EXECUTE: A Multilingual Benchmark for LLM Token Understanding

Lukas Edman[1,3] **Helmut Schmid**[2] **Alexander Fraser**[1,3,4]

[1]School of Computation, Information and Technology, TU Munich
[2]Center for Information and Language Processing, LMU Munich
[3]Munich Center for Machine Learning
[4]Munich Data Science Institute

lukas.edman@tum.de, schmid@cis.lmu.de

Abstract

The CUTE benchmark showed that LLMs struggle with character understanding in English. We extend it to more languages with diverse scripts and writing systems, introducing EXECUTE. Our simplified framework allows easy expansion to any language. Tests across multiple LLMs reveal that challenges in other languages are not always on the character level as in English. Some languages show word-level processing issues, some show no issues at all. We also examine sub-character tasks in Chinese, Japanese, and Korean to assess LLMs' understanding of character components.

1 Introduction

LLMs perform well on many tasks but struggle when they are asked to manipulate character sequences, as shown by the CUTE benchmark (Edman et al., 2024). While CUTE tested Russian, showing this issue is not language-specific, it failed to consider other linguistic differences that may affect results. Language variation extends beyond script differences to writing system differences. English and Russian use alphabets. Other languages use Abugidas, where letters are not strictly ordered within syllables, or Abjads, which mark vowels with diacritics or not at all. Chinese uses a logographic script, where most words are just 1-2 characters long. Multilingual LLMs allocate tokens unevenly across languages: high-resource languages are well represented, but some low-resource languages are mainly processed at the byte level.

We explore these languages in our benchmark EXECUTE: the **E**xpandable **X**(Cross)-Lingual **E**xtension of **CUTE**.[1] We mainly look at 8 languages, shown in Table 1, which vary in script, writing system, tokenization, and resourcedness. We also provide a framework for adding languages to make this benchmark easily expandable. In our results and analysis, we find that:

[1] https://github.com/Leukas/EXECUTE

Language	Script	Writing System	c/w	t/w	c/t
Amharic	Ge'ez	Abugida	3.71	7.69	0.48
Arabic	Arabic	Abjad	4.63	2.43	1.90
Chinese	Simpl. Han	Logographic	1.51	1.25	1.20
English	Latin	Alphabet	4.04	1.32	3.05
Hindi	Devanagari	Abugida	3.66	2.80	1.31
Japanese	Japanese	Mixed	1.54	1.27	1.22
Korean	Hangul	Featural	3.38	2.71	1.25
Russian	Cyrillic	Alphabet	5.06	2.36	2.14

Table 1: CWT statistics of EXECUTE's languages. c, w, and t denote characters, words, and tokens. c/w refers to the average characters per word. t is the average token count across the 5 tokenizers used by the models.

1. Benchmark results for non-English languages often differ from the English results.
2. The results correlate with the languages' CWT (character-word-token) statistics (see Table 1).
3. Surprisingly, the less an LLM knows a language, the better it performs on EXECUTE.
4. LLMs struggle with understanding sub-character components (see Figure 1).

Our results provide more insight into how LLMs process tokens on different granularities.

2 Related Works

Our work builds upon the CUTE benchmark (Edman et al., 2024), which showed that LLMs struggle with character manipulation tasks. CUTE was mainly created for English but also included Russian tasks, showing similar results. Similar studies probe models to spell or modify text on the character level, but either first train the model (Itzhak and Levy, 2022; Kaushal and Mahowald, 2022), or focus on other topics than orthography (Huang et al., 2023; Efrat et al., 2023).

Research on error correction, including spelling correction, has been done for many languages. Maxutov et al. (2024) found spelling correction to be "hard" for LLMs in Kazakh. Li et al. (2023) reported that LLMs perform worse than fine-tuned models for Chinese spelling correction. Similarly,

Findings of the Association for Computational Linguistics: ACL 2025, pages 1878–1887
July 27 - August 1, 2025

	Task	Input	Output	
Composition	Spelling	Spell out the word: there	t h e r e	8+ languages
	Inverse Spelling	Write the word that is spelled out (no spaces): t h e r e	there	
	Contains Character	Is there a 'c' in 'there'?	No	
	Contains Word	Is there a 'the' in 'the sky is blue'?	Yes	
Manipulation	Character Insertion	Add 'b' after every 'e' in 'there'	thebreb	
	Word Insertion	Add 'is' after every 'the' in 'the sky is blue'	the is sky is blue	
	Character Deletion	Delete every 'e' in 'there'	thr	
	Word Deletion	Delete every 'the' in 'the sky is blue'	sky is blue	
	Character Substitution	Replace every 'e' with 'a' in 'there	thara	
	Word Substitution	Replace every 'the' with 'is' in 'the sky is blue'	is sky is blue	
	Character Swapping	Swap 't' and 'r' in 'there'	rhete	
	Word Swapping	Swap 'the' and 'is' in 'the sky is blue'	is sky the blue	
Sub-Composition	Char to Component	What are the components of '둘'?	ㄷㅜㄹ	Chinese Japanese Korean
	Component to Char	How do you combine 'ㄷㅜㄹ'?	둘	
	Contains Component	Is there a 'ㄷ' in '둘'?	Yes	

Figure 1: EXECUTE benchmark. Prompts shortened for brevity. Example of full prompt in Appendix D.

Kwon et al. (2023) showed that fine-tuned models outperform prompted LLMs for Arabic. Spelling correction requires both character-level and semantic knowledge to determine the correct replacement. EXECUTE, like CUTE, aims to remove contextual semantic understanding from the benchmark.

Our sub-character experiments build on work by Wu et al. (2025) who released a detailed analysis of the information in Chinese characters. Our character-to/from-radical tasks resemble theirs, but they focus on simplified Chinese, while we also examine traditional characters via Japanese Kanji.

Character-level LLMs have been proposed as a solution to CUTE and have been shown to outperform subword LLMs in Pagnoni et al. (2024).

3 Benchmark

Figure 1 exemplifies our EXECUTE benchmark. We use the same composition and manipulation tasks as CUTE but drop the similarity tasks which require static embeddings (such as word2vec) and fluent speakers to define similarity thresholds, which vary by language and lack clear criteria. Their removal makes EXECUTE easier to expand. Adding a new language X now only requires an English$\rightarrow X$ translation system. As cross-language alignment is not crucial, translations do not need to be perfect: grammaticality is preferable but not necessary. We modify prompt examples and the dataset used, so English and Russian results differ from CUTE's.[2] Although perfect translations are not required, we have fluent speakers verify that most translations preserve meaning and grammar. Table 1 lists these languages, covering eight major scripts and all known writing systems. While some widely used languages (e.g. Spanish) are missing, their script is represented, and Appendix B shows the performance of languages using the same script is highly correlated.

We keep the prompt texts in English but use language-specific examples, since fully Russian prompts did not improve performance for Russian (Edman et al., 2024). It also ensures that the LLMs understand the task consistently across languages.

3.1 Data Preparation

We now describe the exact differences in preprocessing steps between our benchmark and the CUTE benchmark. Although there are several changes, we find that the scores from CUTE and EXECUTE are still largely comparable, as shown in Appendix A.

To start, we use an updated subset of 5000 stories from the TinyStories dataset (Eldan and Li, 2023), which used GPT-4 to produce outputs rather than the GPT-3.5 outputs, used in CUTE. We find this dataset to be cleaner (with no random foreign characters), and it is purported by the dataset authors to also be of higher quality. For non-English languages, we translate all the stories using Google Translate. At this point, for Chinese and Japanese, it is necessary to apply word segmentation. For Chinese, we use `jieba`[3], and for Japanese we use `nagisa`.[4]

We then generate a character set and vocabulary from the translated stories to use for our tasks. This is unlike what is used in CUTE, which predefined alphabets and vocabularies taken from the Trillion Word Corpus and Wikipedia. This change is necessary as it is more difficult to define a strict character set for some languages, and also more difficult to find a vocabulary.

In the CUTE benchmark, the vocabulary also removed words less than 3 characters to maintain a level of difficulty for the tasks. We remove this cutoff for Chinese, Japanese, and Korean, as it is too restrictive.

As the prompts are few-shot, we require language-specific examples in each prompt. For CUTE, these examples were created manually. Instead, we generate 4 additional examples in the same manner as our test set, with a few additional stipulations:

- At least 2 examples must use a word that contains duplicate letters.

[2] As our changes are minor, users of the English and Russian datasets should cite Edman et al. (2024).

[3] https://github.com/fxsjy/jieba

[4] https://github.com/taishi-i/nagisa

- At least 1 example must operate on the duplicate letters when applicable.
- For the contains tasks, 2 examples must have the label "yes" and 2 "no".

We specify the duplicate letter restrictions so that the LLM understands that it must modify *all* of the targeted characters. The first two restrictions were not applied for Chinese however, as it is exceedingly rare for a Chinese word to contain duplicate characters. The last restriction is intended to ensure the model is not biased to answering either "yes" or "no" due to its frequency in the examples, a phenomenon which has been shown to be problematic in Zhao et al. (2021).

Diacritics Abugidas such as Hindi have diacritics to mark vowel sounds, aspirations, and nasalizations. Due to the complex rules surrounding valid diacritics, which also vary between languages, we opt to consider each "character" as the letter plus any diacritics attached, also known as the grapheme. This is already the case for Amharic, as the diacritics have become fused with consonants in the Ge'ez script itself.

3.2 Sub-Character Experiments

Chinese, Japanese, and to a lesser extent Korean, have few characters per word, so we add language-specific tasks to assess their understanding of character components.

In Chinese, each character can be broken down into parts known as Kangxi radicals. An example of a decomposition is: 晚 → ⿰日免, where ⿰ indicates that 日 should be placed to the left of 免. The radicals often have a related meaning to the composite: 晚 means *evening*, 日 means *sun* and 免 means *avoid*. Japanese Kanji characters originate from traditional Chinese characters and can also be decomposed into radicals. Korean Hangul characters denote syllables and can be split into Jamo, which correspond to phonemes. For example, 둘 (*dul*) becomes ㄷ (*d*), ㅜ (*u*), and ㄹ (*l*).

We test the LLMs' ability to compose and decompose CJK characters into their components. For Chinese and Japanese, we ask the model to split characters into Kangxi radicals, and vice versa.[5] Similarly, we decompose Hangul characters to Jamo and vice versa. These tasks are analogous to the `spelling` and `inverse_spelling` tasks. We further add a task (similar to `contains`) which asks if a character contains a Kangxi radical or Jamo.

Japanese can either be written with Kanji characters or with phonetic Hiragana characters. We test LLMs' ability to convert Kanji in Appendix C.

3.3 Models

We test 5 popular open-source multilingual LLMs: Aya Expanse, Gemma 2, Llama 3.1 and 3.3, Qwen 2.5, and Mistral (Dang et al., 2024; Gemma Team et al., 2024; Dubey et al., 2024; Qwen et al., 2025; Jiang et al., 2023). Their sizes range from 7B to 70B parameters, and their vocabularies contain between 128k and 256k tokens.

4 Results

				Eng		
	Amh	Tzm	Sat	Cipher	Byte	Reg
Spell	96.3	100.0	97.6	100.0	85.0	99.5
Inv Spell	99.8	100.0	99.3	100.0	0.0	99.6
Cont Char	91.8	100.0	98.0	98.6	82.8	75.7
Cont Word	99.6	99.2	98.9	99.0	96.7	99.9
Ins Char	97.8	97.8	98.2	98.6	20.9	13.5
Ins Word	92.8	94.3	91.9	97.1	1.4	96.6
Del Char	97.6	99.7	98.7	98.9	78.8	67.5
Del Word	97.6	76.2	88.8	95.6	3.7	96.5
Sub Char	96.6	98.4	98.3	95.5	61.5	51.4
Sub Word	96.2	96.6	90.1	98.4	5.9	98.5
Swap Char	93.7	97.6	92.8	98.3	29.0	12.7
Swap Word	97.3	87.9	90.0	95.9	6.6	90.9
Avg	96.4	95.6	95.2	98.0	39.4	75.2

Table 2: Llama 3.3 on low-resource languages.

We first examine results by language, showing the best model performance for each in Figure 2.[6] Russian and Arabic results resemble English results. Hindi and Korean perform better at the word level than the character level, though the gap is smaller than for English, with stronger results in character-level insertion and swapping. Japanese and Chinese perform better on the character level, which is expected since each character is a word or almost a word. However, word-level tasks may simply be harder in these languages, as they require modifying multiple tokens instead of just one.

4.1 Amharic and Low-Resourcedness

Amharic stands out from the rest of the results in that the performance is nearly perfect in the best-case scenario. This is particularly surprising as Amharic is the lowest-resource language of the 8, and most characters are split into bytes

[5] One can further split Kangxi radicals down to strokes, but this showed very poor performance in initial tests.

[6] We show the results per task, as well as results for Aya and Mistral, in Appendix E.

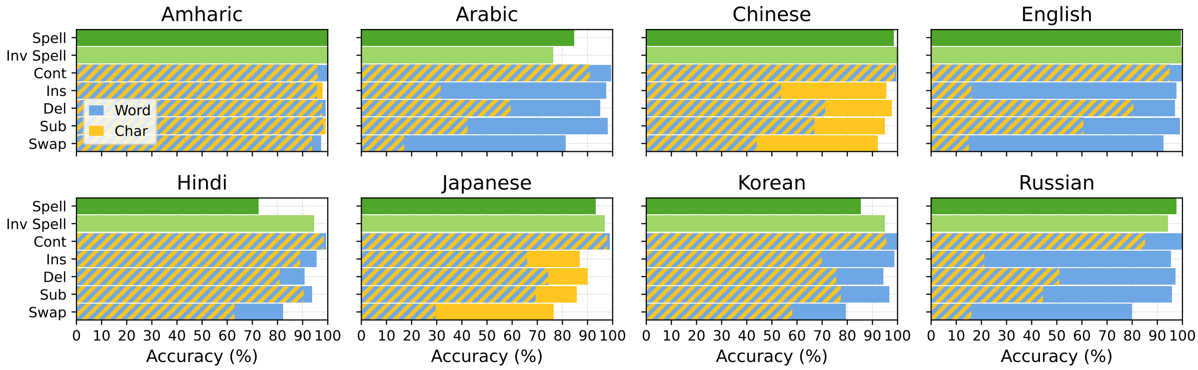

Figure 2: The best result of all models for each language and task.

by the tokenizers, meaning each character is represented by 3 tokens. We suspect that the good performance might actually be *because* of this low-resourcedness. As seen in Edman et al. (2024), and also observed in this work, LLMs are biased to generating real words and grammatical sentences. Their lack of understanding of Amharic might weaken this bias.

	Gemma 2		Llama 3.1		Llama 3.3	Qwen 2.5	
	9B	27B	8B	70B	70B	7B	32B
Amh	80.5	85.3	75.7	**95.9**	96.4	41.9	74.4
Ara	51.6	62.3	52.1	68.1	67.8	47.2	**68.6**
Zho	70.2	74.4	71.3	81.1	79.7	70.4	**83.6**
Eng	64.8	71.6	61.9	75.7	75.2	62.1	**77.3**
Hin	47.9	47.1	43.8	54.0	56.4	43.5	**86.2**
Jpn	60.1	65.2	58.8	73.1	74.6	62.1	**77.9**
Kor	73.6	**80.8**	62.1	76.9	76.1	60.2	**80.8**
Rus	53.6	62.6	51.0	67.8	67.8	52.1	**71.2**
Avg	62.8	68.7	59.6	74.1	74.3	54.9	**77.5**

Table 3: Average score per language. Best in bold.

We provide further evidence that language knowledge inversely correlates with EXECUTE performance by adding two low-resource languages, Tamazight and Santali. Their unique scripts (Tifinagh and Ol Chiki) are not used by any higher resource languages, forcing LLM tokenizers to operate at the byte level. These languages were likely seen rarely, if ever, during training. We compare results on Amharic's best-performing model, Llama 3.3. Additionally, we test two variations of English: one encodes text using a cipher that maps Latin to Amharic characters, and the other forces the inputs to be byte-level (retaining the Latin alphabet). These experiments assess whether byte-level operation alone improves performance or if eliminating English recognition via ciphering is also necessary. We expect ciphered English to perform similarly to Amharic. Table 2 shows that Llama achieves near-perfect results in the low-resource languages, as well as the ciphered English. Byte-level English improves character tasks but fails on word tasks, partly due to bias. The model rarely sees English at the byte level except in social media, leading to random casing and antspeak (extra spacing) in the output. Degenerate output (e.g. "1 1 1 ...") also occurs. Some fine-tuning with this byte-level approach would likely increase performance considerably.

Does Amharic's near-perfect score mean character- and word-level processing is solved? No, it shows LLMs can perform arbitrary manipulations but are hampered by their language understanding. As training data increases, Amharic performance will likely decline. So, this benchmark should complement standard NLU benchmarks for a complete assessment.

4.2 Language Clusters

Table 1 groups languages with similar CWT statistics into five categories: 1) Arabic & Russian, 2) Hindi & Korean, 3) Japanese & Chinese, 4) Amharic, and 5) English. Their similar benchmark performance suggests that segmentation, whether natural or from tokenization, impacts results. As expected, the statistics of Tamazight (4.37, 8.83, 0.49) and Santali (3.54, 8.38, 0.42) closely align with Amharic.

4.3 Model Performance

Table 3 shows model performance. Larger models generally perform better. However, this trend does not hold across model families, as Qwen 2.5 (32B) outperforms the larger 70B Llama 3 models. Llama 3.3, despite its stronger performance than Llama 3.1 on standard benchmarks, performs similarly here.

		Aya Expanse		Gemma 2		Llama 3.1		Llama 3.3	Qwen 2.5		Mistral	
		8B	32B	9B	27B	8B	70B	70B	7B	32B	8B	24B
Zho	Char to Rad	0.0	2.0	0.0	0.8	0.0	1.4	1.8	1.4	**16.4**	0.8	3.5
	Rad to Char	1.4	8.2	2.5	0.8	2.5	10.7	11.9	7.6	**22.8**	2.5	8.2
	Contains Rad	55.4	65.5	**81.1**	79.3	69.4	73.5	72.9	68.0	78.8	62.2	74.5
Jpn	Char to Rad	0.0	0.7	0.7	0.0	0.0	0.0	2.2	2.2	**9.2**	0.4	3.7
	Rad to Char	2.6	8.5	2.2	0.4	1.9	13.7	13.3	5.2	**20.3**	1.5	7.4
	Contains Rad	57.9	61.6	**86.4**	72.7	69.7	73.1	76.0	73.4	76.0	65.7	83.4
Kor	Hangul to Jamo	7.5	48.1	54.6	65.3	24.5	48.8	45.6	24.7	57.4	35.6	**66.4**
	Jamo to Hangul	24.7	49.0	47.2	**63.9**	41.7	24.5	24.3	25.9	42.2	28.1	51.5
	Contains Jamo	63.7	76.2	93.2	**96.6**	92.1	87.5	90.3	75.3	93.4	73.2	88.7

Table 4: Sub-character-level results on CJK languages. Best in bold.

Edman et al. (2024) found that more training data improved results on CUTE, but we find no such trend. Among 7-9B models, Gemma was trained on 8T tokens, Llama on 15T, and Qwen on 18T (Gemma Team et al., 2024; Qwen et al., 2025; Dubey et al., 2024), yet their performance is inversely correlated. While this may be coincidental, results on Amharic, Tamazight, and Santali raise doubts about whether more training data improves performance on this benchmark.

4.4 Sub-Character Performance

Table 4 shows sub-character results. For Japanese and Chinese, models struggle to translate characters to and from their radical components but perform better on the `Contains` task, as it only requires identifying one radical. While some characters, like 晚 (*evening*), have components that clearly contribute to meaning, others are more ambiguous. For example, 木 (*tree*) is likely easier for models to identify in 樟 (*camphor tree*) compared to 章 (*chapter*, *seal*).

LLMs are notably better at converting between Hangul and Jamo, likely due to Hangul's simpler structure or its more frequent decomposition in training data. However, the conversion still falls short of the near-perfect scores seen in the main `Spelling` and `Inverse Spelling` tasks.

5 Conclusion

We present a multilingual, multi-script extension of the CUTE benchmark to test token understanding in a variety of languages. The benchmark is designed to be easily expanded to new languages, allowing the token understanding of LLMs to be tested in any language. Our findings show that manipulation on the character level is challenging in some non-English languages, but word-level manipulation is challenging for some languages too. Understanding the components of characters in Chinese, Japanese, and Korean is also lacking. The performance of a language can be somewhat predicted by its character-word-token ratios. Surprisingly, LLMs perform better on lower-resourced languages, due to their knowledge of high-resourced languages acting as a bias against the benchmark's tasks. While Edman et al. (2024) hypothesized that character-level models would be promising for solving the CUTE benchmark, EXECUTE demonstrates an additional need for debiasing models so they can temporarily forget what they know about a language.

6 Limitations

We limit ourselves to 8 languages for the majority of this work. While we argue that the languages in the same scripts as the ones tested will likely have similar results, pointing to the correlation in results between English, Spanish, German, and Xhosa in Appendix B, we cannot know for sure without testing them all. Several other scripts are not covered which may have differing performances.

We also do not test very large language models above 70B parameters due to compute constraints. The CUTE benchmark added scores for the 405B parameter Llama 3.1 and found it made improvements across the board, but was still lacking on character-level insertion and swapping. We would expect similar improvements for our English results, but it is unclear how it would perform for other languages.

7 Acknowledgments

The work was supported by the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (grant agreement No. 101113091) and by the German Research Foundation (DFG; grant FR 2829/7-1).

References

John Dang, Shivalika Singh, Daniel D'souza, Arash Ahmadian, Alejandro Salamanca, Madeline Smith, Aidan Peppin, Sungjin Hong, Manoj Govindassamy, Terrence Zhao, et al. 2024. Aya expanse: Combining research breakthroughs for a new multilingual frontier. *arXiv preprint arXiv:2412.04261*.

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.

Avia Efrat, Or Honovich, and Omer Levy. 2023. LMentry: A language model benchmark of elementary language tasks. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10476–10501, Toronto, Canada. Association for Computational Linguistics.

Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english? *Preprint*, arXiv:2305.07759.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Jing Huang, Zhengxuan Wu, Kyle Mahowald, and Christopher Potts. 2023. Inducing character-level structure in subword-based language models with type-level interchange intervention training. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12163–12180, Toronto, Canada. Association for Computational Linguistics.

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.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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.

Sang Kwon, Gagan Bhatia, El Moatez Billah Nagoudi, and Muhammad Abdul-Mageed. 2023. Beyond English: Evaluating LLMs for Arabic grammatical error correction. In *Proceedings of ArabicNLP 2023*, pages 101–119, Singapore (Hybrid). Association for Computational Linguistics.

Yinghui Li, Haojing Huang, Shirong Ma, Yong Jiang, Yangning Li, Feng Zhou, Hai-Tao Zheng, and Qingyu Zhou. 2023. On the (in)effectiveness of large language models for chinese text correction. *Preprint*, arXiv:2307.09007.

Akylbek Maxutov, Ayan Myrzakhmet, and Pavel Braslavski. 2024. Do LLMs speak Kazakh? a pilot evaluation of seven models. In *Proceedings of the First Workshop on Natural Language Processing for Turkic Languages (SIGTURK 2024)*, pages 81–91, Bangkok, Thailand and Online. Association for Computational Linguistics.

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srinivasan Iyer. 2024. Byte latent transformer: Patches scale better than tokens. *Preprint*, arXiv:2412.09871.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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. *Preprint*, arXiv:2410.09013.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR.

A Comparison to CUTE

In Table 5, we run Llama 3.1 8B on CUTE and compare the results to English and Russian EXECUTE. The results are very similar, with `Insert Word` appearing slightly easier in CUTE. This confirms that our changes did not dramatically alter any results.

	EXECUTE		CUTE	
	Eng	Rus	Eng	Rus
Spell	98.7	**72.1**	**99.8**	64.5
Inv Spell	96.2	37.9	**98.4**	**74.1**
Cont Char	65.1	57.1	**67.1**	**68.4**
Cont Word	**97.3**	**97.8**	86.8	97.4
Ins Char	**4.4**	6.7	4.2	**7.6**
Ins Word	48.2	48.5	**62.0**	**59.2**
Del Char	56.1	33.1	**56.6**	**43.7**
Del Word	76.2	**91.8**	**83.7**	82.3
Sub Char	**39.3**	29.0	34.4	**33.8**
Sub Word	**94.1**	**87.0**	90.4	76.7
Swap Char	**6.6**	4.8	6.1	**5.2**
Swap Word	60.4	**46.5**	**63.7**	33.3
Average	61.9	51.0	**62.8**	**53.9**

Table 5: EXECUTE versus CUTE with Llama 3.1 8B.

B Language Similarity

We conduct similarity tests to see how similar the trends are across languages. We conduct a Pearson correlation between two languages for each task for a given model and average the models' correlations together. We show the similarity of the languages as determined by the average correlation of the results from the 5 LLMs of size 7-9B in Table 6. The languages are not particularly similar to one another, apart from Japanese and Chinese (which share some characters) and Arabic and Russian. The similarity between Arabic and Russian is not entirely clear, though it could be that their ratios of characters-per-word and characters-per-token are quite similar (such is also the case for Japanese and Chinese).

	Ara	Zho	Eng	Hin	Jpn	Kor	Rus
Amh	0.64	0.01	0.66	0.33	0.23	0.62	0.60
Ara		-0.11	0.76	0.44	0.13	0.85	0.92
Zho			0.17	0.65	0.93	0.26	-0.06
Eng				0.45	0.36	0.77	0.86
Hin					0.75	0.71	0.38
Jpn						0.49	0.17
Kor							0.84

Table 6: Average correlations between the results for each language pair.

We also correlate the results from English to other Latin-scripted languages, German, Spanish, and Xhosa, in Table 7. Here we see the average correlation is at least 95% between English, German, and Spanish, and at least 85% to Xhosa. This suggests that the results for other Latin-scripted languages will likely not deviate too much, even if the languages are distant in relation and differing in resourcedness.

	Deu	Spa	Xho
Eng	0.96	0.95	0.85
Deu		0.99	0.90
Spa			0.90

Table 7: Average correlations between the results for Latin-scripted languages.

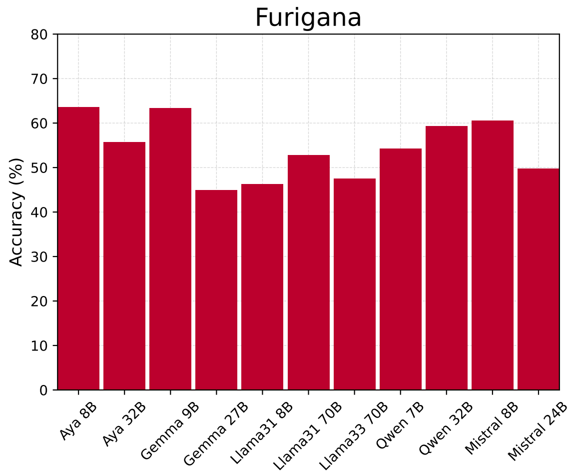

Figure 3: Performance on Kanji to Hiragana conversion.

C Japanese Furigana

Aside from Kanji, Japanese has two other writing forms: Hiragana and Katakana. Typical Japanese text will use all three forms, with several words being a combination of Kanji and Hiragana, and even in rare cases, all three. While Kanji is logographic like Chinese, Hiragana and Katakana are syllabaries. Kanji and Hiragana are the most used, while Katakana is typically only used for foreign words or onomatopoeiae. As such, we focus on Kanji and Hiragana. All Kanji characters can be written as Hiragana, and Kanji is sometimes annotated with its corresponding Hiragana as a method of learning the pronunciation of Kanji characters. This practice is known as Furigana. So we use this Furigana method as a test in our benchmark, prompting the model to translate Kanji to Hiragana.[7] With this, we are essentially testing if the LLMs have a phonetic understanding of the Kanji.

In Figure 3, we see the models' results on the Furigana task. Similar to the Korean Hangul to Jamo, the Kanji to Hiragana tasks show that the LLMs generally understand the task, but have not perfected it. Unlike the other sub-character tasks, converting from Kanji to Hiragana cannot be done purely visually. This requires knowledge of how a Kanji sounds, and which Hiragana denote which sounds. From this, we can see a partial understanding.

```
[INST] Spell out the word, putting spaces
between each letter, based on the following
examples:

1. Spell out the word “ かわいい ”.
Answer: “ かわいい ”
2. Spell out the word “ 出し ”. Answer: “
出し ”
3. Spell out the word “ 応援 ”. Answer: “
応援 ”
4. Spell out the word “ 親友 ”. Answer: “
親友 ”

Question: Spell out the word “ 実行
する ”. [/INST] Answer: “ 実行する ”
```

Figure 4: Example of full prompt for Japanese spelling, with intended output in red. [INST] and [/INST] denote any tokens added to enable normal behavior from each LLM.

D Full Prompt Example

We show an example of a full prompt in Figure 4.

E Full Results

The complete results on EXECUTE for all the models tested are shown in Tables 8 and 9.

[7] We do not do the reverse as multiple Kanji can have the same phoneme, e.g. 考 and 好 both denote *ko*.

		Aya Expanse		Gemma 2		Llama 3.1		Llama 3.3	Qwen 2.5		Mistral	
		8B	32B	9B	27B	8B	70B	70B	7B	32B	8B	24B
Amharic	Spell	25.6	72.4	99.5	91.2	98.6	98.4	96.3	0.9	14.3	97.0	99.8
	Inv Spell	77.6	71.5	99.1	91.4	98.8	99.8	99.8	8.2	52.4	99.1	100.0
	Cont Char	58.5	85.8	73.0	81.9	90.4	91.8	91.8	63.2	93.5	94.7	95.8
	Cont Word	55.9	69.5	97.0	98.3	78.5	99.5	99.6	71.7	99.9	95.9	98.8
	Ins Char	35.2	58.2	57.1	65.5	26.1	92.3	97.8	10.3	57.9	60.6	92.2
	Ins Word	70.7	66.7	95.6	91.7	28.7	93.3	92.8	58.4	92.6	70.5	87.9
	Del Char	54.5	85.0	84.5	89.4	89.6	96.8	97.6	43.5	78.6	85.3	97.7
	Del Word	85.8	91.2	63.1	79.2	93.2	99.1	97.6	70.0	91.1	91.9	89.4
	Sub Char	63.6	82.9	70.1	85.2	87.5	94.9	96.6	29.6	67.3	76.6	99.1
	Sub Word	91.9	98.1	98.1	93.2	91.6	96.7	96.2	79.2	90.9	90.3	96.5
	Swap Char	20.2	53.8	53.0	70.7	52.8	91.3	93.7	12.8	65.8	48.8	86.4
	Swap Word	60.7	84.5	75.7	86.0	73.0	96.5	97.3	55.0	88.1	66.0	94.9
Arabic	Spell	48.7	74.1	36.0	69.9	50.8	84.7	81.0	20.7	52.8	19.3	40.0
	Inv Spell	48.4	64.7	48.3	63.6	44.9	69.7	63.0	39.2	76.4	27.6	60.9
	Cont Char	63.9	74.5	70.3	70.0	70.6	77.4	76.4	74.0	90.7	72.0	78.6
	Cont Word	88.1	97.9	99.0	98.7	96.9	99.1	99.1	95.5	99.4	88.7	99.4
	Ins Char	13.7	8.3	7.6	16.1	2.9	15.7	17.8	11.7	31.4	4.7	12.5
	Ins Word	35.8	61.1	90.8	97.5	51.2	89.1	96.4	61.3	96.3	45.1	86.4
	Del Char	36.0	56.4	36.3	45.5	45.0	55.8	59.4	40.7	53.1	20.8	29.9
	Del Word	74.0	90.4	64.6	83.4	92.5	95.0	88.6	82.0	91.6	63.7	88.8
	Sub Char	17.6	26.6	20.7	33.9	24.3	38.7	36.0	23.0	42.2	11.7	20.7
	Sub Word	72.7	92.6	95.0	97.3	90.8	97.2	98.0	79.3	95.4	77.1	92.2
	Swap Char	5.9	9.1	4.5	8.0	8.1	17.0	16.7	4.0	14.3	2.5	7.5
	Swap Word	26.3	55.1	46.3	63.1	47.6	78.0	81.3	34.8	79.8	29.0	62.0
Chinese	Spell	83.2	93.0	84.3	91.3	93.6	98.4	98.0	96.3	98.2	93.4	90.9
	Inv Spell	98.5	97.3	95.1	96.2	98.6	98.4	98.7	98.9	99.8	98.8	99.6
	Cont Char	84.0	97.2	96.4	95.4	92.5	97.3	91.1	98.7	98.9	96.0	98.7
	Cont Word	84.1	99.0	99.4	98.8	91.7	95.1	86.7	94.3	98.1	94.2	94.8
	Ins Char	70.0	57.2	78.6	81.4	67.3	90.6	92.6	73.6	95.4	53.2	89.3
	Ins Word	28.4	33.2	41.9	53.0	20.5	46.5	47.5	43.7	53.4	34.1	50.5
	Del Char	79.6	90.0	85.4	88.1	86.8	97.0	97.6	89.2	97.3	88.3	94.8
	Del Word	38.4	57.3	46.8	56.9	59.4	71.1	70.1	54.0	65.1	53.2	67.4
	Sub Char	60.6	68.4	69.9	75.1	84.0	94.2	94.8	80.0	94.6	75.0	91.3
	Sub Word	40.2	54.9	55.3	64.7	47.5	56.3	53.4	43.1	66.8	46.1	66.5
	Swap Char	63.9	71.9	73.3	69.8	90.6	92.0	92.1	62.6	92.0	75.4	90.5
	Swap Word	14.2	26.6	15.6	21.9	22.8	36.1	33.3	10.7	43.9	15.1	35.0
English	Spell	96.7	98.5	99.3	99.5	98.7	99.5	99.5	94.7	98.6	97.3	99.0
	Inv Spell	95.4	98.5	99.3	99.6	96.2	99.8	99.6	98.3	99.2	91.0	98.7
	Cont Char	62.6	73.1	68.0	69.5	65.1	80.3	75.7	81.9	94.8	66.7	83.5
	Cont Word	94.3	99.4	99.7	99.7	97.3	100.0	99.9	98.1	100.0	93.3	99.9
	Ins Char	11.8	6.0	9.2	7.8	4.4	10.9	13.5	7.1	15.9	7.4	4.4
	Ins Word	39.9	60.6	86.7	96.8	48.2	94.9	96.6	70.2	97.6	51.9	72.9
	Del Char	35.0	56.3	58.5	80.4	56.1	68.3	67.5	56.8	70.5	33.6	72.4
	Del Word	60.3	77.5	53.7	77.7	76.2	97.1	96.5	78.5	95.7	74.3	69.8
	Sub Char	27.7	42.2	35.4	60.5	39.3	53.5	51.4	29.0	52.1	27.7	51.7
	Sub Word	82.4	96.1	94.9	97.9	94.1	98.1	98.5	92.9	99.0	92.9	97.0
	Swap Char	6.0	9.8	6.9	8.9	6.6	14.9	12.7	3.2	11.5	6.8	11.0
	Swap Word	22.6	64.5	66.2	60.8	60.4	91.0	90.9	34.7	92.4	42.2	85.9

Table 8: Results for Amharic, Arabic, Chinese, and English.

		Aya Expanse		Gemma 2		Llama 3.1		Llama 3.3	Qwen 2.5		Mistral	
		8B	32B	9B	27B	8B	70B	70B	7B	32B	8B	24B
Hindi	Spell	48.4	69.4	12.6	16.0	46.2	72.5	71.3	20.5	57.7	23.4	58.7
	Inv Spell	76.8	83.2	71.9	83.4	76.0	92.7	92.9	71.2	94.6	61.9	87.2
	Cont Char	68.4	85.6	75.1	75.7	72.6	87.3	83.2	90.9	98.3	82.4	88.9
	Cont Word	89.4	98.7	95.7	98.5	93.3	98.9	93.0	94.7	99.3	81.1	94.8
	Ins Char	41.0	10.8	25.8	29.4	15.8	29.0	32.9	45.9	89.0	27.5	36.5
	Ins Word	6.3	11.7	77.3	41.8	13.6	25.5	47.6	31.8	95.5	9.1	16.4
	Del Char	58.8	66.5	50.9	67.6	66.5	76.9	76.9	50.9	80.7	44.3	79.5
	Del Word	6.6	26.6	26.6	24.4	40.6	25.7	31.7	29.4	90.8	13.9	19.7
	Sub Char	45.7	66.4	38.2	56.0	42.1	61.4	63.6	50.8	90.2	33.1	65.7
	Sub Word	3.7	14.9	62.3	15.9	19.1	23.8	25.4	15.3	93.8	4.9	17.0
	Swap Char	14.5	24.5	8.4	29.4	19.5	15.1	22.2	9.9	62.8	21.6	33.6
	Swap Word	2.1	16.9	30.6	27.5	20.1	39.4	35.7	10.3	82.2	3.0	11.5
Japanese	Spell	52.9	83.2	68.5	71.0	73.2	93.3	92.2	72.3	86.5	77.2	84.5
	Inv Spell	92.4	88.4	87.8	90.5	78.1	96.9	95.1	90.4	95.9	88.5	96.7
	Cont Char	67.9	91.6	90.9	88.8	87.0	95.2	93.4	93.7	97.8	86.7	93.0
	Cont Word	82.4	92.7	98.8	97.3	90.0	89.4	85.5	89.9	94.8	84.6	93.9
	Ins Char	48.3	31.5	58.3	68.8	18.6	72.1	77.6	69.7	86.8	29.3	78.5
	Ins Word	21.4	34.2	41.7	61.0	10.6	42.1	47.2	44.0	65.9	30.4	46.2
	Del Char	62.0	78.3	60.5	64.4	79.8	88.2	90.0	70.0	84.3	67.1	86.5
	Del Word	31.9	60.4	49.3	52.5	62.4	68.5	74.2	52.3	61.4	38.0	63.5
	Sub Char	57.2	68.8	58.5	66.3	75.8	83.2	83.7	64.4	85.7	59.3	81.7
	Sub Word	32.4	55.1	48.9	58.8	50.9	62.8	65.1	50.9	69.3	38.5	60.6
	Swap Char	40.8	52.0	50.2	49.2	66.6	63.9	66.7	40.4	76.6	45.7	70.1
	Swap Word	5.9	13.2	8.1	13.3	12.8	21.9	25.0	7.5	29.4	9.3	20.9
Korean	Spell	43.6	71.5	67.5	85.4	51.5	78.0	73.4	42.5	66.6	42.3	56.1
	Inv Spell	85.0	93.8	82.6	86.8	71.7	88.9	88.8	84.2	94.8	64.6	92.2
	Cont Char	71.1	84.5	81.1	86.2	81.6	85.3	74.4	90.7	95.5	76.4	89.0
	Cont Word	92.6	97.2	98.8	99.0	95.6	99.4	99.1	96.7	99.6	84.9	98.7
	Ins Char	33.6	20.6	47.4	54.6	21.0	52.5	60.8	39.6	69.7	20.2	41.7
	Ins Word	36.4	72.0	93.3	98.6	50.8	91.8	94.1	66.8	92.5	45.0	87.7
	Del Char	44.7	64.6	69.1	75.6	62.5	62.2	63.2	44.6	64.1	43.2	56.6
	Del Word	56.5	81.8	77.5	88.6	91.9	94.3	86.3	76.7	91.0	75.4	90.7
	Sub Char	39.1	62.0	71.4	77.4	56.0	65.5	63.9	50.4	67.6	37.9	53.0
	Sub Word	48.9	78.7	90.1	96.7	73.0	91.2	92.1	68.8	90.5	65.1	89.6
	Swap Char	27.3	33.3	48.9	47.5	37.3	42.0	44.0	31.4	58.1	20.0	33.0
	Swap Word	22.6	48.3	55.8	73.6	52.5	71.1	72.8	29.9	79.3	27.1	61.1
Russian	Spell	40.9	80.4	54.5	88.3	72.1	97.6	96.7	54.0	88.9	79.8	94.8
	Inv Spell	54.1	78.7	71.4	90.3	37.9	86.6	86.0	81.9	94.2	39.5	88.0
	Cont Char	55.0	68.8	59.7	58.7	57.1	60.2	59.1	75.3	84.9	63.3	77.6
	Cont Word	96.9	98.1	99.6	99.8	97.8	99.8	99.9	95.8	99.5	94.7	99.5
	Ins Char	10.9	3.8	7.0	9.4	6.7	11.0	12.0	7.2	21.1	5.7	11.3
	Ins Word	29.3	61.9	90.6	95.4	48.5	90.7	93.1	77.8	95.1	62.9	84.2
	Del Char	12.7	34.7	20.9	38.1	33.1	49.9	50.9	24.8	44.7	18.9	45.1
	Del Word	68.6	85.0	75.0	81.0	91.8	97.3	96.7	83.1	90.7	80.1	85.7
	Sub Char	15.5	26.0	17.0	35.3	29.0	38.7	40.2	22.6	44.3	16.2	39.7
	Sub Word	63.8	86.2	94.5	95.1	87.0	95.6	95.8	80.8	95.2	86.5	95.6
	Swap Char	1.3	4.8	2.6	4.0	4.8	12.5	13.6	1.4	16.0	4.4	12.0
	Swap Word	26.1	48.1	50.7	55.8	46.5	74.3	70.0	20.7	79.9	32.6	79.5

Table 9: Results for Hindi, Korean, Japanese, and Russian.