CUTE: Measuring LLMs’ Understanding of Their Tokens

Lukas Edman[1,3] **Helmut Schmid**[1] **Alexander Fraser**[2,3,4]
[1]Center for Information and Language Processing, LMU Munich
[2]School of Computation, Information and Technology, TU Munich
[3]Munich Center for Machine Learning
[4]Munich Data Science Institute
`lukas@cis.lmu.de, schmid@cis.lmu.de`

Abstract

Large Language Models (LLMs) show remarkable performance on a wide variety of tasks. Most LLMs split text into multi-character tokens and process them as atomic units without direct access to individual characters. This raises the question: To what extent can LLMs learn orthographic information? To answer this, we propose a new benchmark, CUTE, which features a collection of tasks designed to test the orthographic knowledge of LLMs. We evaluate popular LLMs on CUTE, finding that most of them seem to know the spelling of their tokens, yet fail to use this information effectively to manipulate text, calling into question how much of this knowledge is generalizable.

1 Introduction

Large Language Models (LLMs) attract a lot of interest due to their strong performance on many NLP tasks. They have demonstrated a level of fluency rivaling humans. However, it is often overlooked that LLMs lack direct access to the characters composing their tokens. They can only infer knowledge about the characters from the context during pretraining or instruction tuning. While there are models that use characters as input units, none of them have been instruction-tuned (to our knowledge).

Our work examines how well LLMs understand the composition of their tokens. This knowledge enables LLMs to better generalize to new languages and to perform well on a variety of tasks involving character-level understanding. Tasks such as word puzzles, poetry generation (e.g. alliterations), or parsing ciphers all require very explicit use of characters to achieve. More popular tasks such as code completion, morphological inflection, or spelling correction also require character-level information to a lesser extent, though these tasks also require semantic knowledge, which we wish to ablate.

We introduce **C**haracter-level **U**nderstanding of **T**okens **E**valuation (CUTE)[1], a benchmark consisting of several tasks designed to be easy for humans to complete, given our ability to process characters individually. We evaluate several LLMs ranging from 7B to 132B parameters in size on CUTE to answer the following questions:

1. Do LLMs know which characters make up their tokens?
2. Do LLMs understand the difference between semantic and orthographic similarity?
3. Can LLMs manipulate text at the character level?

We address these questions with a set of tasks primarily on the character level, and additionally on the word level to see the difference in performance on the two granularities, thereby separating the understanding of the task from the understanding of what makes up a token.

2 Related Work

Itzhak and Levy (2022) mostly analyze encoder-only models and test if they understand how to spell words after fine-tuning on 32k examples. They conclude that models learn to spell their tokens “to some extent.” They also experiment with GPT-2 (Radford et al., 2019) which performed similarly to the other models, and also used training examples. Kaushal and Mahowald (2022) probe models with a task asking if a letter is in a word (similar to our `character contains` task, see §3). Similar to Itzhak and Levy (2022), their probe requires training, as they use models of similar size. By contrast, we examine models with 10 to 200 times as many parameters and apply few-shot prompting without fine-tuning.

Huang et al. (2023) experiment with spelling correction, unscrambling words, and finding words in

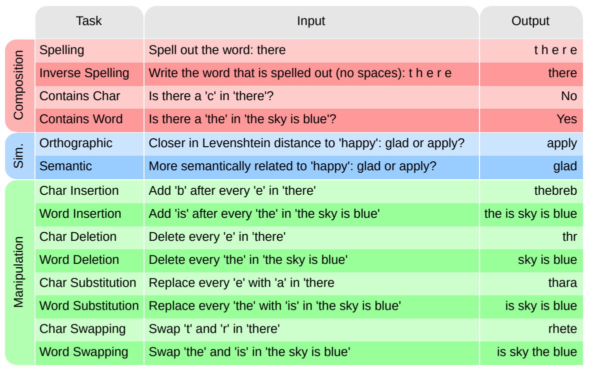
[1]We release our benchmark open-source at: `https://github.com/Leukas/CUTE`.

a string of characters. Most of their experiments concern a training method they propose, but they also include results of GPT-3 (Brown et al., 2020) with few-shot prompting, finding that it performs well on spelling correction, but poorly on other tasks compared to their trained character-based models. Most of their tasks require semantic knowledge, which we wish to ablate in our benchmark.

Other benchmarks testing orthographic knowledge often focus on morphology, such as the SIGMORPHON inflection tasks (e.g. Goldman et al. (2023)). Inflection is fairly regular for most languages, and could be memorized by a language model. Given that many developers of LLMs do not disclose the sources of their pretraining data, it is possible that LLMs memorize these inflection patterns. Therefore, from this we cannot conclude that LLMs can inflect a new word given a new set of rules, and furthermore we cannot conclude that LLMs can apply an arbitrary manipulation to a sequence. Most LLMs are trained with performance on English in mind, and English being less morphological in nature makes inflection a non-ideal measure for a model's understanding of the composition of tokens.

The most similar benchmark to the one we propose is LMentry (Efrat et al., 2023). The purpose of LMentry is similar in that the goal is to test models on tasks that are trivial to humans. Some of the tasks included test orthography and are similar to our tasks, for example, they ask the model to write a word containing a letter, or ask to write the first or last letter of a given word. Our benchmark is distinguished from LMentry as most of our tasks explicitly require knowledge of *every* character in a word. Testing whether a model knows the first letter of a word is not sufficient for concluding that it understands orthography, as there may be more pressure to learn about the first letter of a word from pretraining and/or instruction tuning (e.g. via alliterations or acronyms). As such, the task to write a word containing a letter could be more trivially solved by writing a word that starts with said letter.

There is an extensive body of research on character-level models, where each character forms a token (Lee et al., 2017; Xue et al., 2022; Tay et al., 2022, *inter alia*). Several works compared these models to subword models (Libovický et al., 2022; Edman et al., 2022, 2024, *inter alia*), but they evaluated on tasks requiring additional training. We assume that character-based models would perform



	Task	Input	Output
Composition	Spelling	Spell out the word: there	t h e r e
	Inverse Spelling	Write the word that is spelled out (no spaces): t h e r e	there
	Contains Char	Is there a 'c' in 'there'?	No
	Contains Word	Is there a 'the' in 'the sky is blue'?	Yes
Sim.	Orthographic	Closer in Levenshtein distance to 'happy': glad or apply?	apply
	Semantic	More semantically related to 'happy': glad or apply?	glad
Manipulation	Char 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
	Char Deletion	Delete every 'e' in 'there'	thr
	Word Deletion	Delete every 'the' in 'the sky is blue'	sky is blue
	Char 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
	Char Swapping	Swap 't' and 'r' in 'there'	rhete
	Word Swapping	Swap 'the' and 'is' in 'the sky is blue'	is sky the blue

Figure 1: All of the tasks in CUTE.

well on our benchmark, but we cannot test it since none of these models have been instruction-tuned.

3 Benchmark

We split our tasks into 3 categories: understanding composition, understanding orthographic similarity, and ability to manipulate sequences. Figure 1 shows an example for each task.[2] Data gathering and processing details can be found in Appendix C. Our tasks are synthetically generated from existing corpora. There are non-synthetically generated datasets which partially test our research questions, but these datasets have external factors (e.g. domain and/or language in a translation dataset) that would likely obscure our findings. Some of these datasets also might have been leaked into the LLM's pretraining data and been memorized, resulting in an unrealistically good performance.

3.1 Composition

We start with a straightforward benchmark: spelling. Similar to Itzhak and Levy (2022), we include a task where the input is a word given as a single token[3], and the output is the same word with spaces in between, so that each character becomes a separate token. This is the most straightforward probe to see whether a model has knowledge of the characters forming the tokens. We also add the inverted task ("inverse spelling") to check if characters can also be mapped to tokens.

Another method for assessing a model's understanding of composition is to ask if a token contains a certain character. If a model managed the previous tasks, we would expect it to succeed here as

[2]The prompts shown here are not the full prompts. See Appendix B for more details.

[3]This is in the ideal case. We cannot guarantee every LLM tested uses only a single token for each input, but we minimize the chance of splitting by using frequent words in our task.

well. However, a model might not understand the relationship between spelling and membership of characters to a word, so we test this as well. We also test whether the LLMs are able to solve the corresponding word-level task (i.e. is a word in a sentence) to separate the model's general understanding of the task from its ability to solve the task at the character level.

3.2 Similarity

Since the introduction of word2vec (Mikolov et al., 2013), language models have typically been trained to predict words from their context. The resulting token embeddings mainly reflect the semantic and syntactic similarity of tokens. Our next tasks examine whether LLMs also comprehend orthographic similarity. We ask which one of two candidate words is orthographically (or semantically) more similar to a given word. Our candidate words are chosen to be relatively easy to distinguish for a human without explicit knowledge of how to measure orthographic or semantic similarity. For more details, see Appendix C.

3.3 Manipulation

Our previous tasks focus on the understanding of the model. Now, we turn our focus to acting on that understanding. The next tasks involve 4 types of manipulation of the input at the character or word level: Insertion, Deletion, Substitution, and Swapping. We consider these tasks as elementary tasks for modifying a text sequence.

Insertion First we test how well the model can insert an element X after every instance of some element Y in the sequence. Similar modifications occur when we replicate letters to emphasize a word (e.g. "Yay!" vs. "Yaaaaay!"), or when we add an adjective next to a noun.

Deletion Deletion requires the model to recognize an element and remove all instances of it. This can occur in natural language at the character level with inflection in languages (e.g. turning an English plural noun into singular), or removing adjectives from a sentence.

Substitution Substitution replaces all instances of an element in a sequence with another element. This can occur with spelling or vocabulary variations across dialects or related languages (e.g. "defense" vs. "defence", or "elevator" vs. "lift").

Swapping Swapping is a simplified case of reordering acting on two elements.[4] Though reordering is not very common in English, it features heavily in languages with free word order, such as Greek, where stressed words can be moved to the front of the sentence.

Model	Params (B)	Tokens (k)	Lang
Llama 2	7, 13, 70	32	EN
Gemma	7	256	EN
Mistral	7, 47	32	EN
Aya 23	8, 35	256	Multil.
Cmd-R(+)	35, 104	256	Multil.
DBRX	132	100	EN
Llama 3	8, 70	100	EN

Table 1: Models evaluated on our benchmark. We note that Gemma has a multilingual tokenizer, but English-centric training.

4 Experimental Setup

Models We use the models shown in Table 1.[5] We choose freely-available[6] LLMs largely based on their popularity, as we are unaware of LLMs that specifically address the problems raised. While there are many differences between the models, we highlight 3 that could possibly affect performance: parameter count, vocabulary size, and multilingual versus English-centric or English-only training.

Prompts We use a template inspired by Bsharat et al. (2023)'s few-shot template to prompt our models with 4 in-context examples. Further details and a sample prompt can be found in Appendix B.

Russian Experiments We additionally create CUTE-Rus, a Russian version of our benchmark. We discuss this fully in Appendix D. The trends largely follow what we see for English.

5 Results

Figure 2 shows the results for each model. The random baseline is 50% for the `contains` and `similarity` tasks, and 0% for all other tasks.

5.1 Composition

The models perform very well on the tasks `spelling` and `inverse spelling`, though `inverse spelling` appears slightly more difficult.

[4]Due to the poor performance on swapping, we leave out more complex forms of reordering, but these could be easily added in the future.

[5]We link all of the models in Appendix A.

[6]We define "freely available" as those not requiring payment for use, and with available information on the training process.

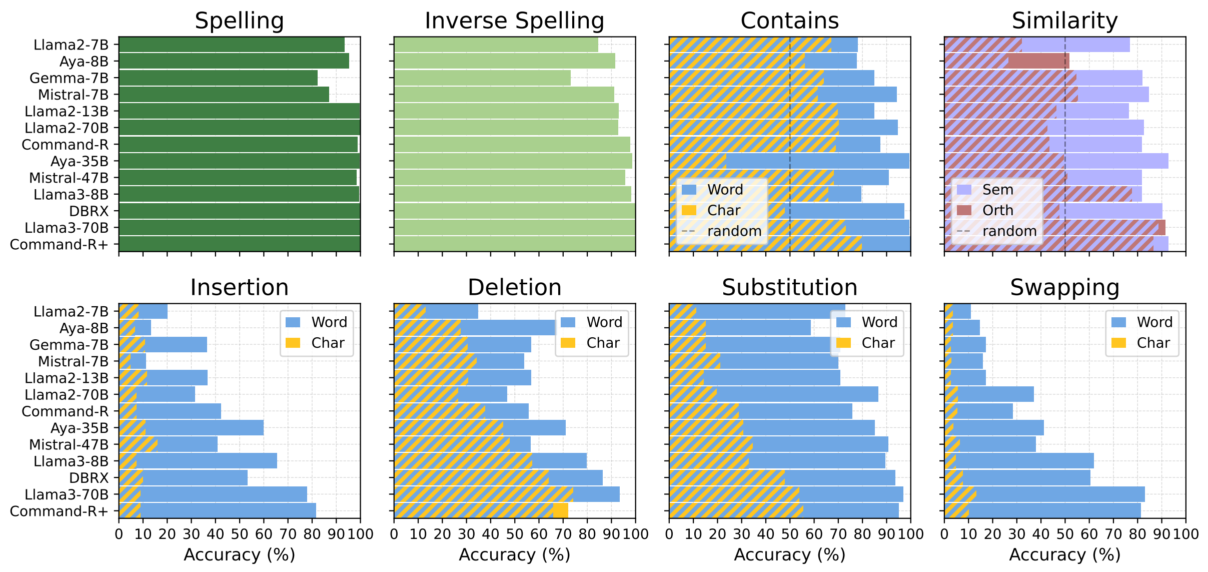


Figure 2: Accuracy on each task of CUTE. Models ordered by average accuracy over all tasks.

Although we are not aware of any spelling tasks in instruction tuning or pretraining datasets, we suspect that the similarity of this task with data seen during training allows the models to perform very well, in contrast to the following tasks.

On the `contains` tasks, the performance at the word level is quite good, showing that the models understand the task, but the performance breaks down at the character level. This indicates the models do not fully understand the relationship between spelling and membership of a character to a word.

5.2 Orthography and Semantic Similarity

In the `semantic similarity` task, the models correctly choose the more semantically related word 76-93% of the time (with the exception of Aya-8B), and the performance generally increases with model size. For `orthographic similarity`, the performance is below or near random for all models except Command-R+ and Llama3. It is not yet clear why they perform so well on this task, but apparently it is not solely a scaling effect since DBRX fails to perform above random chance. It may be due to the amount of training data, however DBRX and Command-R+ have not disclosed these amounts.

5.3 Manipulation

In our manipulation tasks, the models struggle more at the character level than at the word level. The difference is quite profound, with performance gaps of up to 72.8% on Command-R+ for `insertion`. Larger models such as Command-R+ perform well on deleting characters, with 72% accuracy, though it should be noted that we are only testing on the 1000 most frequent words, making the evaluation fairly generous.

Like the `contains` task and `orthographic similarity` task, the manipulation tasks show that LLMs lack a complete understanding of their tokens, although they can literally spell them out. The higher word-level performance indicates that it is not due to a lack of understanding of the task itself.

5.4 Vocabulary Size and Multilinguality

There appear to be no noticeable effects of vocabulary size from the results shown. Looking at the 7/8B models, while Llama 3 performs well with a vocabulary size of 100k, using a larger vocabulary (i.e. Gemma) does not improve performance, and neither does using a smaller vocabulary (i.e. Llama 2 and Mistral). It remains to be seen if noticeable effects arise as the vocabulary size approaches the number of characters. We leave this for future research.

For multilinguality, the results are similarly mixed. We focus on Aya-35B versus Command-R, as Aya is a fine-tuned version of Command-R, using multilingual instruction tuning data. Overall, Aya makes slight improvements over Command-R on the character-level, but this could easily be due to training on additional English data, rather than the additional non-English data.

5.5 Scaling

There are 2 major factors to scale: parameter count and training data. In terms of parameter count, larger models clearly tend to perform better. With respect to amount of training data, only Llama 2 and 3 have disclosed this, and based on the results, it appears that more training data also improves performance. This aligns well with the myriad of works showing the benefits of scaling and raises the question: Is scaling all we need for good performance on character-level tasks? Looking at the manipulation tasks, it seems that `deletion` and `substitution` could become manageable in the near future, but for `insertion` and `swapping`, the performance gap between word and character level tasks is large. Many real-world text manipulation tasks are a combination of the tested tasks, so we will likely need more than just scaling.

6 Conclusion

While current LLMs with BPE vocabularies lack direct access to a token's characters, they perform well on some tasks requiring this information, but perform poorly on others. The models seem to understand the composition of their tokens in direct probing, but mostly fail to understand the concept of orthographic similarity. Their performance on text manipulation tasks at the character level lags far behind their performance at the word level. LLM developers currently apply no methods which specifically address these issues (to our knowledge), and so we recommend more research to better master orthography. Character-level models are a promising direction. With instruction tuning, they might provide a solution to many of the shortcomings exposed by our CUTE benchmark.

7 Limitations

We prompt instruction-tuned LLMs without any fine-tuning on benchmark data. While this can be seen as a limitation, we note that it is not feasible to add more training data whenever we discover a new issue with LLMs. We expect that the performance of all models would increase after fine-tuning.

We do not evaluate any character-level models since there are no instruction-tuned versions (to our knowledge). Additionally, there are no decoder-only pretrained LLMs available, with the closest model being ByT5-XXL (13B), which has a heavy encoder and smaller decoder. Training character-level models also requires a much higher computational budget, as the sequence lengths are roughly 5 times longer, resulting in 5 times longer training. As such, training a truly comparable model falls outside the scope of this work.

Our benchmark does not control whether LLM tokenizers split words into multiple tokens. We minimize that chance by choosing frequent words, but we can never guarantee that future models will not split words into multiple tokens. We found that the impact of removing split tokens is minimal, with less than 1% change on average (see Appendix F).

We only test on English and Russian, with our primary focus being on English. While we did not see any major differences in the LLMs' performance between English and Russian when it comes to character-level versus word-level performance, it is possible that there may be differences in other languages.

Lastly, we do not control for generations that do not match the pattern of the examples given in the prompt. Therefore, we cannot guarantee that all generations considered correct by humans are evaluated as such. Hence the performance of some models may be lower than expected. We provide the outputs of all models in our repository for further analysis.

8 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). We also thank Lisa Bylinina for helping with the Russian translation of our benchmark.

References

Viraat Aryabumi, John Dang, Dwarak Talupuru, Saurabh Dash, David Cairuz, Hangyu Lin, Bharat Venkitesh, Madeline Smith, Jon Ander Campos, Yi Chern Tan, Kelly Marchisio, Max Bartolo, Sebastian Ruder, Acyr Locatelli, Julia Kreutzer, Nick Frosst, Aidan Gomez, Phil Blunsom, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. 2024. Aya 23: Open weight releases to further multilingual progress. *Preprint*, arXiv:2405.15032.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Sondos Mahmoud Bsharat, Aidar Myrzakhan, and Zhiqiang Shen. 2023. Principled instructions are all you need for questioning llama-1/2, gpt-3.5/4. *arXiv preprint arXiv:2312.16171*.

Lukas Edman, Gabriele Sarti, Antonio Toral, Gertjan van Noord, and Arianna Bisazza. 2024. Are character-level translations worth the wait? comparing byt5 and mt5 for machine translation. *Preprint*, arXiv:2302.14220.

Lukas Edman, Antonio Toral, and Gertjan van Noord. 2022. Subword-delimited downsampling for better character-level translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 981–992, Abu Dhabi, United Arab Emirates. 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.

Omer Goldman, Khuyagbaatar Batsuren, Salam Khalifa, Aryaman Arora, Garrett Nicolai, Reut Tsarfaty, and Ekaterina Vylomova. 2023. SIGMORPHON–UniMorph 2023 shared task 0: Typologically diverse morphological inflection. In *Proceedings of the 20th SIGMORPHON workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 117–125, Toronto, Canada. Association for Computational Linguistics.

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, 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.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

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.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Jindřich Libovický, Helmut Schmid, and Alexander Fraser. 2022. Why don't people use character-level machine translation? In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2470–2485, Dublin, Ireland. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay

Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

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.

A All Models

Here we link to all of the models. Those associated with published works are as follows: Touvron et al. (2023); Jiang et al. (2023, 2024); Team et al. (2024); Aryabumi et al. (2024). All models can be found at the following links:

- `https://hf.co/meta-llama/Llama-2-7b-chat-hf`
- `https://hf.co/meta-llama/Llama-2-13b-chat-hf`
- `https://hf.co/meta-llama/Llama-2-70b-chat-hf`
- `https://hf.co/google/gemma-7b-it`
- `https://hf.co/mistralai/Mistral-7B-Instruct-v0.2`
- `https://hf.co/mistralai/Mixtral-8x7B-Instruct-v0.1`
- `https://hf.co/CohereForAI/aya-23-8B`
- `https://hf.co/CohereForAI/aya-23-35B`
- `https://hf.co/CohereForAI/c4ai-command-r-v01`
- `https://hf.co/CohereForAI/c4ai-command-r-plus`
- `https://hf.co/databricks/dbrx-instruct`
- `https://hf.co/meta-llama/Meta-Llama-3-8B`
- `https://hf.co/meta-llama/Meta-Llama-3-70B`

B Prompting Details

We show an example of a full prompt in Figure 3. All of our prompts are available with the release of our benchmark. For generation, we use greedy search.

For evaluation of the generation, we rely on the given start-quote to denote the start of the answer, and we filter out anything after the end quote (e.g. "I hope this answer helped!"). Some models also were prone to starting generation with a generic response such as "Sure I can do that for you.". For these generations, we observe that it would repeat "Answer: ", so we filter out all generations before this point. Of the remaining generations, some could be considered correct though did not match the desired pattern (e.g. "H-E-L-L-O" rather than "h e l l o" for the spelling task). These we ultimately

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

1. Spell out the word “alphabet”.
Answer: “a l p h a b e t”
2. Spell out the word “hello”.
Answer: “h e l l o”
3. Spell out the word “zebra”.
Answer: “z e b r a”
4. Spell out the word “tongue”.
Answer: “t o n g u e”

Question: Spell out the word “cow”.
[/INST]
Answer: “
```

Figure 3: An example of a full prompt to spell the word “cow”, with examples, for the task `spelling`.

consider incorrect so as not to unfairly elevate any model’s performance.

Concerning the wording of the orthographic similarity task, it could be argued that models do not understand the concept of Levenshtein distance, and thus it is not comparable to the semantic similarity task. We also tested using “closer in edit distance” and “closer in spelling” in the prompt, and the results were very similar, so we opted for Levenshtein distance as it is more well-defined. Similarly, we used “closer in meaning” rather than “more semantically related” and achieved similar results, though it is debatable whether an antonym should be considered close in meaning, so we opted for the latter.

C Data Processing

Here we detail our exact method for gathering and processing the data into our tasks. The scripts for processing the data and the resulting data can be found at our Github repository.[7]

Data Sources For almost all tasks, we require a set of frequent English words that were most likely to be tokenized into a single token. For this, we use a dataset derived from the Google Web Trillion Word Corpus.[8]

For the word-based tasks, we use the TinyStories (Eldan and Li, 2023) dataset, which consists of stories written by an LLM in a style appropriate for a 3-4 year old reader. This has the benefit of using simple sentences with a limited vocabulary, maximizing the chances of words being tokenized into a single token in the models we test, while also ensuring that the complexity of the sentence is not a confounding factor in a model’s performance on the tasks.

Filtering For character-based tasks, we select the 1000 most frequent words that are at least 3 characters long. For word-based tasks, we similarly filter for 1000 sentences of length 3-10 words, in order to make the length similar to the number of characters in a word seen in the character-level tasks.

For insertion, deletion, and substitution, we apply the modification to those 1000 words, resulting in our dataset.

For swapping, we need to sure that the word or sentence has 2 items that are unique, so as to avoid an ambiguous prompt (e.g. swap the ‘e’ and ‘g’ in ‘engineering’). As such, we select the 1000 most frequent words or the first 1000 sentences that satisfy this criteria, as well as satisfying our length constraints.

Similarity Data For our similarity data, we require our candidate pairs to be sufficiently easy for a human to distinguish which is closer orthographically and which is closer semantically.

To accomplish this, our candidate words must satisfy two thresholds, one based on normalized Levenshtein distance (for othographic similarity), and one based on cosine similarity to other fastText (Bojanowski et al., 2017) embeddings (for semantic similarity). That is to say, the word must be sufficiently similar in one metric (0.7+ and 0.5+ for Levenshtein and cosine, respectively) and sufficiently dissimilar in the other (0.3− and 0.2−, respectively). These thresholds are decided empirically. We note that this process occasionally ends up with semantic pairs that are antonyms (e.g. “good” and “bad”), and thus we refrain from stating that the pairs are similar in meaning.

D Russian Experiments

Here, we detail the creation and evaluation on our Russian version of CUTE, CUTE-Rus.

[7]https://github.com/Leukas/CUTE
[8]https://www.kaggle.com/datasets/rtatman/english-word-frequency/data

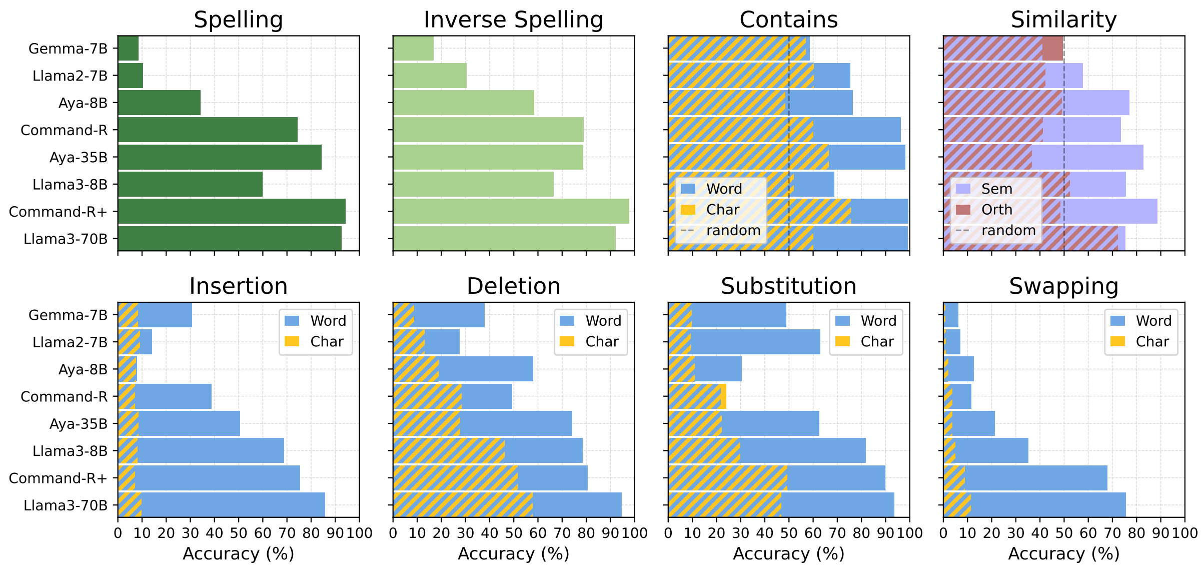


Figure 4: Accuracy on each task of CUTE-Rus. Models ordered by average accuracy over all tasks.

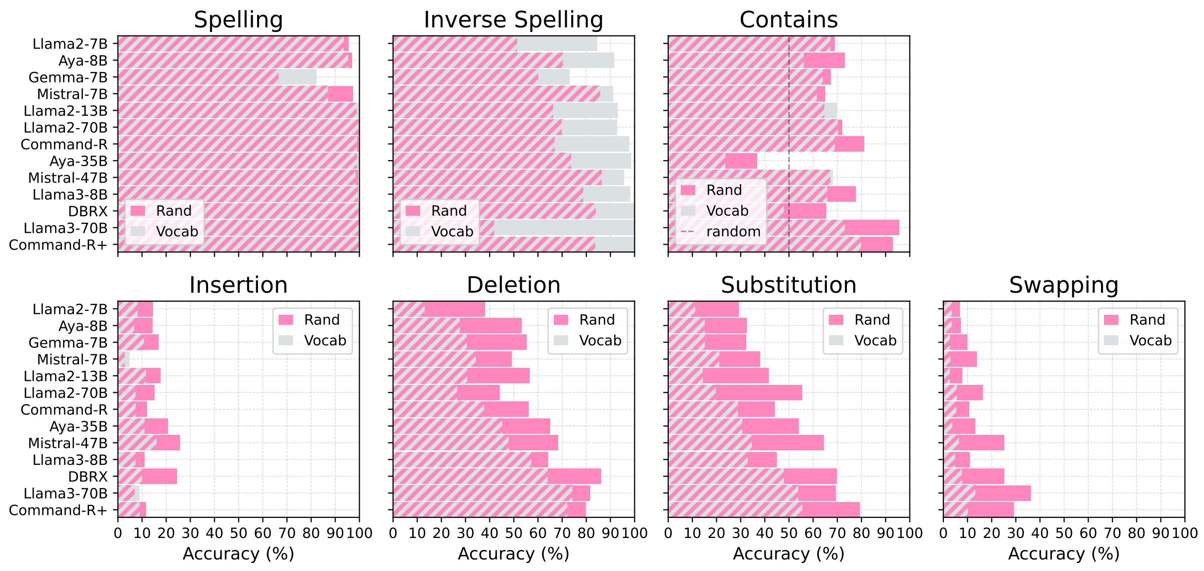


Figure 5: Evaluation of the performance on random strings versus strings in the vocabulary. "Vocab" is equivalent to the character-level tasks from Figure 2.

Data Processing Our data processing followed our processing in Appendix C. We collected a frequency list of Russian words from Wiktionary[9], embeddings from FastText, and we translated the English TinyStories dataset using Google Translate. We adjusted the thresholds for gathering our orthographic and semantic similarity pairs to (0.55+, 0.55+) and (0.3-, 0.1-) for Levenshtein and cosine, respectively. The thresholds were more difficult to adjust without excluding too many pairs, and as a result, some of the triplets could be argued as unclear or incorrect. For example, for the triplet (общий, община, совместный) meaning (*common*, *community*, *joint*), it is less clear which of the last two are more semantically related to the first (more so in Russian), though *joint* is the intended semantically-related word.

The prompts were also machine translated with Google Translate and post-edited by a native Russian and fluent English speaker, but after testing with both English and Russian prompts, we found the models generally performed better with English prompts (with Russian examples in the prompt), so we only include those in our results.

[9]https://w.wiki/AupG

Results We test a subset of the models used in the English version, focusing mainly on the multilingual LLMs (Aya, Command-R(+), and Gemma to the extent of the tokenizer), as well as Llama 2 and 3 for reference.

Figure 4 shows a similar trend to the English results. Generally speaking, the character-level performance lags behind the word-level. An interesting difference is that the models struggle much more with spelling. Even though the prompt is in English and shows examples of Russian words being spelled out, this is not enough for most models to understand the concept of spelling. The additional multilingual instruction tuning done for training Aya appears to be necessary for better performance.

E Random String Evaluation

While we cannot directly assess the performance of character-level LLMs without training an equivalent model from scratch, we can evaluate the performance of the existing models when the number of tokens per word approaches the number of characters per word. We can do this by conducting our tasks using random strings of consonants rather than complete words.

Since practically every word in English requires a vowel, random sequences of consonants are typically quite rare, and thus BPE will not dedicate a singular token for sequences like "fxqg". As such, we generate random strings for use in our tasks (excepting the similarity tasks). The resulting strings use on average 1.6 characters per token, compared to 5.4 characters per token in the original word list.

In Figure 5, we can see the performance of the LLMs on the character level tasks using regular words (as shown before in Figure 2), as well as random strings. We can see that, apart from `inverse spelling`, the models perform the same or better on random strings than on actual words. These tasks appear much easier for LLMs to handle when their tokenization is close to character-level, suggesting that a truly character-level LLM would perform the best.

As for `inverse spelling`, the decrease may be a result of the models' bias towards generating words. Upon inspection of the outputs, we observe that occasionally the model would hallucinate a word, changing a string such as "c q n r w" to "conquer", essentially filling in what it considers to be the missing vowels. This phenomenon is particularly common in Llama3-70B, whose performance decreased the most.

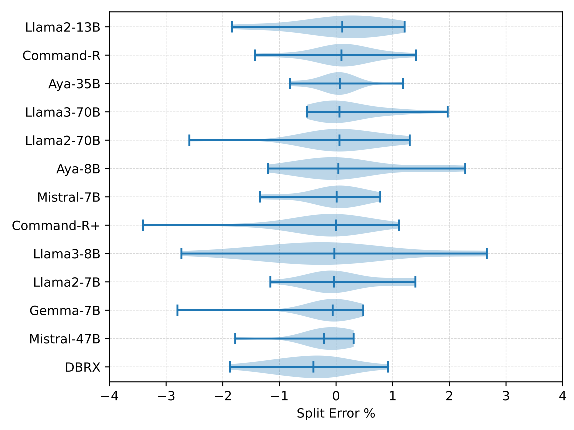


Figure 6: Distribution of change in accuracy for all tasks per model, sorted by median change.

F Token Splitting Impact

We mention that while we use frequent words for evaluation to maximize the chance they are given a single token, we cannot guarantee that some words will not be split into multiple tokens. Here, we evaluate the impact of this splitting on our results to elucidate whether this issue could affect the trends we see in the paper.

Figure 6 shows the percent change in accuracy were those examples to be removed from each task. All of the tasks are grouped into a violin plot. Here we can see that the maximum accuracy difference is around -3.5%, and the median errors for each model are no greater than $\pm 0.5\%$. This is far from substantially affecting the disparity we see between the character-level and word-level tasks. This also reveals that even if an LLM's tokenizer splits a word into two or more tokens, the LLM will still have difficulty performing the tasks in the CUTE benchmark.