. We evaluate the same set of LMs with subword-level tokenization as in § 3. For each LM, we create two splits of words, the token-syllable aligned (A; STAD = 0) split and the token-syllable misaligned (M; STAD > 0.25) one, with samples of 1,000 words in each split. We train ridge regression probes on the hidden states of each layer for both G2P and syllable counting tasks, following the same setup as in § 3.

Results and Discussion. For both tasks, all LMs except BERT and Mistral-7B-Instruct-v3 show significantly better performance on words with low STAD scores at layers deeper than 20% of the model depth under the paired one-sided t-tests reported in Table 3, suggesting that tokenization structure influences the models' ability to cap-

[5] https://github.com/eoleedi/syllabify

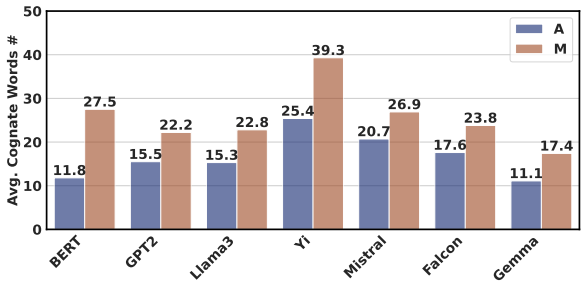


Figure 2: Average number of CogNet-related entries for token-syllable aligned words (A; STAD $= 0$) and token-syllable misaligned words (M; STAD > 0.25) for different tokenizers.

ture prosodic features. Generally, better alignment between tokenization and syllabification (i.e., lower STAD) corresponds to better performance on phonological tasks. Even for Mistral-7B-Instruct-v3, which shows no clear trend in the G2P task, performance on low-STAD words is consistently higher in the syllable counting task.

We hypothesize that the absence of a clear pattern in BERT arises from its distinct architecture—specifically, its use of bidirectional attention, which differs from the autoregressive setup of all the other models. We leave a detailed examination of this phenomenon to future work.

4.3 Conjecture: Loanwords and Cognates Cause Syllabification–Tokenization Misalignment

We propose a possible explanation for why certain words exhibit tokenizations that misalign with their syllabifications and, consequently, are more difficult for LMs to process in phonological tasks. These words may tend to display high orthographic variability in the training corpus, often stemming from historical factors such as lexical borrowing or etymological divergence. When a word is shared across languages, it is often adapted to different spelling conventions (e.g., *music* vs. *musik*), increasing orthographic variability in the training corpus. As a result, such words may contain letter n-grams that occur infrequently in the tokenizer's training data, leading subword-based tokenizers to produce segmentations that deviate from natural syllable boundaries. Taking these together, we hypothesize that words with high cross-linguistic relatedness are more likely to be tokenized in ways that misalign with their syllabification.

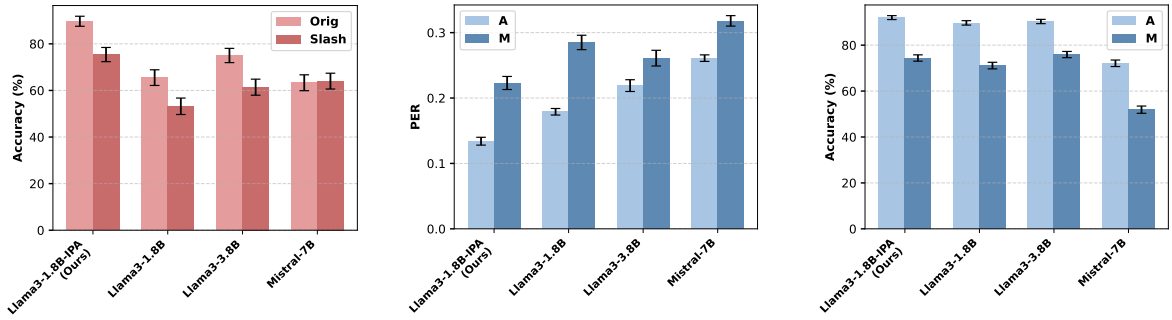
We use CogNet (Batsuren et al., 2019), a comprehensive database of cognate words and loanwords, to estimate the level of cross-linguistic relatedness for each word in our dataset. Specifically, we use the number of related entries listed in CogNet as a proxy for this relatedness, and present the average number of such entries for words in the aligned (A) and misaligned (M) groups (Figure 2). In addition to tokenizers of aforementioned models, we also include three additional tokenizers, Yi (Young et al., 2024), Falcon (Almazrouei et al., 2023), and Gemma (Mesnard et al., 2024). The results clearly indicate that words in group M systematically have a higher number of cognate or loanword variants than those in group A across all evaluated tokenizers. This finding suggests that words with extensive cross-linguistic variation are more likely to undergo tokenization that misaligns with their syllabification, thereby complicating the model's ability to capture their phonological features.

In summary, our experiments in this section demonstrate that tokenization strategies substantially influence LMs' ability to capture both local phonological features (e.g., rhymes) and global prosodic structures (e.g., syllable structures). These findings suggest that STAD can serve as a simple diagnostic for evaluating phonology-aware tokenization, and that tokenization strategies that better align with phonological principles may lead to improved phonological reasoning in LMs.

5 Experiment 3: Fine-tuning LMs for Better Phonological Reasoning

Having revealed that tokenization materially affects how LMs internally encode phonological features through probing analyses, we now move toward a practical question. Specifically, we seek to address **RQ3**: *How can we enhance LMs' phonological reasoning while largely preserving their general language understanding?*

We propose a simple yet effective strategy: fine-tuning LMs with explicit phonological awareness through a carefully curated instruction-tuning dataset. While designing a syllable-aware tokenizer would be a more direct architectural solution, replacing the tokenizer of an existing foundation model is substantially more invasive than post-training. Our goal here is therefore to provide a practical patch for widely used text-only LMs without modifying their vocabulary. The International Phonetic Alphabet (IPA; International Phonetic Association, 1999) provides a standardized and transparent representation of word pronunciation, offering direct cues to syllable structure and rhyme.



(a) Rhyming Awareness. **(b)** G2P. **(c)** Syllable Counting.

Figure 3: Performance evaluated on three phonology-related tasks.

However, prior work shows that even though LMs possess partial knowledge of IPA, they struggle to leverage it effectively in phonology-related tasks (Suvarna et al., 2024).

To overcome this limitation, we introduce a data augmentation approach that integrates IPA transcriptions into general-purpose QA examples and fine-tunes LMs to reason explicitly over IPA in phonology-related tasks, while maintaining exposure to general tasks to mitigate catastrophic forgetting (Kirkpatrick et al., 2017). For general QA data, we randomly sample 0–2 words per question and enclose them with IPA indicator tokens (i.e., `<IPA>` and `</IPA>`, inserted as two tokens into the vocabulary) to signal the model to utilize IPA information. The model's answer is then prefixed with the IPA transcription of the selected words; if no words are sampled, the example remains unchanged. Even when IPA is not semantically necessary for the QA content itself, this interleaving serves a dual purpose: it preserves exposure to the original instruction-following distribution while teaching the model to attend to phonological side information in parallel with semantics. For phonology-specific tasks, we construct conversational examples in which the answer explicitly reasons over IPA transcriptions (see Appendix C and Figures 6a to 6d for detailed templates and examples).

Fine-tuning setups. Our dataset for fine-tuning consists of two sources: (1) high-quality general instruction-tuning conversation data sampled from OpenHermes2.5 (Teknium, 2023), and (2) phonology-related tasks, where we select words out of Google-10000-English as training examples to inform the model about using IPA for the phonology-related tasks, leaving the evaluation set completely unseen during training. We

Data Type	Number of Examples
Conversation	3,000
Rhyming Awareness	200
Syllable Counting	500
G2P	500

Table 4: Number of fine-tuning examples from each source.

Model	GSM8K	MMLU
Llama3.1-8B-Instruct	69.9 ± 0.4	65.3 ± 0.5
Llama3.1-8B-IPA (Ours)	68.8 ± 0.5	64.4 ± 0.6

Table 5: Performance of Llama3.1-8B-Instruct and the fine-tuned Llama3.1-8B-IPA model on general-purpose reasoning tasks. Metrics are accuracy (mean ± standard deviation) across 3 inference runs.

fine-tune Llama3.1-8B-Instruct using low-rank adaptation (LoRA; Hu et al., 2021) on our constructed instruction-tuning dataset (with statistics presented in Table 4), and denote it as Llama3.1-8B-IPA. For evaluation, we assess three strong baselines from instruction-tuned LLMs, Llama3.1-8B-Instruct, Llama3-8B-Instruct, and Mistral-7B-Instruct-v3. Additional fine-tuning and evaluation details are provided in Appendix F.

Evaluation metrics. We evaluate the same tasks using the same datasets described in §3, conducting zero-shot inference, with prompt templates provided in the appendix. However, unlike the probing experiments, we now conduct performance-based evaluations, by directly instructing the models to output answers. We report the accuracies for the rhyming awareness and syllable counting tasks, and the phoneme error rate (PER; the Levenshtein distance between the predicted and reference over the number of phonemes in the reference) for the G2P task. Lower PER indicates better performance.

To ensure there is no catastrophic forgetting in our fine-tuned model, we also evaluate it on two widely used reasoning benchmarks: GSM8K

(Cobbe et al., 2021) and MMLU (Hendrycks et al., 2021). We employ a chat-style zero-shot evaluation to simulate real-world user interactions. For MMLU, we randomly sample three subjects from each major category (STEM, social sciences, humanities, and other), totaling 12 subjects. Results are averaged over 3 runs using decoding parameters `top-p`=0.95, `temperature`=0.8.

Results and discussion. As presented in Figure 3, fine-tuning on the IPA-augmented dataset consistently improves performance across all three phonology-related tasks. For the two prosodic structure tasks (G2P and syllable counting), LMs perform considerably better on words with low STAD scores (A), indicating that tokenization structure also affects inference performance, consistent with our findings in representation-level probing (§4). For the rhyming awareness task, however, we find that simply inserting slashes into words does not necessarily improve performance, suggesting that tokenization alone is not sufficient to enhance phonological reasoning at the performance level.

Despite 1.1% and 0.9% drops on GSM8K and MMLU, respectively, our fine-tuned model, Llama3.1-8B-IPA, retains most of its general reasoning and knowledge abilities (Table 5) without clear signs of catastrophic forgetting. The trade-off is therefore not zero-cost, but favorable for our goal of specialized phonological adaptation. Our objective here is not to improve general reasoning benchmarks per se, but to show that interleaving IPA signals with general instruction data can inject phonological knowledge while maintaining most general capability. This may be attributed to the domain mismatch between the fine-tuning dataset and the evaluation benchmarks, our limited hyperparameter tuning due to limited compute budget, as well as the relatively small capacity of the base model (compared to the commercial ones). We remain fairly optimistic that other advanced techniques, such as reinforcement learning–based ones, can be explored to further mitigate forgetting while enhancing phonological reasoning.

6 Conclusion and Discussion

We analyze a set of LMs on three tasks that cover both local and hierarchical phonological structures, and show that tokenization can introduce systematic biases in word representation and thereby limit phonological performance. More broadly, these results suggest that phonological failures in LMs need not reflect a complete absence of phonological knowledge; much of the difficulty appears to arise at the interface between subword segmentation and phonological structure. STAD turns this observation into a concrete diagnostic for tokenizer evaluation, and our cognate analysis suggests one linguistic source of such misalignment, though the full causal relationship remains an open question. Together with the fine-tuning results, these findings motivate two complementary paths forward: tokenizer design that better respects phonological boundaries, and lightweight post-training with explicit phonological supervision.

These lessons may also inform joint speech and text language models (e.g., Chou et al., 2023, *inter alia*), by highlighting the importance of text tokenizers that preserve phonological structure, though direct comparisons with speech or multimodal systems remain future work. We do not see risks beyond the minimal risks of any research in computational linguistics.

Limitations

First, it is worth noting that our experiments have been primarily focused on English, a representative alphabetic language. Findings in this work need significant work to be possibly adaptable to logographic languages. We leave the exploration of a broader range of modal architectures and additional languages for future work.

Second, most of the analytical experiments in this work are correlational in nature, and we acknowledge that there are many confounding factors that may affect the causality of the observed effects, particularly when it comes to hypotheses related to cognates. We encourage future work to further investigate these relationships.

Finally, while our proposed fine-tuning method has demonstrated effectiveness in enhancing phonological understanding, it is important to note that the improvements are not uniform across all evaluated models. In addition, due to the compute constraint, there is no guarantee that findings from this work generalize to larger, proprietary models. We leave a more comprehensive evaluation of the proposed analysis (§§3 and 4) and method (§5) on a wider range of models to future work.

Acknowledgment

We thank three anonymous reviewers and the area chair for the valuable suggestions. This work is

supported in part by an NSERC Discovery Grant (RGPIN-2024-04395) and a Canada CIFAR AI Chair Award to FS.

References

Guillaume Alain and Yoshua Bengio. 2018. Understanding intermediate layers using linear classifier probes, 2017. In *URL https://openreview. net/forum*.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models. *Preprint*, arXiv:2311.16867.

Morris Alper and Hadar Averbuch-Elor. 2024. Kiki or bouba? sound symbolism in vision-and-language models. *Advances in Neural Information Processing Systems*, 36.

Amos Azaria and Tom Mitchell. 2023. The internal state of an llm knows when it's lying. *arXiv preprint arXiv:2304.13734*.

Khuyagbaatar Batsuren, Gábor Bella, Fausto Giunchiglia, et al. 2019. Cognet: A large-scale cognate database. In *ACL 2019 The 57th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pages 3136–3145. Association for Computational Linguistics.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.

Euan Bonner, Ryan Lege, and Erin Frazier. 2023. Large language model-based artificial intelligence in the language classroom: Practical ideas for teaching. *Teaching English with Technology*, 23(1):23–41.

Bastian Bunzeck, Daniel Duran, Leonie Schade, and Sina Zarrieß. 2024. Small Language Models Like Small Vocabularies: Probing the Linguistic Abilities of Grapheme- and Phoneme-Based Baby Llamas. *arXiv preprint*. ArXiv:2410.01487.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2022. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT press.

Ju-Chieh Chou, Chung-Ming Chien, Wei-Ning Hsu, Karen Livescu, Arun Babu, Alexis Conneau, Alexei Baevski, and Michael Auli. 2023. Toward joint language modeling for speech units and text. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Dario Di Palma, Alessandro De Bellis, Giovanni Servedio, Vito Walter Anelli, Fedelucio Narducci, and Tommaso Di Noia. 2025. LLaMAs have feelings too: Unveiling sentiment and emotion representations in LLaMA models through probing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6124–6142, Vienna, Austria. Association for Computational Linguistics.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 134–139.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Wes Gurnee and Max Tegmark. 2023. Language models represent space and time. *arXiv preprint arXiv:2310.02207*.

Vita A Hamaniuk. 2021. The potential of large language models in language education. *Educational Dimension*, 5:208–210.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Jennifer Hu and Roger Levy. 2023. Prompting is not a substitute for probability measurements in large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5040–5060, Singapore. Association for Computational Linguistics.

International Phonetic Association. 1999. *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Guy Kaplan, Matanel Oren, Yuval Reif, and Roy Schwartz. 2025. From tokens to words: On the inner lexicon of LLMs. In *The Thirteenth International Conference on Learning Representations*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Junteng Liu, Shiqi Chen, Yu Cheng, and Junxian He. 2024. On the universal truthfulness hyperplane inside LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18199–18224, Miami, Florida, USA. Association for Computational Linguistics.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Jerry Ngo and Yoon Kim. 2024. What do language models hear? probing for auditory representations in language models. *arXiv preprint arXiv:2402.16998*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Mahta Fetrat Qharabagh, Zahra Dehghanian, and Hamid R. Rabiee. 2024. LLM-Powered Grapheme-to-Phoneme Conversion: Benchmark and Case Study. *arXiv preprint*. ArXiv:2409.08554.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

June E Shoup. 1980. Phonological aspects of speech recognition. *Trends in speech recognition*, pages 125–138.

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

Kaiser Sun, Peng Qi, Yuhao Zhang, Lan Liu, William Wang, and Zhiheng Huang. 2023. Tokenization consistency matters for generative models on extractive NLP tasks. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13300–13310, Singapore. Association for Computational Linguistics.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. 2020. Adv-bert: Bert is not robust on misspellings! generating nature adversarial samples on bert. *arXiv preprint arXiv:2003.04985*.

Mukuntha Narayanan Sundararaman, Ayush Kumar, and Jithendra Vepa. 2021. Phoneme-bert: Joint language modelling of phoneme sequence and asr transcript. *arXiv preprint arXiv:2102.00804*.

Ashima Suvarna, Harshita Khandelwal, and Nanyun Peng. 2024. PhonologyBench: Evaluating Phonological Skills of Large Language Models. *arXiv preprint*. ArXiv:2404.02456.

Teknium. 2023. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, et al. 2024. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*.

Chengyue Yu, Lei Zang, Jiaotuan Wang, Chenyi Zhuang, and Jinjie Gu. 2024. Charpoet: A chinese classical poetry generation system based on token-free llm. *arXiv preprint arXiv:2401.03512*.

Guangyan Zhang, Kaitao Song, Xu Tan, Daxin Tan, Yuzi Yan, Yanqing Liu, Gang Wang, Wei Zhou, Tao Qin, Tan Lee, et al. 2022. Mixed-phoneme bert: Improving bert with mixed phoneme and sup-phoneme representations for text to speech. *arXiv preprint arXiv:2203.17190*.

Ran Zhang and Steffen Eger. 2024. Llm-based multi-agent poetry generation in non-cooperative environments. *arXiv preprint arXiv:2409.03659*.

Brian Siyuan Zheng, Alisa Liu, Orevaoghene Ahia, Jonathan Hayase, Yejin Choi, and Noah A Smith. 2025. Broken tokens? your language model can secretly handle non-canonical tokenizations. *arXiv preprint arXiv:2506.19004*.

A Cognates, Loanwords & CogNet

Words	Language
muzikál	Czech
mjuzikl	Croatian
musikalsk	Norwegian
musikalsk	Danish
мьюзикл	Mongolian
musical	French
musikal	Basque
muzikálny	Slovak
մյուզիքլ	Armenian
mużikali	Maltese
musical	Galician
musical	Catalan
müzikal	Turkish
musikal	Swedish
músíkalskur	Icelandic
musical	Italian
musikaali	Finnish
musical	Dutch
musical	German
мюзикл	Russian
musikal	Indonesian
musical	English
музикален	Bulgarian
muzikalan	Crotian
muzika	Esperanto
musikalisk	Swedish
muzikaal	Dutch
muzikalen	Slovenian
musical	Spanish
muzikal	Malay
musical	Portuguese
muzical	Romanian
מְיוּזִיקָל	Hebrew
მიუზიკლი	Georgian

Figure 4: CogNet entries related to "musical" across multiple languages, including both cognates and loanwords. The dataset includes phonetically and orthographically similar words from multiple language families.

Cognate words are words in different languages that share a common etymological origin, whereas loanwords are words borrowed across languages. Both can exhibit similar spellings and meanings, making them useful in linguistic studies and multilingual natural language processing.

CogNet (Batsuren et al., 2019) is a linguistic resource that provides structured data on cognate words and loanwords across multiple languages. It is built using a combination of linguistic datasets and automated methods for identifying words with shared etymology. The dataset includes words from a wide variety of language families, offering a valuable resource for comparative linguistics and language modeling. In this study, we used CogNet as a dictionary that gives related cognate/loanword forms of an input word. For instance, the English word *musical* is tokenized as `[mus', ical']` by the Llama3 tokenizer, a segmentation that does not align well with its phonological structure. This misalignment may stem from the word's extensive cross-linguistic variation, as *musical* has numerous related forms across different languages. Some of these, such as *muzikal* in Czech and *mjuzikl* in Croatian, exhibit distinct orthographic representations that may influence tokenization irregularities.

CogNet identifies cognates using a combination of phonetic similarity, orthographic resemblance, and historical linguistic data. The process typically involves phonetic matching to identify words with similar pronunciation patterns across languages, orthographic similarity to detect common roots based on spelling, and etymological analysis leveraging linguistic databases and historical texts to verify word origins. Additionally, the dataset is refined using both automated algorithms and manual linguistic validation.

Figure 4 is an example of cognate words for the English word *musical*, as retrieved from CogNet. The table includes translations in multiple languages, showing how the word has evolved across different linguistic groups.

B Evaluate Prompt For Phonology Inference

In this section, we demonstrate the prompt we used to evaluate the phonology three phonology-related tasks.

C Phonology-related Task Training Template

In Algorithms 1 to 4, $\texttt{fill}(T, x)$ denotes simple template instantiation: it replaces the placeholders

G2P Prompt

ARPAbet is a phonetic transcription system used to represent the pronunciation of words Below are the ARPAbet symbols

Vowels:
AA, AE, AH, AO, AW, AY, EH, ER, EY, IH, IY, OW, OY, UH, UW

Consonats:
B, CH, D, DH, F, G, HH, JH, K, L, M, N, NG, P, R, S, SH, T, TH, V, W, Y, Z, ZH

Provide ARPAbet transcription using only the symbols above, add space between each phoneme. Transcribe the following word, output the answer as 'ARPAbet: <phoneme sequence>'.
Word: {word}

(a) The prompt template we used to evaluate the G2P task.

Rhyming Awareness Prompt

Rhyming words are words that have the same ending sound. Determine if the following two words are in Rhyme.
{word1}, {word2}
Give the answer as "Answer: True" if they rhyme and "Answer: False" if they do not.

(b) The prompt template we used to evaluate the rhyming awareness task.

Syllable Counting Prompt

Count the number of syllables in the word: '{word}'
Give the answer in the format "Answer: <number of syllables>"

(c) The prompt template we used to evaluate the syllable counting task.

in a question or answer template T with the provided content x (e.g., IPA transcription of sampled words).

Here, we demonstrate the detailed training template we used for constructing the training dataset for each problem. We demonstrate how we construct 4 categories of QA pairs as our fine-tuning dataset.

Conversation. We used OpenHermes2.5 (Teknium, 2023) as the source of the conversation dataset, it involves all kinds of conversational datasets consisting of question and answer. In the question, we randomly select 0 - 2 words and wrap the words with an IPA token to indicate that we want the IPA of the word, and in the answer, we add one sentence indicating the IPA of the words. This augmentation is intentionally orthogonal to the QA semantics: it preserves the original task while training the model to attend to phonological side information. We present the process of data creation in Algorithm 1 and show an example in

Algorithm 1 Conversation Data Creation with IPA Annotations

Require: Dataset $\mathcal{D}$ with question-answer pairs (q, a), IPA sentence template L. Function $\texttt{fill}(T, w)$ to fill a template T using word w. Function `get_IPA` to get IPA.
Ensure: Modified dataset $\mathcal{D}'$ with IPA-annotated questions and answers

$\mathcal{D}' \leftarrow \emptyset$
for each $(q, a) \in \mathcal{D}$ **do**
  Split q into words: $W \leftarrow \text{split}(q)$
  $q' \leftarrow q$
  Sample $k \sim \text{Uniform}(\{0, 1, 2\})$
  Uniformly sample $S \subset W$ with $|S| = k$
  $S_{\text{IPA}} \leftarrow \emptyset$
  for each selected word $w_i \in S$ **do**
    Replace w_i in q' with <IPA>w_i</IPA>.
    Obtain IPA transcription of w_i: $p_i = \text{get\_IPA}(w_i)$
    Add (w_i, p_i) to S_{IPA}
  end for
  filling in words from S_{IPA} into L, $l = \text{fill}(L, S_{\text{IPA}})$
  Prepend sentence l indicating IPA representation to a: $a' \leftarrow l + a$
  Add modified pair (q', a') to $\mathcal{D}'$
end forreturn $\mathcal{D}'$

Algorithm 2 Dataset Creation for Rhyming Awareness Task

Require: Word pair list with IPA transcriptions, possible templates $P_1, \ldots, P_5$. Positive answer template A_P, negative answer template A_N. Function $\texttt{fill}(T, w)$ to fill a template T using word w.
Ensure: Dataset with question-answer pairs

for each $(\text{word}_1, \text{word}_2)$ pair **do**
  Sample $i \sim \text{Uniform}(\{1, 2, 3, 4, 5\})$
  $P \leftarrow \text{fill}(P_i, (\text{word}_1, \text{word}_2))$
  Extract IPA endings of word_1 and word_2
  if IPA endings match **then**
    response $\leftarrow \text{fill}(A_P, (\text{word}_1, \text{word}_2))$
  else
    response $\leftarrow \text{fill}(A_N, (\text{word}_1, \text{word}_2))$
  end if
  Store $(P, \text{response})$ in dataset
end for

Algorithm 3 Dataset Creation for Grapheme-to-Phoneme (G2P) Task

Require: Word list with IPA transcriptions, possible templates $P_1, \ldots, P_5$. Answer template A. ARPAbet-to-phoneme dictionary M. Function $\texttt{fill}(T, w)$ to fill a template T using word w.
Ensure: Dataset with question-answer pairs

for each word w in dataset **do**
  Sample $i \sim \text{Uniform}(\{1, 2, 3, 4, 5\})$
  $P \leftarrow \text{fill}(P_i, w)$
  Obtain IPA transcription of w: $I = \text{get\_IPA}(w)$
  IPA phonemes to ARPAbet: $A = [M[p] \text{ for } p \in I]$
  response $\leftarrow \text{fill}(A, (w, I, A))$
  Store $(P, \text{response})$ in dataset
end for

Algorithm 4 Dataset Creation for Syllable Counting Task

Require: Word list with IPA transcriptions, possible templates $P_1, \ldots, P_5$. Answer template A. Function $\texttt{fill}(T, w)$ to fill a template T using word w.
Ensure: Dataset with question-answer pairs
1: **for** each word w in dataset **do**
2: Sample $i \sim \text{Uniform}(\{1, 2, 3, 4, 5\})$
3: $P \leftarrow \text{fill}(P_i, w)$
4: Obtain IPA transcription of w: $I = \text{get\_IPA}(w)$
5: Identify vowel nuclei (monophthongs or diphthongs) in I
6: Compute syllable count: $S = \text{count\_syllables}(I)$
7: Format response using S: response $\leftarrow \text{fill}(A, (w, S))$
8: Store $(P, \text{response})$ in dataset
9: **end for**

Figure 6a.

Rhyming Awareness. In the rhyming awareness, we prepare 5 possible question templates $P_1, P_2, \ldots P_5$ to mimic the possible users' questions. In the answer, we first give the IPA of the word as in Conversation. Then, from the IPA, we extract the same part of the IPA if two words are in rhyme, or state two words are not in rhyme if the IPA does not have the same ending. We present the process of rhyming awareness data creation in Algorithm 2 and we show the question templates and example answer in Figure 6b.

G2P. In the G2P task, we also prepare 5 possible question templates. In the answer, we break the IPA transcript phoneme by phoneme and use the map from phoneme to ARPAbet to convert the IPA to ARPAbet. We present the process of G2P data creation in Algorithm 3 and the question templates and an example answer in Figure 6c.

Syllable Counting. In the syllable counting task, there are also 5 possible questions as before. In the solution, we use a simple English IPA heuristic: each vowel nucleus (a monophthong or diphthong) contributes one syllable, so counting these nuclei gives the total number of syllables. We present the process of syllable counting data creation in Algorithm 4 and the question templates and an example answer in Figure 6d.

D Controlled-Experiment of the Probing

To verify that our linear probes do not artificially inflate performance, we repeat every probing experiment with *randomly generated targets*. For rhyming awareness, we assign a random binary label to each word-pair; for grapheme-to-phoneme (G2P) prediction we draw a random integer in the range $[0, 39]$ for every phoneme slot; and for syllable counting, we sample a random integer between 0 and 8. We train the same logistic- and ridge-regression probes as in the main study on GPT-2

Q:**<IPA>Every</IPA>** *day, a tree drops 7 leaves. How many* **<IPA>leaves</IPA>** *would it drop in a month of February in a non-leap year? Include your logic.*

A: **word Every has IPA /ˈɛv.(ə.)ɹi/ and leaves has IPA /liːvz/.**
Here's the logic behind this:\n\n1. We know that February has 28 days in a non-leap year.
2. If the tree drops 7 leaves every day, then over the course of February, it would drop: Leaves dropped in February = Leaves per day * Days in February = 7 leaves * 28 days = 196 leaves.
So, the tree would drop 196 leaves in February in a non-leap year

(a) An example of how we construct QA data from OpenHermes2.5.

P1: *Consider words <IPA>{word1}</IPA> and <IPA>{word2}</IPA>. Do they rhyme?*
P2: *Are the following words in rhyme? <IPA>{word1}</IPA>, <IPA>{word2}</IPA>?*
P3: *Does word <IPA>{word1}</IPA> rhyme with word <IPA>{word2}</IPA>?*
P4: *Is word <IPA>{word1}</IPA> and <IPA>{word2}</IPA> in rhyme?*
P5: *Rhyming words are words that have the same ending sound. Is word <IPA>{word1}</IPA> in rhyme with word <IPA>{word2}</IPA>?*

Example Answer 1: cat has IPA /kæt/ and hat has IPA /hæt/.
From the IPA transcriptions, cat and hat have the same ending sound /æt/, therefore they are in rhyme.
Answer: Yes
Example Answer 2 : rain has IPA /ɹeɪn/ and bloom has IPA /bluːm/.
From the IPA transcriptions, rain and bloom have different ending sounds, therefore they are not in rhyme.
Answer: No

(b) All possible questions and example answers for the rhyming awareness task.

P1: *Give the ARPAbet transcription of the following word.*
Word: <IPA>{word}</IPA>"
P2: *Convert the following word into ARPAbet.*
Word: <IPA>{word}</IPA>
P3: *ARPAbet is a phonetic transcription system used to represent the pronunciation of words. Below are the ARPAbet symbols:*
Vowels:
AA, AE, AH, AO, AW, AY, EH, ER, EY, IH, IY, OW, OY, UH, UW
Consonants:
B, CH, D, DH, F, G, HH, JH, K, L, M, N, NG, P, R, S, SH, T, TH, V, W, Y, Z, ZH
Provide ARPAbet transcriptions using only the symbols above, add space between each phoneme.
Now, transcribe the following word, output the answer as 'ARPAbet: <phoneme sequence>' and stop generating after the answer.
Word: <IPA>{word}</IPA>
P4: *ARPAbet is a phonetic transcription system used to represent the pronunciation of words. for example, if the word is 'cat', the ARPAbet transcription is 'K AE T'. If the word is 'university', the ARPAbet transcription is 'Y UW N AH V ER S AH T IY'. What is the ARPAbet transcription of the following word?*
Word: <IPA>{word}</IPA>
P5:*ARPAbet is a phonetic transcription system used to represent the pronunciation of words. For example:*
Word: cat
ARPAbet: K AE T
Word: dog
ARPAbet: D AW G
What is the ARPAbet transcription of the following word?
Word: <IPA>{word}</IPA>

Example Answer:
ideology has IPA /aɪdiɑlədʒi/.
From the IPA transcription, we can look at each phoneme and find the corresponding ARPAbet transcription:
aɪ corresponds to AY
d corresponds to D
i corresponds to IY
ɑ corresponds to AA
l corresponds to L
ə corresponds to AH
dʒ corresponds to JH
i corresponds to IY
ARPAbet: AY D IY AA L AH JH IY

(c) All possible questions and an example answer for G2P task.

P1: *How many syllables does the word <IPA>{word}</IPA> have?*
P2: *How many syllables does the word <IPA>{word}</IPA> have?*
Example:
Word: cat
Answer: 1
Word: take
Answer: 1
Word: <IPA>{word}</IPA>
Answer:
P3: Count the number of syllables in the word <IPA>{word}</IPA>.
Example:
Word: cat
Answer: 1
Word: take
Answer: 1
Word: <IPA>{word}</IPA>
Answer:
P4: Count the number of syllables in the word <IPA>{word}</IPA>.
Give the answer in the format:
Answer: <number of syllables>
P5: Count the number of syllables in the word <IPA>{word}</IPA>.
The give the answer as 'Answer: <number of syllables>

Example Answer:
struggling has IPA /ˈstɹʌglɪŋ/.
From the IPA transcription, the vowels are /ʌ/ and /ɪ/
The number of syllables in the word is 2
Answer: 2

(d) All possible questions and an example answer for syllable counting task.

Figure 6: Examples of our question templates and some example answers. The yellow part is the common part of the fine-tuning dataset, which helps the model to identify which word to consider IPA and give the IPA explicitly.

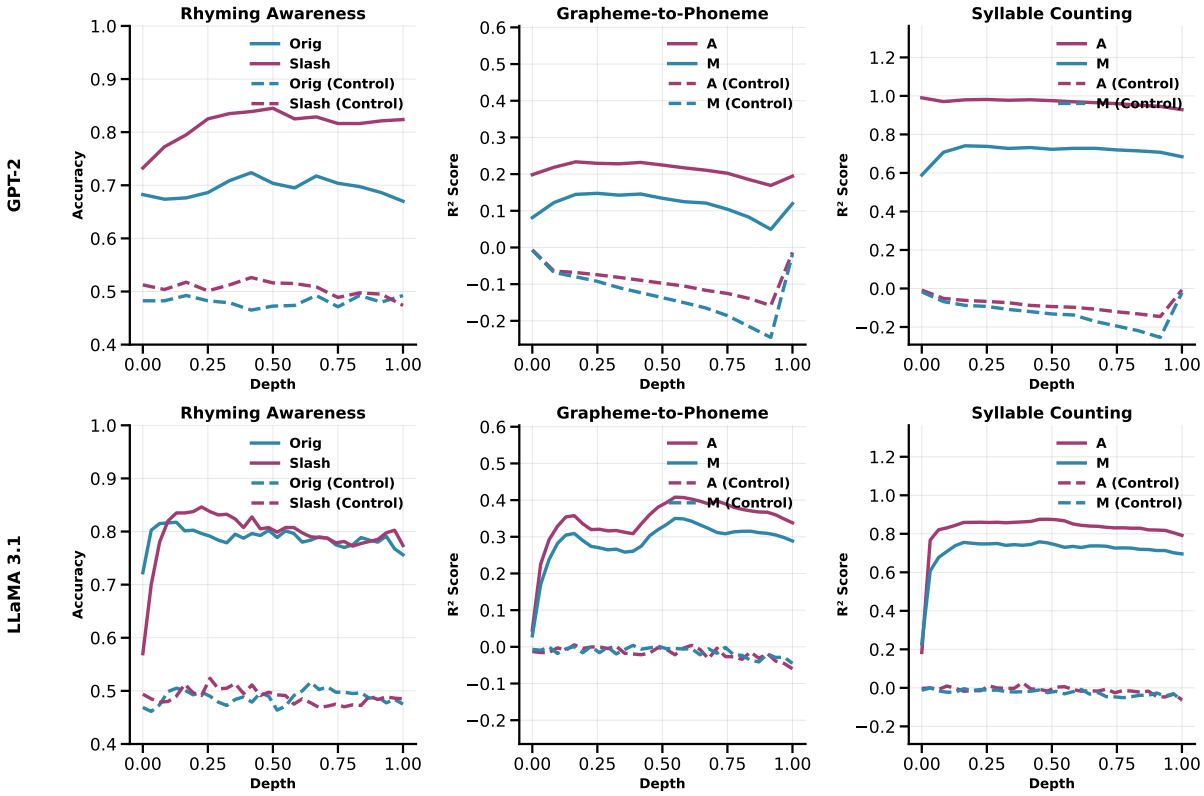


Figure 7: **Control-label sanity check.** Probing performance with *random* targets for GPT-2 (upper block) and Llama-3.1-8B (lower block). Left-to-right: (i) Rhyming awareness–accuracy; (ii) G2P–R^2; (iii) syllable counting–R^2. Solid lines reproduce the original probes, dashed lines the corresponding control probes. All curves collapse to chance (accuracy ≈ 0.5) or sub-chance ($R^2 \leq 0$), demonstrating that the linear probes do not overfit when the target carries no linguistic signal.

and Llama-3.1-8B using these synthetic labels.

As summarized in Figure 7, the control probes behave exactly as expected: accuracy hovers around the chance rate of 0.5 for the binary classification task, and all R^2 values for the two regression tasks are zero or negative across layers. This confirms that the probes themselves lack the capacity to memorize arbitrary labelings and that the positive results reported in the main paper genuinely stem from information encoded in the models' hidden representations rather than from overfitting artifacts.

E Probing Details

If we have n words/pairs of words as input, after prompting them to LMs, for each layer l, we will get a matrix of $\boldsymbol{H}_l \in \mathbb{R}^{n \times d}$, where d is the dimension of the hidden states. Then, if we have the ground truth $\boldsymbol{y}$, we can train models using $\boldsymbol{H}_l$ and $\boldsymbol{y}$, and we will discuss the details of our probing for each task. For the model we trained, we used the scikit-learn (Pedregosa et al., 2011) implementation. For each experiment, we ran 10 times with seeds 0 - 9, and did an 80 - 20 train-test split, reporting the metrics on the test set. We only selected a linear model for evaluation, since the goal of our work is not to achieve high performance on the downstream task but to illustrate the bias introduced by the tokenizers. Also, Hewitt and Liang (2019) illustrated that using a complex model like a Neural Network may cause the probing result to be unreliable since the model will learn the feature, and our evaluated tasks are not very hard tasks, thus, the linear model is enough to reveal the representation quality of different words. To obtain the embeddings, we use a single A40 GPU and do a single forward pass.

Rhyming Awareness. For the rhyming awareness task, the input is a pair of words, and ground truth $\boldsymbol{y} \in \mathbb{R}^n$ is a binary label indicating if the words pair is rhyming. Then we used logistic regression `LogisticRegression`, and we set the max iterations to 1000, and Inverse of regularization strength $C = 10$, other hyperparameters are set as default. We trained two logistic regression classifiers on both original words and words with slash inserted.

G2P. For the G2P task, the label is the categorical encoding of the ARPAbet symbols (0 - 39), we

either truncated or padded the label with 0. Therefore, we have $\boldsymbol{Y} \in \mathbb{R}^{n \times 8}$. We used the Cross-validation Ridge regression `RidgeCV` to regress the label and set the alphas to be chosen from $\{10, 100, 500, 1000, 2000\}$, other hyperparameters are set as default. We train two ridge regressors on both syllable-token aligned and misaligned groups.

Syllable Counting. For the syllable counting task, the label is the number of syllables in the word. Therefore, we have $\boldsymbol{y} \in \mathbb{R}^{n}$. We also used Cross-validation Ridge regression `RidgeCV` to regress the label and set alphas to be chosen from $\{10, 100, 500, 1000, 2000\}$, other hyperparameters are set as default. We train two ridge regressors on both syllable-token aligned and misaligned groups.

F Fine-tuning & Evaluation Details

For the evaluation, we used the chat template of the corresponding model to form the QA. And we used vllm (Kwon et al., 2023) and set the decoding strategy with `top-p`=0.95 and `temperature`=0.8.

We do not report task-specific non-neural baselines in the main performance-based evaluation because rhyming awareness and syllable counting labels are derived from CMU-based dictionary/rule preprocessing, making lookup-based systems near-oracle label generators, whereas dedicated G2P systems are separately supervised models and are therefore not directly comparable to zero-shot evaluation of pretrained foundation models.

For the fine-tuning, we leveraged the Hugging Face `transformers` library alongside Parameter-Efficient Fine-Tuning (PEFT) to integrate LoRA (Hu et al., 2021). We specifically targeted the query (q_proj) and value (v_proj) projection layers for adaptation. We set the LoRA Rank (r) to 8, LoRA scaling factor (α) to 16, LoRA Dropout to 0.1.

For multi-GPU training, we employed Hugging Face Accelerate, which facilitated seamless distributed training across the two A40 GPUs using Pytorch Distrubuted Data Parallel (DDP) for less than 5 GPU hours. The model and dataset were automatically partitioned and synchronized, ensuring efficient computation.

G Rhyming probing using Different Delimiters

In the rhyming awareness probing experiment, we initially used the slash ("/") as a delimiter to split word pairs, enabling more structured and representative hidden states. To assess the robustness of this delimiter choice—and to test whether performance gains stem from improved tokenization granularity rather than the specific symbol—we conducted an ablation study using alternative delimiters: the comma (",") and the dot (".").

Model Size	Delimiter	Depth (Accuracy ↑) 0%	20%	40%	60%	80%	100%
Sub-word Tokenization							
BERT							
110M	None	56.0	67.6	68.3	70.9	71.0	70.5
	Slash	68.6	74.5	73.4	77.5	79.5	78.1
	Comma	68.6	75.7	71.8	77.8	80.3	79.2
	Dot	68.6	74.7	73.6	80.8	79.8	78.4
GPT-2							
124M	None	63.4	64.7	66.1	66.2	66.0	61.6
	Slash	71.7	76.9	77.2	79.1	78.5	77.5
	Comma	71.7	77.6	77.1	78.8	77.9	77.3
	Dot	72.3	77.3	77.6	79.5	78.4	76.9
Llama-3.1-8B-Instruct							
8B	None	72.5	79.8	79.0	77.9	77.3	74.9
	Slash	56.3	85.1	84.0	80.0	78.9	79.5
	Comma	56.8	86.7	83.7	79.3	78.5	77.4
	Dot	56.7	85.2	82.0	79.8	77.8	76.3
Mistral-7B-Instruct-v3							
7B	None	64.5	80.6	80.8	78.8	77.4	74.7
	Slash	55.8	81.1	82.7	79.5	79.0	77.6
	Comma	55.8	81.7	85.4	82.1	80.3	79.5
	Dot	55.8	80.5	81.7	79.1	77.9	77.2

Table 6: Ablation study of delimiter formats ("None", "Slash", "Comma", "Dot") across different depths of the hidden states.

We evaluated probing performance across four language models—BERT, GPT-2, LLaMA3.1-8B, and Mistral-7B—and report results in Table 6. Across all models, the probers trained with any delimiter (slash, comma, or dot) yield comparable performance throughout the depth of the hidden layers. Importantly, all delimiter-based variants consistently outperform the baseline where no delimiter is used (None), confirming that the performance gains are primarily due to the introduction of fine-grained structure in the input rather than the specific choice of delimiter.

H R^2 on G2P and syllable counting for More Models

We report the R^2 of G2P and syllable counting probing for more models in Table 7. All additional models exhibit similar trends to the main models discussed in the paper.

I LLM Usage

LLMs were used to assist with text refinement and data formatting, but not for generating core research content.

Model	STAD	G2P (R^2 by Layer Depth)						Syllable Counting (R^2 by Layer Depth)					
		0%	20%	40%	60%	80%	100%	0%	20%	40%	60%	80%	100%
BERT	.000 (A)	.004	.056	.001	.002	**.030**	.009	*****.999**	*****.952**	.009	.022	.129	.054
	.290 (M)	**.009**	**.085**	.001	.002	.024	**.010**	.404	.626	**.068**	**.074**	**.264**	**.143**
GPT-2	.000 (A)	*****.198**	*****.229**	*****.232**	*****.217**	*****.185**	*****.194**	.027	*****.980**	*****.980**	*****.969**	*****.952**	*****.929**
	.388 (M)	.081	.148	.146	.124	.080	.119	**.589**	.740	.732	.728	.714	.684
BLOOM-560M	.000 (A)	.058	****.238**	***.231**	**.222**	**.214**	****.193**	.476	*****.947**	*****.966**	*****.950**	*****.936**	*****.922**
	.376 (M)	**.096**	.196	.215	.215	.199	.168	**.489**	.766	.753	.711	.674	.608
GPT-neo-2.7b	.000 (A)	*****.179**	****.219**	*****.211**	*****.148**	*****.078**	*****.004**	*****.945**	*****.953**	*****.967**	*****.942**	*****.930**	*****.914**
	.388 (M)	.072	.169	.111	.005	-.124	-.219	.555	.787	.758	.692	.634	.539
Mistral-7b-Instruct-v3	.000 (A)	**.001**	.212	**.301**	.314	**.297**	.239	.028	*****.800**	*****.913**	*****.911**	*****.854**	*****.816**
	.348 (M)	.000	**.214**	.283	**.317**	.282	**.261**	**.045**	.708	.804	.806	.789	.762
gemma-7b	.000 (A)	*****.168**	***.279**	**.247**	**.260**	***.312**	**.193**	.383	*****.921**	*****.934**	*****.976**	*****.946**	***.782**
	.303 (M)	.054	.229	.231	.226	.284	.183	**.640**	.773	.769	.772	.758	.672
Llama3-8b-Instruct	.000 (A)	*****.034**	*****.349**	*****.356**	*****.370**	*****.366**	*****.325**	.152	*****.931**	*****.935**	*****.923**	*****.899**	*****.860**
	.372 (M)	.023	.295	.297	.333	.308	.276	**.165**	.769	.795	.769	.749	.717
Llama3.1-8b-Instruct	.000 (A)	**.033**	***.325**	****.321**	****.387**	****.357**	**.166**	.188	*****.936**	*****.939**	*****.921**	*****.898**	*****.859**
	.372 (M)	.029	.304	.285	.349	.317	.157	**.211**	.783	.789	.769	.754	.723
Falcon3-7b-Instruct	.000 (A)	*****.100**	****.209**	****.192**	*****.153**	*****.151**	*****.149**	.419	*****.974**	*****.977**	*****.975**	*****.975**	*****.977**
	.337 (M)	.050	.148	.155	.054	.004	.025	**.618**	.776	.769	.734	.728	.729
	.000 (A)	.056	**.240**	**.269**	***.245**	****.210**	****.189**	.252	****.925**	*****.937**	*****.940**	*****.941**	*****.936**
Control: Randomized Embedding		-.070	-.101	-.043	-.073	-.066	-.082	-.082	-.073	.001	-.115	-.097	-.022

Table 7: R^2 for G2P and syllable counting, comparing words with aligned (A; STAD = 0) vs. misaligned (M; STAD > 0.25) tokens and syllables. For each model and layer, the best R^2 is presented in boldface. A one-sided t-test is performed between the A and M splits at each layer to evaluate whether aligned words yield significantly better performance, with significance levels denoted as * ($p < 0.05$), ** ($p < 0.01$), and *** ($p < 0.001$). Results for more models can be found in the Appendix.

J License and Terms for Use of Artifacts

We rely on publicly released datasets and pretrained models, used strictly under their original terms. Specifically, we use the *CMU Pronouncing Dictionary* (BSD-2-Clause), the *Google-10000 English* word list (MIT), *MMLU* (MIT), and *GSM8K* (MIT). We also use *CogNet*, which is released under CC BY-NC-SA 4.0; accordingly, we treat its content as non-commercial and attribution-preserving and make no redistribution beyond what the license permits. For the *OpenHermes 2.5* compilation, no single uniform license is specified; the maintainers note heterogeneous upstream licenses and a placeholder policy, so we handle it as research-only, cite sources, and do not redistribute any of its contents. For pretrained models, our usage adheres to each model's license: *BERT* (Apache-2.0), *GPT-2* (MIT/Modified MIT), *ByT5* (Apache-2.0), *Mistral-7B* (Apache-2.0), *Llama 3/3.1* (Meta's Llama Community License), *Falcon-7B* (Apache-2.0), *BLOOM* (BigScience OpenRAIL-M with use-based restrictions), and *Yi-1.5* (Apache-2.0). We do not attempt to relicense any third-party artifacts; we redistribute nothing beyond what is explicitly permitted; and any derivatives (e.g., analysis outputs or small synthetic examples) are shared only in ways compatible with the upstream terms. We encourage downstream users to review the original license pages before reusing any asset and to note that "responsible-AI" licenses (e.g., OpenRAIL-M) and community licenses (e.g., Llama) include behavioral or compatibility restrictions that may differ from permissive open-source terms.