Spelling-out is not Straightforward: LLMs' Capability of Tokenization from Token to Characters

Tatsuya Hiraoka[1,2,3] **Kentaro Inui**[1,2]
[1] Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)
[2] RIKEN
[3] NARA Institute of Science and Technology
{tatsuya.hiraoka, kentaro.inui}@mbzuai.ac.ae

Abstract

Large language models (LLMs) can spell out tokens character by character with high accuracy, yet they struggle with more complex character-level tasks, such as identifying compositional subcomponents within tokens. In this work, we investigate how LLMs internally represent and utilize character-level information during the spelling-out process. Our analysis reveals that, although spelling out is a simple task for humans, it is not handled in a straightforward manner by LLMs. Specifically, we show that the embedding layer does not fully encode character-level information, particularly beyond the first character. As a result, LLMs rely on intermediate and higher Transformer layers to reconstruct character-level knowledge, where we observe a distinct "breakthrough" in their spelling behavior. We validate this mechanism through three complementary analyses: probing classifiers, identification of knowledge neurons, and inspection of attention weights.

1 Introduction

While large language models (LLMs) have grown remarkably in recent years, several studies report that they still struggle with fine-grained character-level manipulations, such as inserting, deleting, or extracting individual characters within tokens (Edman et al., 2024; Wang et al., 2024; Chai et al., 2024; Shin and Kaneko, 2024). Although most LLMs operate over subword tokens, true mastery of subtoken information is essential for a range of applications, such as morphological inflection (Marco and Fraser, 2024), letter counting (Fu et al., 2024), typoglycemia (Wang et al., 2025), and handling typos (Tsuji et al., 2025). To improve their reliability in such scenarios, we must understand how LLMs internally represent and process characters.

A paradox emerges from prior work: LLMs can accurately spell out entire tokens as sequences of characters (Edman et al., 2024; Xiong et al., 2025), while they often fail at simple tasks such as identifying a single character at a fixed position (Itzhak and Levy, 2022; Kaushal and Mahowald, 2022; Hiraoka and Okazaki, 2024; Chai et al., 2024). For instance of a token "language," well-pretrained models readily generate "l a n g u a g e" on demand but cannot reliably extract "u" at position five. This discrepancy suggests that, despite lacking explicit access to compositional character knowledge, LLMs have some mechanism for character-by-character spelling.

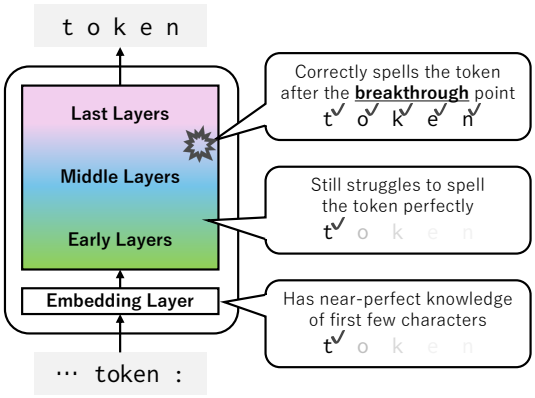

Figure 1: Summary of our findings. LLMs spell out tokens by relying directly on embedding-level information for the first character, but gradually shift to using distributed, higher-layer representations for later characters. We found a "breakthrough" layer where character knowledge becomes detectable.

In this paper, we investigate how and where LLMs capture and deploy character-level knowledge during spelling-out. We begin by constructing a token-characters dataset from the vocabularies of four representative LLMs (§3) and confirming their ability to spell out tokens with few-shot prompts (§4). Probing the embedding layer reveals that it does not encode subtoken characters directly (§5), but downstream Transformer layers gradually recover this information. We identify a clear "breakthrough" layer at which character identities become

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 13340–13353

reliably detectable by our probing classifier (§6), and observe that the first few characters are handled differently from subsequent ones. Finally, through neuron-level analyses (§7) and attention-weight inspections (§7.4), we trace how character knowledge is stored and routed, demonstrating that its locus aligns precisely with the breakthrough point.

Taken together, our study sheds new light on the internal character-level machinery of LLMs. Our main contributions are as follows:

- We demonstrate that token embeddings do not fully encode character-level information, and that subsequent Transformer layers play a major role in reconstructing this information during the spelling-out process.
- We pinpoint the exact layer where compositional character information "breaks through," validated via probing classifiers.
- We investigate character knowledge to specific neurons and attention patterns, showing alignment with the breakthrough layer.
- We provide a diagnostic framework (dataset, probes, and analyses) for future research into subtoken and character-level modeling.

Codes for our experiments are available at: https://github.com/tatHi/token2char.

2 Related Work

Our research is in line with work focusing on the LLMs' capability of character-level manipulation. Kaushal and Mahowald (2022) investigated the LLMs' capability to identify the appearance of particular characters in a given token using probing classifiers. Itzhak and Levy (2022) directly analyzed the information stored in token embeddings to spell out tokens using additional character-level language models. More recently, Edman et al. (2024) introduced a dataset to evaluate this capability from some viewpoints of character-level manipulation, including spelling-out of words or tokens. Similarly, Wang et al. (2024) provided a dataset of complicated character-level manipulation tasks. Both works conclude that LLMs have limited capability for complicated character processing. Furthermore, these works reported that LLMs can spell out words in the original order, as reported in Xiong et al. (2025), while they do not have sufficient knowledge of their single characters inside words (Shin and Kaneko, 2024; Hiraoka and Okazaki, 2024; Chai et al., 2024) unless fine-tuning for the models to learn the token internal information directly (Xu et al., 2024).

This line of literature motivates us to investigate the LLMs' internal workings of spelling-out behavior, despite the lack of character-level knowledge. Recently, we can see a trend of understanding LLMs' capability of recognizing character-level information, such as the ability of counting characters (Fu et al., 2024). Moreover, beyond the word-to-character spelling out, Wu et al. (2025) investigates their ability to recognize radicals inside Chinese characters.

Our work is also related to the findings of LLMs' "detokenization" ability in their later layers (Kaplan et al., 2025; Kamoda et al., 2025), which is an ability to internally merge tokens into words or phrases. In contrast, we focus on the inverse problem: how LLMs internally "tokenize" tokens into characters.

	# Vocab	# Dataset	%
LLaMA3-8B	128,256	19,724	15.38
Gemma-7B	256,000	47,833	18.68
Qwen2.5-7B	152,064	18,973	12.48
Amber-6.7B	32,000	6,130	19.16

Table 1: The number of tokens in the vocabulary of each LLM (**# Vocab**) and in our dataset (**# Dataset**, §3.2). % shows the ratio of in-dataset tokens in the vocabulary.

3 Experimental Setup

3.1 Target Models

We investigate four medium-sized LLMs (≈ 7B parameters): LLaMA3-8B (Dubey et al., 2024), Gemma-7B (Team et al., 2024), Qwen2.5-7B (Yang et al., 2024), and Amber (Liu et al.). We selected them because our preliminary trials showed that smaller models (e.g., 3B) fail to spell out tokens with sufficient accuracy. All models have the Transformer-based architecture (Vaswani et al., 2017). Table 1 shows each model's vocabulary size, which ranges from 32K to 256K tokens.

3.2 Evaluation Dataset

Our evaluation focuses on spelling out single tokens into their constituent characters. In other words, we exclude multi-token words to ensure a well-controlled experimental setting. For example, in the case of "token/s", the model could easily reveal the final "s" without requiring character-level understanding.

We construct the dataset from each model's vocabulary by selecting all single tokens that: 1) contain only lowercase alphabets (a-z), 2) begin with

Few-shot Example	`hello : h e l l o,`
	`world : w o r l d,`
	`orange : o r a n g e,`

Single Token Input	`libert :`
Expected Output	`l i b e r t`

Table 2: An input example for the three-shot setting.

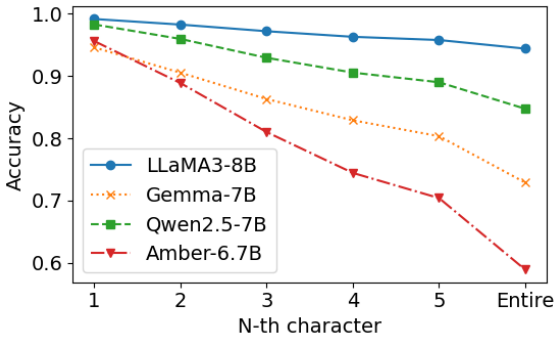

Figure 2: Experimental results with the few-shot setting. $N = 1...5$ means the accuracy of the N-th character prediction. "Entire" shows the accuracy of the prediction for all characters of tokens.

the special prefix indicating the head of words (e.g., "_" in "_token"), and 3) are at least five characters. This yields a set comprising roughly 15 % of each model's full vocabulary (Table 1). Note that our dataset allows tokens that are subword fragments rather than complete words (e.g., "somew," "immedi"). Figure 9 and 10 in the appendix show the distribution of frequency of token length and alphabet, respectively.

In the spelling-out task, given the target tokens in our dataset, LLMs are expected to output a sequence of compositional characters of each token. We represent the spelling-out with a whitespace separation in our experiments. For example, LLMs with an input "token" are expected to generate a sequence " t o k e n". In other words, the targeted LLMs need to generate character tokens with the head-prefix such as "_t, _o, _k, _e, _n" [1].

4 LLMs can Spell-out Tokens

We first assess each LLM's ability to spell out single tokens given few-shot examples, following prior work on word-level spelling-out (Edman et al., 2024; Xiong et al., 2025). Table 2 shows a prompt of a three-shot example for this experiment. We measure performance over the full evaluation dataset created in §3.2 for each LLM with three-shot examples randomly selected from the dataset, excluding the tested sample. We count a prediction as correct only if the model's output exactly matches the ground-truth character sequence with no missing or extra characters.

The data points named "Entire" in Figure 2 report the overall token-level spelling-out accuracy of each LLM. Although direct comparison across models is impossible because each uses its own vocabulary subset, we observe substantial differences: LLaMA3-8B achieves 94.41% accuracy, whereas Amber-6.7B reaches only 58.86%. This variation shows the impact of model architecture and pretraining data on character-level capabilities.

In Figure 2, we also report accuracy by character position within each token. All models achieve over 94% accuracy on the first character, but accuracy steadily declines for later positions. This result indicates that correctly generating characters further along in the token becomes increasingly difficult. Notably, however, every model maintains over 70% accuracy through the fifth character, demonstrating LLMs' robust mid-token spelling capability.

5 Embeddings do not Know All Compositional Characters

While Section 4 demonstrated that LLMs can spell out single tokens, two key questions remain: where is the character-level knowledge stored in the model, and how is that knowledge utilized during spelling? A natural hypothesis is that token embeddings themselves encode this information (Itzhak and Levy, 2022), since the spelling of a token (i.e., its sequence of characters) is inherently context-independent and embeddings are derived directly from token identities. This section investigates whether the token embedding stores the knowledge of spelling tokens.

5.1 MLP Probing with Token Embedding

To investigate where LLMs store character-level information, we train probing classifiers that predict the N-th character of a token t from its embedding $\mathbf{v}_t \in \mathbb{R}^d$. We train a separate MLP classifier[2] for each character position N ($1 \leq N \leq 5$), using an identical architecture across positions.

The MLP classifier maps the d-dimensional token embedding $\mathbf{v}_t$ to a 26-dimensional logits vector, corresponding to the 26 lowercase English char-

[1] Using other separators such as "t/o/k/e/n" also yields qualitatively similar results in our experiments.

[2] We selected a non-linear probing because a linear classifier was incapable of extracting character-level information from embeddings, following Kaushal and Mahowald (2022).

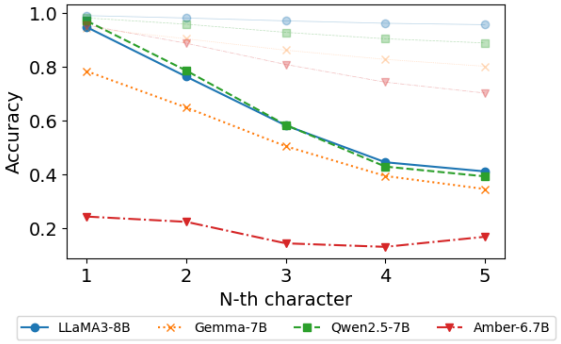

Figure 3: The performance of probing classifiers for each N (dark-colored lines). The light-colored lines show the few-shot performance copied from Figure 2.

acters. The probability that the N-th character of t is c is computed as:

$$p(t_N = c|t) = \mathrm{softmax}(f(\mathbf{v}_t; \theta_N))_c, \quad (1)$$

where $f : \mathbb{R}^d \to \mathbb{R}^{26}$ is the MLP classifier with three linear layers with the tanh activation, and θ_N is the set of trainable parameters for predicting the N-th character. The $\mathrm{softmax}(\cdot)_c$ operation extracts the probability assigned to character c.

We train each probing classifier on randomly sampled 90% of the dataset (§3.2) and evaluate it on the remaining 10%. This process is repeated ten times as k-fold cross-validation. If the classifier can accurately predict characters at a given position, we interpret this as evidence that the token embedding encodes character-level information at that position. Training is performed using cross-entropy loss and the Adam optimizer (Kingma, 2014) with 300 epochs of training, for which we observed convergence of the training loss in all experimental setups.

5.2 Experimental Result

Figure 3 shows the performance of the probing classifiers across character positions. For LLMs with larger vocabulary sizes (i.e., LLaMA3-8B, Gemma-7B, and Qwen2.5-7B), the classifier achieves over 80% accuracy in predicting the first character. However, performance declines consistently as the character position increases. In parallel, the gap between probing accuracy (dark-colored lines) and few-shot accuracy (light-colored lines) widens at later positions. These results suggest that LLMs rely on token embeddings to retrieve the first character, but access character-level information from upper layers when spelling out later characters.

Amber-6.7B, the model with the smallest vocabulary, shows a distinct trend. Its probing accuracy is significantly lower even for the first character. This result indicates that its token embeddings carry little or no character-level information[3].

In summary, these results indicate that token embeddings in LLMs do not encode full character composition. While the first character is sometimes recoverable, character-level knowledge beyond that is primarily stored in the LLM's upper layers.

6 Which Layer Knows Spelling-out?

This section extends the probing analysis to internal Transformer layers to investigate where character-level knowledge emerges within the model.

6.1 MLP Probing with Layer Output

To analyze how character knowledge develops across the model, we apply the same probing classifier from §5, but instead of the token embedding $\mathbf{v}_t$, we use hidden states from individual Transformer layers. Specifically, for predicting the N-th character, we extract the hidden state $\mathbf{h}^l_{N-1}$ from the l-th layer, corresponding to the token immediately before the character being predicted. For example, to predict the third character ($N = 3$) of the token "`token`," we extract the hidden state for the final token of the input sequence "`[few-shot examples] token : t o`", expecting the model to predict "k".

All other aspects of the probing setup, including model architecture, training procedure, and evaluation, remain the same as described in §5.

6.2 Experimental Result

Figure 4 presents the character prediction accuracy of probing classifiers across Transformer layers for all four LLMs. A consistent two-peak trend emerges: classifiers using early-layer hidden states can predict characters to some extent but not perfectly, whereas those using higher-layer representations can almost perfectly predict characters at all positions, following a performance drop in the intermediate layers. The accuracy at the final layer closely matches that in Figure 2, supporting the validity of our evaluation setup.

Interestingly, LLaMA3-8B and Gemma-7B exhibit a notable dip in accuracy at the first Transformer layer (i.e., the second data point from the left). This suggests that character-level information

[3]Given that probing performance on intermediate layers of Amber is reasonable (§6), the lower embedding-level accuracy is unlikely to be caused by the smaller dataset size (Table 1).

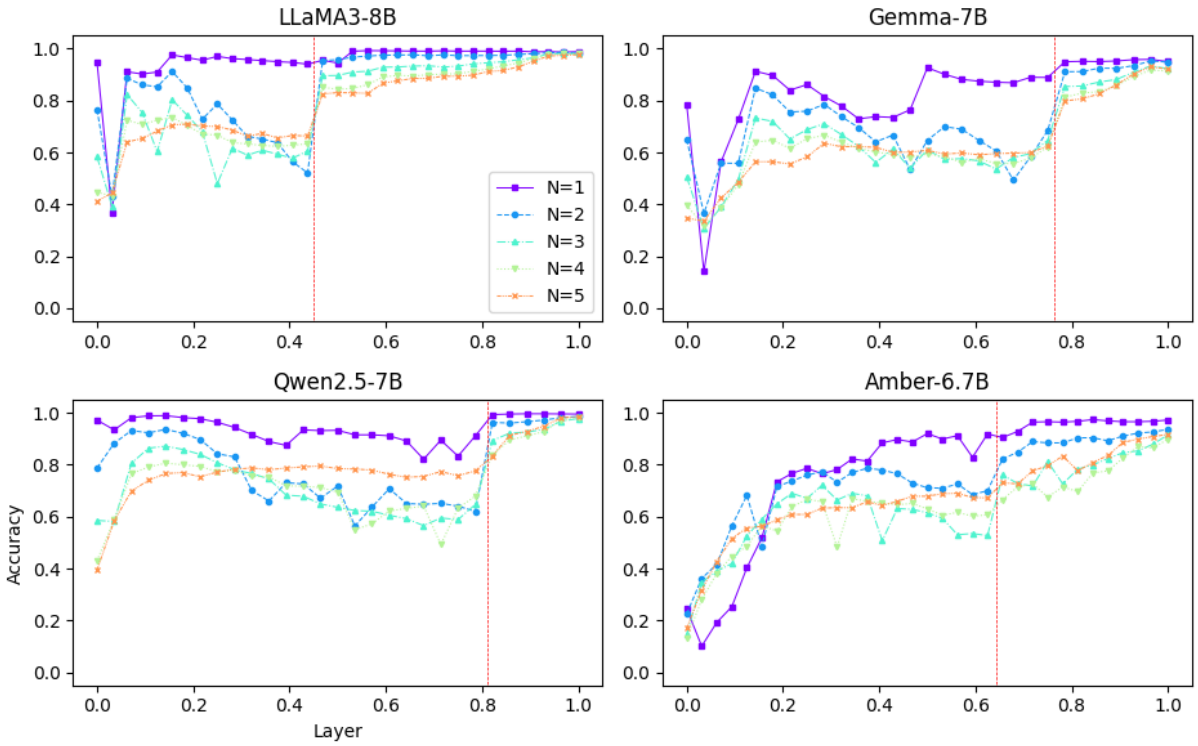

Figure 4: Accuracy of N-th character prediction by probing classifiers at each Transformer layer. The X-axis represents relative layer depth (0.0 = embedding layer, 1.0 = final layer). Red vertical lines indicate the breakthrough layers, which are calculated based on the average performance improvement between adjacent layers for $2 \leq N$.

is not immediately accessible after the embedding layer and may even be disrupted at this stage.

For models other than Amber-6.7B, character prediction reaches higher accuracy in the early layers, but then temporarily declines before recovering in later layers. This suggests that LLMs do not simply pass character-level information through the network, but they engage in intermediate processing that reorganizes it as part of building the knowledge for solving the spelling-out task.

All models show a distinct "breakthrough" point, highlighted as red vertical lines, where the accuracy of the character prediction sharply increases in later layers. For example, LLaMA3-8B exhibits a jump in accuracy around layer depth 0.45. Amber-6.7B also shows an upward trend in the later layers, though the improvement is more modest compared to the other models. This trend can be seen much more clearly when we use an explicit separator "/" for the spelling-out (Figure 12).

These results suggest that LLMs begin to consolidate character-level spelling knowledge in the later stages of the network. In other words, the model first interprets the spelling-out task in its intermediate layers and then may act more like *a character-level language model* in its final layers. This interpretation aligns with prior findings showing that LLM hidden states gradually shift toward representations resembling the next predicted token (Voita et al., 2024; Chang and Bergen, 2025). Furthermore, this result also aligns with research reporting the significant workings in LLMs' later layers on specific tasks (Merullo et al., 2024; Lad et al.; Nikankin et al., 2025).

Finally, we observe that probing accuracy continues to rise even after the breakthrough point, particularly in the final two layers. This reinforces the view that character-level information is not statically stored in the embedding layer but is dynamically constructed and refined throughout the model, especially toward the end of the forward pass.

7 Knowledge Neuron for Spelling-out

The breakthrough point observed in §6 implies that large language models (LLMs) possess the ability to spell out tokens in the layers preceding this point. Recent work on interpretability has shown that individual neurons in Transformer models may encode factual knowledge or skills, as in knowledge neurons (Dai et al., 2022) and skill neurons (Wang et al., 2022). Inspired by this line of research, we

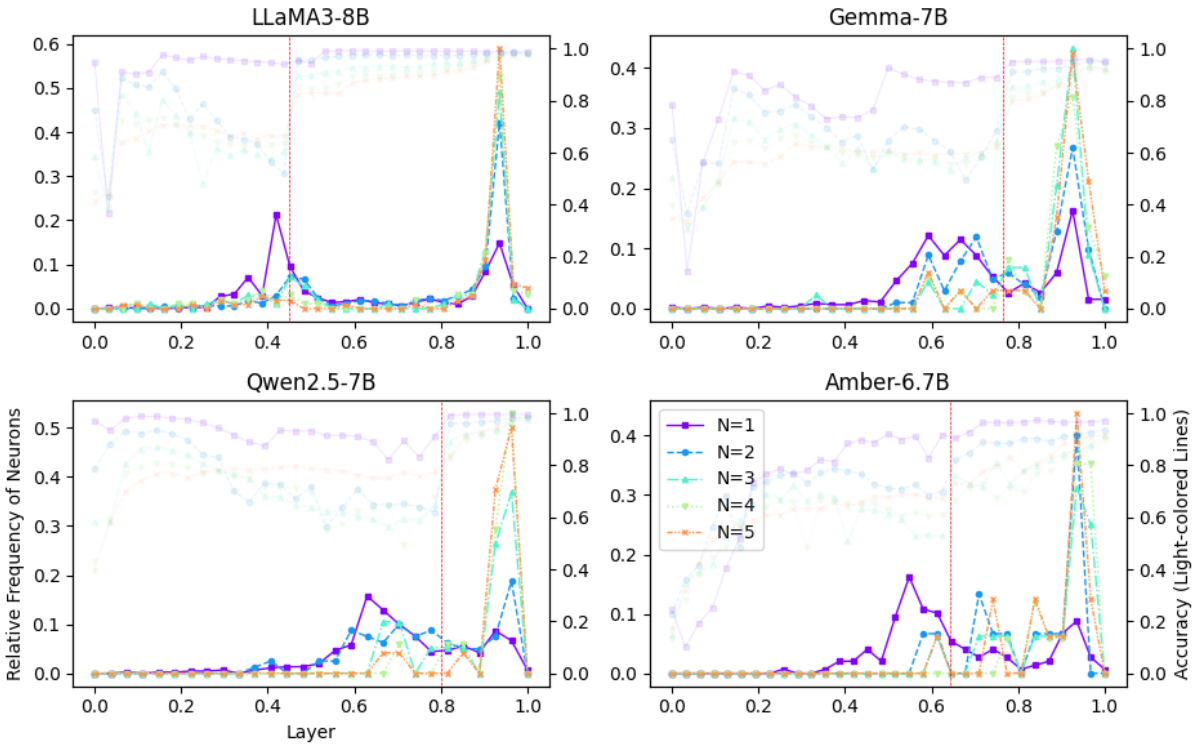

Figure 5: Distribution of knowledge neurons for each layer (dark-colored lines). The accuracy of probing classifiers (light-colored lines) are copied from Figure 4.

investigate where the knowledge for spelling out tokens resides within the neurons of each LLM.

7.1 Knowledge Neurons

In line with previous studies on neuron-level analysis of Transformer architectures (Geva et al., 2021), we define knowledge neurons as the outputs of activation functions in the feed-forward network (FFN) sublayers of Transformer blocks. To quantify each neuron's contribution to a specific output character, we compute an attribution score.

For instance, given an input x ="`token : to`", we aim to measure how much a neuron w contributes to predicting the next correct character c ="k". We define the attribution score $\mathrm{Attr}(w|c,x)$ for the neuron w using the following integrated gradient-based approximation:

$$\mathrm{Attr}(w|c,x) = \frac{\bar{w}}{m}\sum_{k=1}^{m}\frac{\partial P_{c,x}(\frac{k}{m}\bar{w})}{\partial w}, \qquad (2)$$

where $\bar{w}$ is the activation value of neuron w when the LLM processes the input x. Here, $P_{c,x}(\bar{w})$ denotes the model's predicted probability of character c given input x when w is $\bar{w}$, and $m = 20$ is the number of interpolation steps used in the Riemann sum approximation, as in Dai et al. (2022).

Using this attribution score, we identify knowledge neurons responsible for generating the N-th character in the spelling-out task. In other words, we examine how neuron activations vary depending on the position of the predicted character.

To identify knowledge neurons for each character position N, we proceed as follows. For each of the 1,000 sampled tokens from the dataset, we compute $\mathrm{Attr}(w|c,x)$ for all neurons and rank them. Then, for each token, we select the top 1% of neurons with the highest attribution scores at position N. Finally, we define a neuron as a knowledge neuron for position N if it appears in the top 1% set of at least 75% of the 1,000 tokens.

7.2 Distribution of Neurons for N

Figure 5 shows the distribution of knowledge neurons across layers for predicting the N-th character. The distributions commonly exhibit two peaks: one at intermediate layers and another near the final layers. However, the peak for the first character tends to occur in intermediate layers, while the peaks for later characters (e.g., $N = 2$ and beyond) appear more prominently near the final layers. This trend is consistent across all four models, suggesting a general phenomenon shared by various LLMs.

We also observe that the location of the first peak

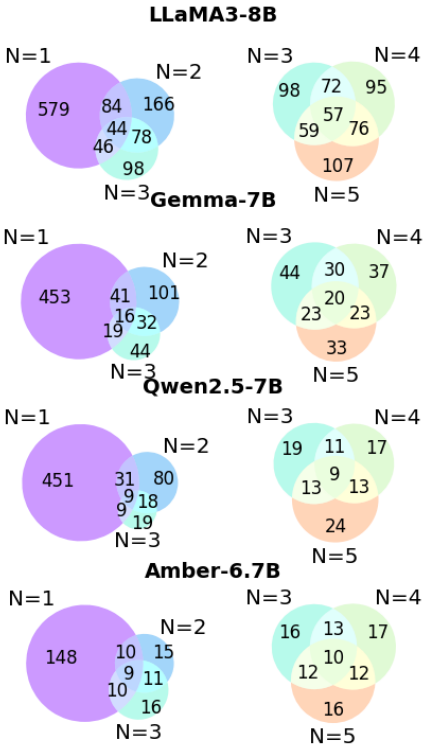

Figure 6: Venn diagrams showing the number of overlapping knowledge neurons across different character positions N for four LLMs.

is loosely aligned with the breakthrough point identified earlier. For example, in LLaMA3-8B, where the breakthrough occurs around mid-depth (layer ~ 0.5), the first peak of knowledge neurons also appears around the same depth. Conversely, in Gemma-7B, where the breakthrough appears later, the first peak of neurons shifts to a correspondingly later layer. These findings suggest that knowledge neurons in the mid-depth layers may play a foundational role in the spelling-out process.

7.3 Intersection of Neurons

Figure 6 illustrates the overlap of knowledge neurons across different character positions N. Across all models, the largest number of neurons is uniquely identified for the first character prediction, in contrast to those for the second and third characters, which share more neurons.

Moreover, the total number of knowledge neurons tends to decrease from $N = 1$ to $N = 3$. This suggests that LLMs use a broader and more redundant set of neurons when initiating the spelling-out process, but rely on fewer and more specialized neurons as the character position progresses.

For later positions ($N = 3, 4, 5$), both the number of knowledge neurons and the extent of their

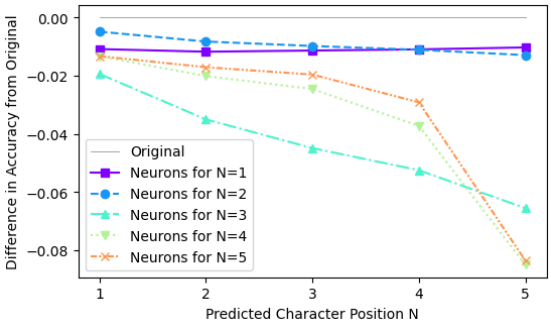

Figure 7: Difference in accuracy with ablating 100 neurons for each character position N (LLaMA3-8B).

overlap remain relatively small and stable across all models. This suggests a shift in internal strategy: while LLMs rely on a broad and shared set of neurons to initiate spelling-out at $N = 1$, they gradually move toward using more position-specific and case-dependent neurons for later characters. In other words, the model appears to generalize the beginning of spelling with common mechanisms, but adapts to the unique structure of each token as the character position advances.

These consistent trends across the four LLMs support a general conclusion: LLMs tend to rely on a broad and shared set of neurons for predicting the first character, and to some extent the second, using a general mechanism to initiate the spelling-out process. In contrast, for characters beyond the third position, they employ more case-specific neurons, indicating a shift toward more specialized processing tailored to individual token structures.

7.4 Neuron Ablation

To gain deeper insight into the roles of individual neurons, we ablated the top 100 most influential neurons for each character position N and measured the resulting performance degradation relative to the original accuracy shown in Figure 2. We used LLaMA3-8B for this experiment, as it provides a sufficient number of neurons for meaningful ablation analysis (Figure 6).

The experimental results in Figure 7 show that ablating neurons for $N = 1, 2$ leads to only modest accuracy loss, whereas ablating neurons for $N \geq 3$ causes a more substantial drop in performance. This suggests that later-position neurons play a more critical role in the spelling-out task, compensating for the functions of earlier ones by leveraging contextual information. These results support our hypothesis in Section 7.3: early-position neu-

rons primarily extract features from initial character embeddings, whereas later-position neurons are responsible for context-based character prediction.

8 Attention Weight for Spelling-out

Given the nature of the spelling-out task, LLMs must attend to the target token while generating its characters. The ability to focus attention on the correct token is crucial for accurate spelling-out, and we hypothesize that this ability corresponds to the breakthrough identified in §6. This section further investigates the relationship between spelling-out capability and attention weights to the target token.

Given a sequence consisting of the target token and its spelling-out from our dataset, we compute the average attention weight from all related elements to the target token. For example, when the input is `[few-shot examples] token : t o k e n`, where the target token is "`token`" following a few-shot example, we calculate the average attention weight to "`token`" from each of the elements: `token`, `:`, `t`, `o`, `k`, `e`, and `n`. We average these attention weights over 1,000 randomly selected samples and across all attention heads in each layer[4]. To isolate attention to the target token, we remove attention weights to "`<BOS>`" and renormalize the distributions prior to averaging, inspired by Kobayashi et al. (2020).

Figure 8 presents the average attention scores across the 1,000 examples. Interestingly, the layer with the highest attention to the target token (red bars) coincides with the breakthrough point (red lines) in three out of four models. This finding suggests that the performance improvement observed in intermediate layers is due to the model's increasing ability to correctly attend to the target token.

The distinct result observed in Amber-6.7B suggests that performance breakthroughs do not necessarily align with the behavior of attention weights. Given the overall lower attention weights assigned to the target token compared to other models, we consider that this model needs to attend to broader contextual information rather than focusing on the single target token, possibly due to a lack of character-level knowledge in its embeddings.

9 Conclusion

This paper investigated the paradox that, although large language models (LLMs) struggle to recognize individual characters within words, they can accurately spell out words character by character.

Our probing analyses revealed that the token embedding layer does not encode complete character-level information, especially beyond the first character. Instead, character-level features are dynamically reconstructed in the intermediate and later layers, where we observe a distinct "breakthrough" point. Around this stage, models begin to reliably access and utilize character-level knowledge.

We further examined the spelling-out behavior through the lens of knowledge neurons and attention patterns, both of which align with the breakthrough layers. Crucially, this behavior suggests that spelling out is a learned task (i.e., dependent on identifying the target token and retrieving character-level information), rather than a simple extraction of character-level information stored in the embedding layers.

As a result, our findings imply that the apparent success of LLMs in spelling out does not extend to more complex or unfamiliar tasks such as reverse spelling, letter insertion, or character-level reasoning in novel contexts. Our findings indicate that current LLMs are not inherently character-aware; rather, they rely on task-specific heuristics acquired during training or prompting. This suggests that models must explicitly learn how to apply character knowledge to more complex manipulations.

Limitations

While we believe that our experiments were conducted under well-controlled conditions and provide sufficient support for our hypotheses, we acknowledge the following limitations:

- To ensure well-controlled experimental conditions, we restricted the target vocabulary to single-token words composed of lowercased alphabetic characters. Different trends may emerge when using other languages. However, considering prior work showing that LLMs can predict the initial radicals of Chinese characters (Wu et al., 2025), we expect similar tendencies to hold across languages.

- This study investigates LLM behavior using probing classifiers, knowledge neurons, and attention heads. It is important to note, however, that these methods do not fully capture or explain the underlying mechanisms of model knowledge and capabilities (Jain and Wallace,

[4]These tokens are selected from the dataset as those that each LLM can correctly spell out in the few-shot setting (§4).

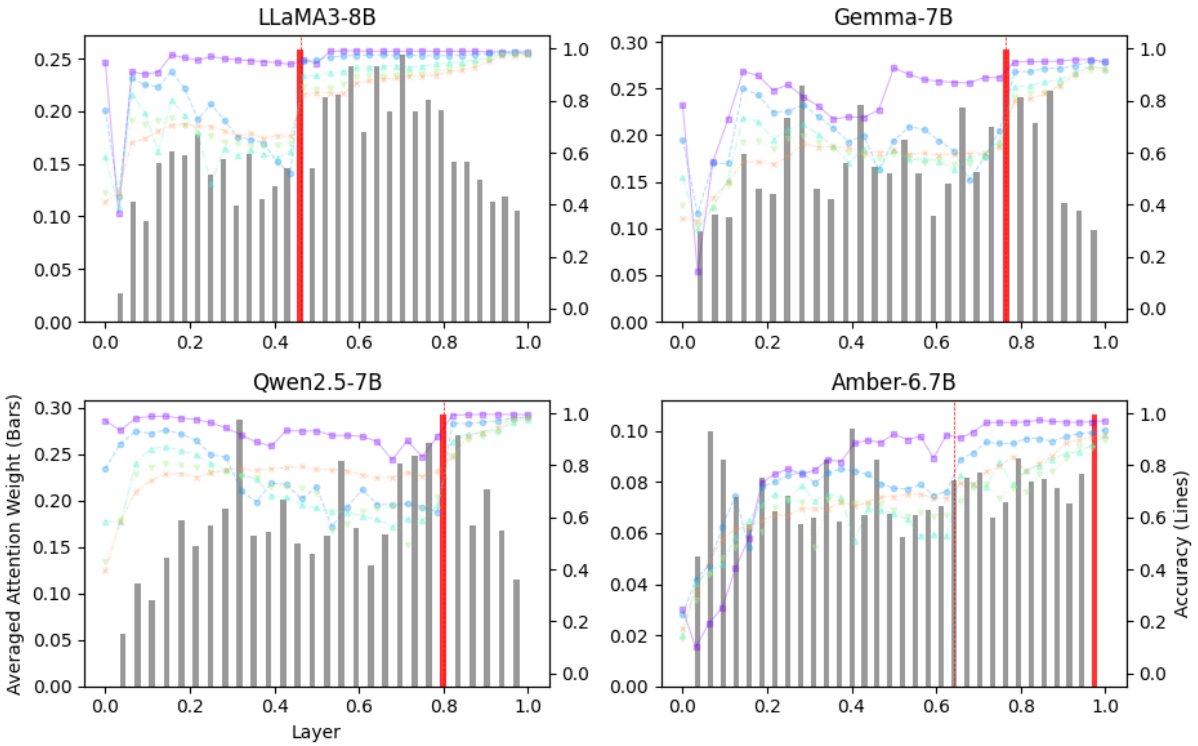

Figure 8: Averaged attention weights to the target token across layers (bar graph). The red bar indicates the layer with the highest attention weight. The line graph shows the performance of predicting the N-th character using the probing classifier, replicated from Figure 4.

2019; Belinkov, 2022; Kumar et al., 2022; Niu et al.).

- Our experiments focus on spelling-out behavior using whitespace as a separator. Although similar trends were observed when using an alternative separator ("/", see Figure 12), the results may vary depending on input formatting.

- Our findings do not necessarily generalize to all LLMs. In fact, Amber-6.7B exhibited divergent behavior across several experiments, deviating from the patterns observed in other models. Nevertheless, we argue that presenting such counterexamples is essential for a deeper understanding of model behavior. In this work, we attribute Amber's deviation to its apparent lack of character-level information in the embedding layer, a distinctive property not shared by the other models. Given that the remaining three models consistently followed the same trends, we consider Amber-6.7B to be a special case rather than a representative counterpoint.

Acknowledgement

A part of this work was supported by JSPS KAKENHI Grant Number JP24K20852.

References

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. Computational Linguistics, 48(1):207–219.

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

Tyler A Chang and Benjamin K Bergen. 2025. Bigram subnetworks: Mapping to next tokens in transformer language models. arXiv preprint arXiv:2504.15471.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8493–8502.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela

Fan, et al. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783.

Lukas Edman, Helmut Schmid, and Alexander Fraser. 2024. CUTE: Measuring LLMs' understanding of their tokens. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 3017–3026, Miami, Florida, USA. Association for Computational Linguistics.

Tairan Fu, Raquel Ferrando, Javier Conde, Carlos Arriaga, and Pedro Reviriego. 2024. Why do large language models (llms) struggle to count letters? arXiv preprint arXiv:2412.18626.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 5484–5495.

Tatsuya Hiraoka and Naoaki Okazaki. 2024. Knowledge of pretrained language models on surface information of tokens. arXiv preprint arXiv:2402.09808.

Itay Itzhak and Omer Levy. 2022. Models in a spelling bee: Language models implicitly learn the character composition of tokens. In 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. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. 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 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Go Kamoda, Benjamin Heinzerling, Tatsuro Inaba, Keito Kudo, Keisuke Sakaguchi, and Kentaro Inui. 2025. Weight-based analysis of detokenization in language models: Understanding the first stage of inference without inference. In Findings of the Association for Computational Linguistics: NAACL 2025, pages 6324–6343, Albuquerque, New Mexico. Association for Computational Linguistics.

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.

Ayush Kaushal and Kyle Mahowald. 2022. What do tokens know about their characters and how do they know it? In 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. Association for Computational Linguistics.

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 7057–7075, Online. Association for Computational Linguistics.

Abhinav Kumar, Chenhao Tan, and Amit Sharma. 2022. Probing classifiers are unreliable for concept removal and detection. Advances in Neural Information Processing Systems, 35:17994–18008.

Vedang Lad, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference? In ICML 2024 Workshop on Mechanistic Interpretability.

Zhengzhong Liu, Aurick Qiao, Willie Neiswanger, Hongyi Wang, Bowen Tan, Tianhua Tao, Junbo Li, Yuqi Wang, Suqi Sun, Omkar Pangarkar, et al. Llm360: Towards fully transparent open-source llms. In First Conference on Language Modeling.

Marion Di Marco and Alexander Fraser. 2024. Subword segmentation in LLMs: Looking at inflection and consistency. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 12050–12060, Miami, Florida, USA. Association for Computational Linguistics.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Language models implement simple Word2Vec-style vector arithmetic. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 5030–5047, Mexico City, Mexico. Association for Computational Linguistics.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2025. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In The Thirteenth International Conference on Learning Representations.

Jingcheng Niu, Andrew Liu, Zining Zhu, and Gerald Penn. What does the knowledge neuron thesis have to do with knowledge? In The Twelfth International Conference on Learning Representations.

Andrew Shin and Kunitake Kaneko. 2024. Large language models lack understanding of character composition of words. In ICML 2024 Workshop on LLMs and Cognition.

Gemma Team, 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.

Kohei Tsuji, Tatsuya Hiraoka, Yuchang Cheng, Eiji Aramaki, and Tomoya Iwakura. 2025. Investigating neurons and heads in transformer-based llms for typographical errors. arXiv preprint arXiv:2502.19669.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc.

Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. 2024. Neurons in large language models: Dead, n-gram, positional. In Findings of the Association for Computational Linguistics: ACL 2024, pages 1288–1301, Bangkok, Thailand. Association for Computational Linguistics.

Chenxi Wang, Tianle Gu, Zhongyu Wei, Lang Gao, Zirui Song, and Xiuying Chen. 2025. Word form matters: Llms' semantic reconstruction under typoglycemia. arXiv preprint arXiv:2503.01714.

Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li. 2022. Finding skill neurons in pre-trained transformer-based language models. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 11132–11152.

Xilong Wang, Hao Fu, Jindong Wang, and Neil Zhenqiang Gong. 2024. Stringllm: Understanding the string processing capability of large language models. arXiv preprint arXiv:2410.01208.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Xiaofeng Wu, Karl Stratos, and Wei Xu. 2025. The impact of visual information in Chinese characters: Evaluating large models' ability to recognize and utilize radicals. In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 331–350, Albuquerque, New Mexico. Association for Computational Linguistics.

Zhen Xiong, Yujun Cai, Bryan Hooi, Nanyun Peng, Zhecheng Li, and Yiwei Wang. 2025. Enhancing llm character-level manipulation via divide and conquer. arXiv preprint arXiv:2502.08180.

Zhu Xu, Zhiqiang Zhao, Zihan Zhang, Yuchi Liu, Quanwei Shen, Fei Liu, Yu Kuang, Jian He, and Conglin Liu. 2024. Enhancing character-level understanding in llms through token internal structure learning. arXiv preprint arXiv:2411.17679.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. arXiv preprint arXiv:2412.15115.

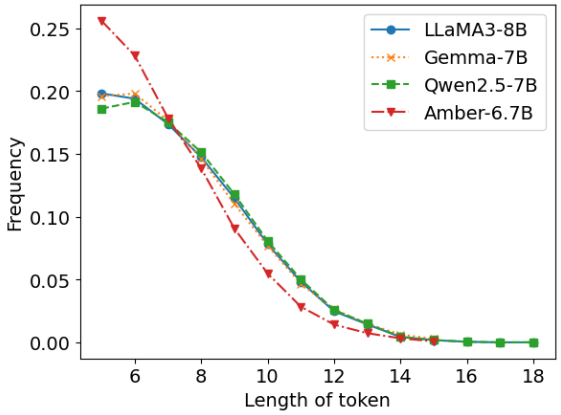

Figure 9: Token length frequency distribution in the dataset.

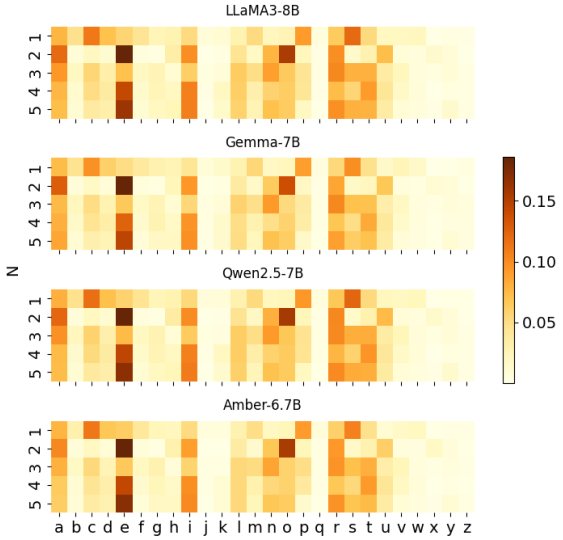

Figure 10: Alphabet frequency distribution at each character position in the dataset.

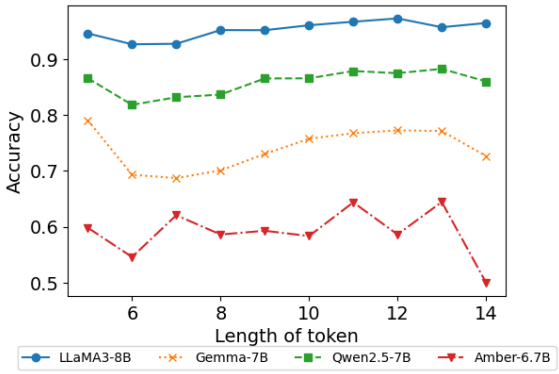

Figure 11: Few-shot spelling-out accuracy by input token length.

A Experimental Environment

All experiments in this paper were conducted using the HuggingFace Transformers library (Wolf et al., 2020). Unless otherwise specified, we used default hyperparameter settings. Most experiments were performed on NVIDIA A40 GPUs, with the exception of those involving Gemma-7B, which were conducted on NVIDIA H100 GPUs.

The most computationally intensive part of our work was the identification of neurons described in §7, which took over 24 hours to process 1,000 samples. All other experiments were completed within 24 hours using a single GPU.

B Additional Dataset Statistics

Figure 9 shows the distribution of token lengths in our dataset. As illustrated, the three LLMs with larger vocabularies exhibit similar distributions. Figure 10 presents the distribution of characters (alphabets) at each position in the tokens, showing highly consistent patterns across models.

C Few-shot Results by Token Length

Figure 11 plots the few-shot spelling-out accuracy from §4 as a function of token length (5–14 characters). Contrary to the intuition that longer tokens would be harder to spell out correctly, the relatively flat trend suggests that token length has a limited impact on exact token-level accuracy.

D Layer Probing with "/" Separation

Figure 12 presents the results of probing classifiers when the separator used for spelling out is changed from whitespace to a forward slash ("/"). The experimental setup remains the same as in §6, except for this separator change. This modified separator was used consistently across few-shot examples, target inputs, and expected outputs (see Figure 2). For instance, the input "`hello :/h/e/l/l/o/`" was used in one of the examples.

To ensure consistency of representation across character positions, we added a slash before and after the spelling so that the model does not need to use tokens with the special prefix indicating the word head. In contrast, the main experiments used inputs such as "`_h_e_l_l_o`", where the underscore ("_") denotes a prefix.

As shown in Figure 12, we observe a similar trend to that in Figure 4, including the occurrence of a "breakthrough" at approximately the same layer. This consistency suggests that our findings regarding breakthrough behavior may generalize across different input formats for spelling tasks.

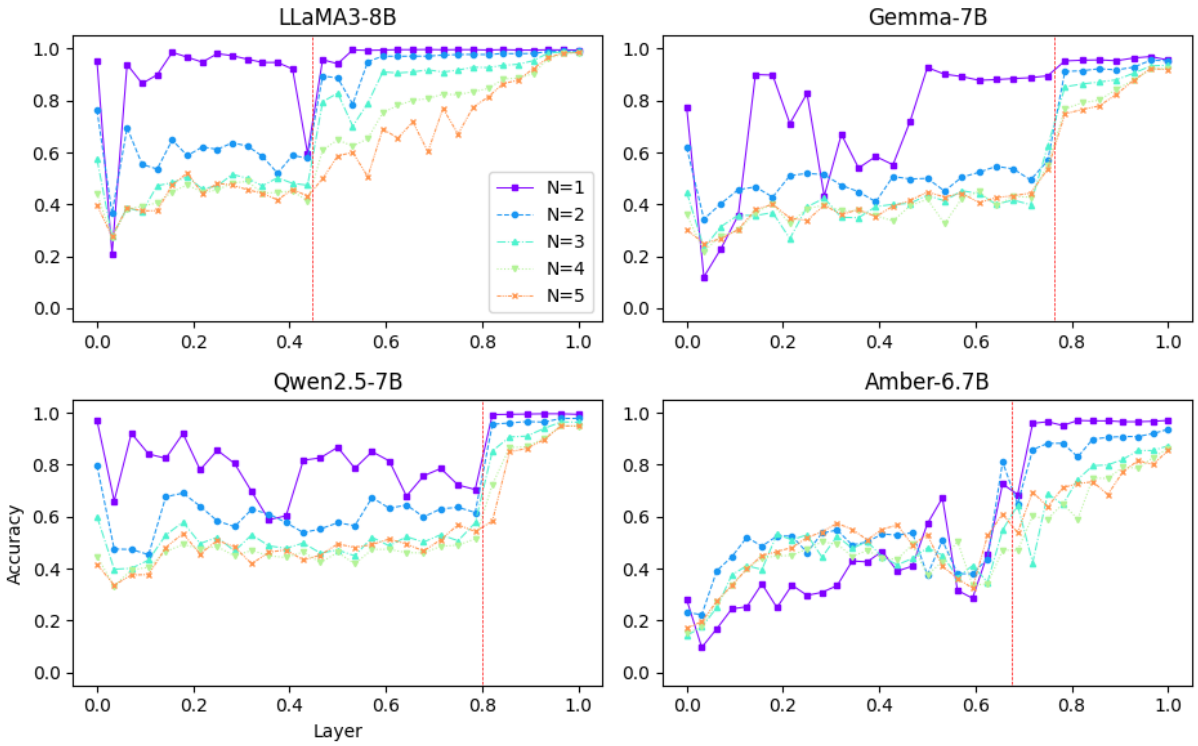

Figure 12: Accuracy of N-th character prediction using the "/" separator, measured via probing classifiers across Transformer layers.

E Neurons for Alphabet

Figure 13 shows the distribution of knowledge neurons associated with the output of each alphabet character. We used the same identification method as described in §7, but focused on individual characters rather than positional indices (N). The results indicate that neurons responsible for alphabet outputs are primarily located in the near-final layers of the models.

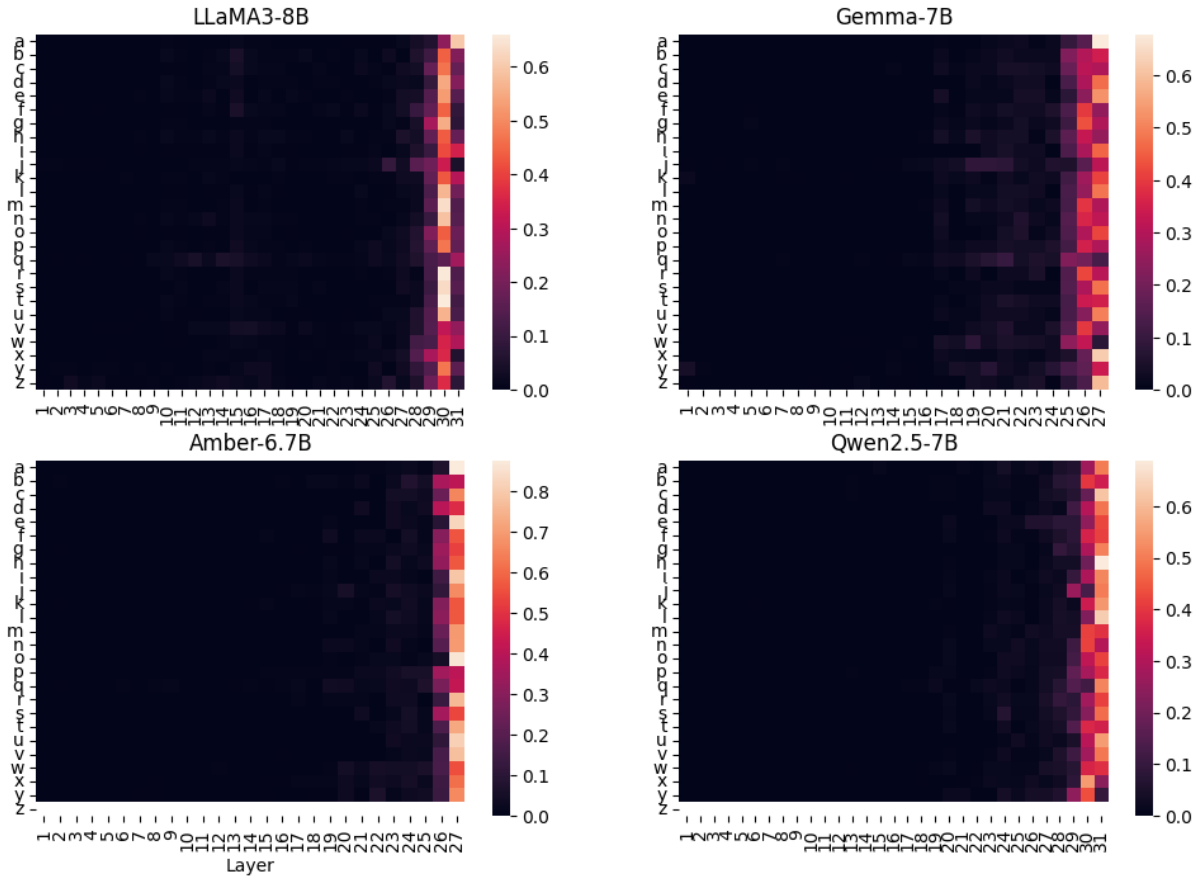

Figure 13: Distribution of knowledge neurons responsible for outputting each alphabet character.