

Broken Words, Broken Performance: Effect of Tokenization on Performance of LLMs

Sachin Pawar, Manoj Apte, Kshitij Jadhav, Girish K. Palshikar*, Nitin Ramrakhiani

TCS Research, Tata Consultancy Services Limited, India.

{sachin7.p, manoj.apte, kshitij.jadhav, nitin.ramrakhiani}@tcs.com, girishpalshikar@gmail.com

Abstract

Tokenization is the first step in training any Large Language Model (LLM), where the text is split into a sequence of tokens as per the model's fixed vocabulary. This tokenization in LLMs is different from the traditional tokenization in NLP where the text is split into a sequence of *natural* words. In LLMs, a natural word may also be broken into multiple tokens due to limited vocabulary size of the LLMs (e.g., Mistral's tokenizer splits *martial* into *mart* and *ial*). In this paper, we hypothesize that such breaking of natural words negatively impacts LLM performance on various NLP tasks. To quantify this effect, we propose a set of penalty functions that compute a tokenization penalty for a given text for a specific LLM, indicating how *bad* the tokenization is. We establish statistical significance of our hypothesis on multiple NLP tasks for a set of different LLMs.

1 Introduction

Recently, Large Language Models (LLMs) have been showing remarkable language understanding and generation capabilities. However, it is also observed that LLMs tend to produce inaccurate or unexpected results for some specific queries, the reasons for which can be traced back to the initial step of tokenization (Wang et al., 2024; Karpathy, 2024). Tokenization is the very first step in training any LLM where the text is split into a sequence of tokens as per the model's fixed vocabulary. This vocabulary is generally determined based on a different training corpus (and often much smaller) than the model's actual training corpus, using techniques such as Byte-Pair Encoding (BPE) (Sennrich et al., 2016). Due to this fixed and limited vocabulary of tokens, LLMs often break a *natural* word into multiple tokens. Here, by *natural* word, we mean the words obtained by traditional

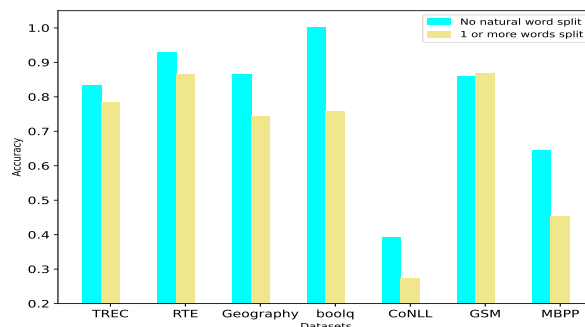


Figure 1: Effect of splitting words into multiple tokens

tokenization¹ in NLP. Nature of this tokenization significantly influences how the text is interpreted, processed and generated by the models.

Consider the word *unhappiness*. A human can break this long word into subwords intuitively like *un*, *happy* and *ness* which might lead to preserving the morphological structure. However, the tokenizer of an LLM (say Phi-3.5-mini-instruct) may break it as *unh*, *app*, *iness*. Since the model has learned the sequence of such tokens, the model's understanding of the query may shift subtly or drastically based on the specific tokens created by the tokenizer. When such shifts accumulate across an input they might create inconsistencies and in turn measurable change in the output of the model. For a variety of NLP tasks (Section 4.1), we observed that whenever there is no such breaking of any natural word in the input text, the model performance is generally better than the case when at least one word is split into multiple tokens (see Figure 1). We also observed a few examples where a small change in the input text to avoid such breaking of natural words (e.g., by using synonyms for such words) can lead to better performance. Table 1 shows one such example for the textual entailment task where the LLM output is corrected by a small change in the input.

*Work done while working at TCS Research

¹<https://spacy.io/api/tokenizer/>

P: *Ostriches* put their heads into the sand to avoid the wind.
H1: *Ostriches* **bury** their heads in the sand.
H2: *Ostriches* **hide** their heads in the sand.

Phi-3.5-mini-instruct predicts **No-Entailment** for (P, H1)
Phi-3.5-mini-instruct predicts **Entailment** for (P, H2)

Natural words which are split into multiple tokens:
Ostriches $\Rightarrow$ *_O, str, ich, es* ; *bury* $\Rightarrow$ *_b, ury*

Table 1: An example where the prediction is corrected with a small change (*bury* $\Rightarrow$ *hide*) in the input

Thus, one important question arises – how do small differences in tokenization lead to significant downstream effects on model behavior? In this paper, we attempt to answer this question by defining and quantifying “tokenization penalty” as the change in model performance or output quality that is attributable to tokenization induced distortions. One of the most relevant related work is by Land and Bartolo (2024) where they use token embeddings to find a set of *under-trained* tokens. One of our proposed penalty functions is designed based on their work. Another relevant work is by Wang et al. (2024) where they create an adversarial dataset for an LLM from the tokenization perspective. However, the constructed dataset covers only one kind of NLP task (sentence-based QA) and it contains artificially introduced errors (e.g., concatenating *moves* and *table* to *movestable*). On the other hand, we explore multiple types of NLP tasks (Section 4.1) and analyse the original input text as it is, to quantify tokenization related issues in it. There is another line of research (Chizhov et al., 2024; Lian et al., 2025b,a) which deals with improving the basic BPE tokenization algorithm. This is complimentary to our work in the sense that they attempt to improve tokenization quality by eliminating noisy tokens whereas we attempt to quantify the quality of tokenization (obtained by any tokenization algorithm) for a specific text.

Our specific contributions are:

- multiple *tokenization penalty* functions to quantify the effect of “bad” tokenization (Section 2)
- statistical significance tests to measure the effect of tokenization penalty on the performance of LLMs (Section 3)

2 Tokenization Penalty Functions

We propose 4 different penalty functions where a penalty is calculated for each natural word in the LLM input text. If any natural word corresponds to a single token as per the LLM’s tokenizer,

S1: *Bacteria* is winning the war against **antibiotics**.
S2: *It* is winning the war against **antibiotics**.

Natural words which are split into multiple tokens:
Bacteria $\Rightarrow$ *_B, acter, ia* ; *antibiotics* $\Rightarrow$ *_ant, ib, iot, ics*

Table 2: Tokenization example (Phi-3.5-mini-instruct)

there is no penalty considered. Here, we use the spaCy (Honnibal et al., 2020) tokenizer to get the list of tokens in the input text and each purely alphabetic token² is considered as a valid natural word. Consider a text T which is an input to LLM M . Let w be a valid natural word in T which is split into k tokens $t_{w_1} \cdots t_{w_k}$. Let the tokens to the left of t_{w_i} in T be $T_{<w_i}$. Let $\vec{t}$ be the vector representation of the token t as per M ’s output embedding matrix (following Land and Bartolo (2024)).

Penalty based on Token Anomaly Scores ($AS(w)$): The intuition behind this penalty function is – higher the anomaly score of any token, higher is the penalty. We use Isolation Forest (IF) (Liu et al., 2008) over all the tokens in M ’s vocabulary to compute anomaly score for each token and normalize the scores to lie in $[0, 1]$ (see F.1). Overall penalty for the word w is computed as:

$$AS(w) = \frac{1}{k} \sum_{i=1}^k AnomalyScore_{IF}(\vec{t}_{w_i}) \quad (1)$$

Penalty based on Similarity with Under-trained Tokens ($UT(w)$): It is motivated by Land and Bartolo (2024) that a token is *under-trained* if its embedding is closer to the average embeddings ($\vec{u}$) of unused tokens in the model’s vocabulary. So in this case the penalty for word w is:

$$UT(w) = \frac{1}{k} \sum_{i=1}^k (1 - CosineDistance(\vec{t}_{w_i}, \vec{u})) \quad (2)$$

Penalty based on Pairwise Distance between Tokens ($PD(w)$): The intuition behind this penalty function is – higher the distance between two consecutive tokens of w , higher is the penalty.

$$PD(w) = \frac{1}{k-1} \sum_{i=1}^{k-1} CosDist(\vec{t}_{w_i}, \vec{t}_{w_{i+1}}) \quad (3)$$

Contextual Penalty ($CP(w)$): All the earlier penalty functions are *non-contextual*, i.e., the penalty for the word w is independent of its context within T . We propose another penalty function

²Matching the regex $\wedge[A-Za-z]+\wedge$

which is *contextual*, i.e., the penalty for the same word w may vary if its context changes. Here, the intuition is that if the left context of w in T is such that the model M is less *perplexed* to see the tokens in w , then the tokenization penalty should be lower, and vice versa. We quantify this penalty using conditional probability of the tokens in w given their left context, as per the model M . E.g., in Table 2, the penalty for the word *antibiotics* would be much higher in sentence S2 as compared to S1 in which the context contains the word *Bacteria*. Also, part-of-speech (POS) tag of w may change based on its context in T . We hypothesize that certain POS tags (verbs, common nouns, adjectives and adverbs) contribute more to the overall meaning compared to certain other POS tags like proper nouns, prepositions, etc. Hence, tokenization penalty for w with POS tag $p(w)$ is multiplied by POS importance weight $wt_{p(w)}$ ³. So, the overall penalty⁴ for the word w is:

$$CP(w) = \frac{1}{k} \left(wt_{p(w)} \cdot \sum_{i=1}^k (1 - P_M(t_{w_i} | T_{<w_i})) \right) \quad (4)$$

Here, P_M denotes the next token probability as per model M and $p(w)$ denotes the POS tag of w . Appendix Table 6 shows various tokenization penalty functions computed for two example sentences.

Aggregation Techniques: All the above functions compute the tokenization penalty for a single word w appearing in text T . We explore 4 different aggregation techniques to compute the tokenization penalty for T aggregating over penalties for all the words in it –

- i **sum**: addition of all the word penalties
- ii **mean**: average of all the word penalties
- iii **max**: maximum among all the word penalties
- iv **top_K**: average of the top K word penalties

3 Measuring Effect of Bad Tokenization

To quantify the effect of “bad” tokenization on the performance of LLMs, we perform a statistical test. Consider an NLP task (e.g., classification, NER) and a corresponding dataset D with n instances. For each instance in D , we compute its tokenization penalty. We then use an LLM to generate the

³We use $wt_{p(w)} = 2$ for verbs, common nouns, adjectives, adverbs and $wt_{p(w)} = 1$ for other POS tags.

⁴Though Eq. 4 looks similar to *perplexity*, there is a key difference - unlike *perplexity* which considers all the tokens in a text, CP considers only those tokens which are part of some split natural word. See Appendix E for more details.

output for each of these instances⁵ and prepare two sets of tokenization penalties – (i) C (tokenization penalties for instances where the LLM produced *correct* output) and (ii) I (tokenization penalties for instances where the LLM produced *incorrect* output). We then use a one-sided two-sample **Student’s t -test** with the null and alternate hypotheses:

H_0 $Mean(I) = Mean(C)$ (average tokenization penalties are same for both correct as well as incorrect instances)

H_1 $Mean(I) > Mean(C)$ (average tokenization penalties for incorrect instances are higher than the correct instances)

We also perform the **Mann–Whitney U test** in a similar manner, as it is a non-parametric method that does not assume normality.

4 Experiments

We evaluate the proposed tokenization penalty functions on 7 different NLP tasks and 4 different LLMs – Phi (Abdin et al., 2024), Mistral (Jiang et al., 2023), Qwen (Qwen et al., 2025), and Llama (Grattafiori et al., 2024) having varying vocabulary sizes (see Figure 3).

4.1 NLP Tasks and Datasets

We consider a variety of NLP tasks and choose one representative dataset from each (Table 4). For all the tasks, tokenization penalties are computed for only the input text (and not for the other details in the prompts like instructions and few-shot examples which are common for all the instances).

Text Classification: TREC (Voorhees and Tice, 2000)
Text Pair Classification: RTE (Wang et al., 2019)
Passage-based QA: boolq (Clark et al., 2019)
Factual QA: Geography (Ramrakhiani et al., 2025)
NER: CoNLL 2003 (Sang and De Meulder, 2003)
Math Reasoning: GSM (Cobbe et al., 2021)
Code Generation: MBPP (Austin et al., 2021)

Table 4: NLP Tasks and corresponding datasets (more implementation details in Appendix A and B)

4.2 Experimental Results

Baselines: We consider two simple baselines for tokenization penalty functions – (i) **B1**: computes total number of tokens in the input, and (ii) **B2**: computes total number of natural words in the input that are split into multiple tokens.

⁵Appendix Table 5 shows the detailed prompts.

Dataset	Model	Acc	B1	B2	AS		UT		PD		CP			
					sum	max	sum	max	sum	max	sum	avg	max	top3
TREC	Phi	.796	.005	.100	.097	.098	.144	.154	.084	.083	.068	.746	.040	.043
	Mistral	.710	.258	.123	.055	.007	.120	.033	.136	.037	.110	.040	.033	.028
	Qwen	.776	.182	.096	.112	.091	.098	.083	.097	.077	.037	.019	.039	.033
	Llama	.754	.276	.184	.162	.133	.198	.173	.177	.143	.079	.026	.076	.069
RTE	Phi	.865	.490	.225	.264	.569	.217	.147	.237	.250	.109	.150	.263	.242
	Mistral	.818	.933	.987	.989	.733	.982	.395	.987	.514	.830	.632	.560	.459
	Qwen	.895	.871	.933	.924	.528	.949	.586	.938	.576	.781	.542	.670	.574
	Llama	.748	.999	.999	.999	.974	.999	.968	.999	.966	.971	.291	.639	.469
boolq	Phi	.757	.155	.178	.182	.009	.260	.810	.174	.089	.011	.001	.000	.000
	Mistral	.755	.271	.126	.098	.009	.148	.262	.116	.074	.045	.006	.015	.012
	Qwen	.773	.571	.601	.604	.656	.621	.827	.596	.737	.059	.037	.009	.018
	Llama	.798	.206	.370	.310	.456	.433	.844	.363	.671	.107	.122	.018	.012
Geography	Phi	.751	.082	.015	.026	.000	.136	.007	.024	.000	.000	.000	.000	.000
	Mistral	.790	.258	.000	.000	.001	.013	.017	.000	.000	.000	.000	.000	.000
	Qwen	.726	.342	.001	.001	.000	.003	.000	.001	.000	.000	.000	.000	.000
	Llama	.699	.554	.998	.998	.888	.997	.840	.997	.775	.756	.645	.207	.078
CoNLL	Phi	.280	.000	.000	.000	.001	.000	.000	.000	.005	.000	.336	.001	.012
	Mistral	.336	.000	.000	.000	.000	.000	.000	.000	.000	.000	.358	.000	.000
	Qwen	.481	.000	.000	.000	.000	.000	.000	.000	.000	.000	.114	.000	.030
	Llama	.389	.000	.000	.000	.000	.000	.000	.000	.000	.000	.312	.000	.011
GSM	Phi	.867	.000	.069	.092	.464	.075	.265	.061	.457	.009	.666	.205	.040
	Mistral	.640	.000	.047	.084	.512	.065	.382	.044	.303	.119	.998	.434	.265
	Qwen	.916	.000	.690	.753	.509	.694	.374	.686	.387	.426	.929	.493	.230
	Llama	.813	.000	.156	.176	.172	.177	.214	.154	.154	.134	.954	.198	.116
MBPP	Phi	.553	.131	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.002
	Mistral	.372	.003	.000	.000	.001	.000	.000	.000	.001	.004	.015	.015	.037
	Qwen	.595	.014	.046	.049	.065	.044	.055	.043	.053	.116	.114	.134	.150
	Llama	.532	.016	.002	.002	.018	.002	.019	.002	.017	.010	.022	.041	.061
#settings@5% significance			12	12	11	13	10	11	12	11	14	12	17	17
#settings@10% significance			13	14	16	16	13	13	15	16	17	12	18	20

Table 3: Statistical significance results (p-values) for various Dataset-LLM settings using **Student’s t-test**. The settings with significance at 5% & 10% are shown in **green** and **orange**, respectively. (Penalty functions – **B1**: No. of tokens, **B2**: No. of natural words that are split (baselines); **AS/UT/PD**: Penalties based on token anomaly scores, distance from unused tokens, and pairwise distance between tokens, respectively; **CP**: Contextual penalty)

Analysis of Results: Table 3 reports the results of the Student’s t-test⁶ for multiple Dataset-LLM combinations for the proposed tokenization penalty functions with various aggregation techniques. For the non-contextual functions, we report only *sum* and *max* due to space constraints. The contextual penalty (CP) function is observed to be the best as it achieves statistical significance at 5% for 17 out of 28 Dataset-LLM combinations. Also, there is agreement between both the statistical tests for most of the combinations. CP is better than other non-contextual functions (AS, UT, and PD) which are better than the baselines. We also carried out ablation study for the design decision of POS importance weight (wt_p) used in CP (Table 8). Except for RTE and GSM, on all other datasets, CP shows statistically significant effect on the accuracy for

most LLMs considered. We leave further investigation for RTE and GSM (and necessary changes to the tokenization penalty functions) as a future work. Figure 2 shows another interesting analysis where we observe significant difference between accuracy among the top and bottom deciles (most and least challenging instances w.r.t. tokenization) as per CP with *top_3* aggregation. This difference is observed across all datasets, even for RTE and GSM. We also observe that the smaller the vocabulary size of an LLM (i.e. higher its tokenizer fertility (Ali et al., 2024)), the greater the number of datasets on which tokenization significantly affects the LLM’s performance, as shown in Figure 3.

Transformation Strategies: The primary objective of this paper was to establish the statistical significance of the impact of “bad” tokenization on model performance. However, it is equally impor-

⁶See Appendix Table 7 for Mann-Whitney U Test.

tant to explore input transformation strategies to mitigate the issue of bad tokenization. As part of our future work, we plan to investigate several such strategies (illustrated here using examples from Mistral-7B-Instruct_v0.3):

- **Case modification.** For example, replacing *hollywood* (tokenized as *hol*, *ly*, *wood*) with *Hollywood* which is a single token.
- **Synonym substitution.** For example, *unexceptional* (split as *une*, *x*, *ception*, *al*) could be replaced with *ordinary* which is single token.
- **Morphology-aware tokenization.** For example, *genders* is split as *g* + *enders*; but a more linguistically informed tokenization would be *gender* + *s*, recognizing *s* as a common plural suffix.

5 Conclusions and Future Work

Due to the limited vocabulary size of LLMs, natural words are often split into multiple tokens. In this paper, we hypothesized that such splitting of natural words may adversely affect the performance of LLMs on various NLP tasks. To investigate this, we proposed a set of tokenization penalty functions which quantify how “bad” is the tokenization for a given text with respect to a specific LLM. We established the statistical significance of our hypothesis on seven different NLP tasks across four different LLMs. In future, we plan to explore a few simple transformations of input text (as discussed above) that reduce the tokenization penalty and potentially improve model performance. Additionally, we believe that insights from this analysis – both on tokenization issues and mitigation strategies – can help the design of more effective tokenizers for future LLMs.

Limitations

- **Tokenization not being the only issue:** Bad tokenization is not the only cause behind the unexpected performance of LLMs. It is just one of the many causes and that too a weak cause (e.g., there are errors even in instances where there is no tokenization penalty, as seen in Table 1). Hence, attributing some unexpected response by an LLM specifically to only bad tokenization issue is very challenging. What we have attempted to do in this work is only to establish a correlation between tokenization and performance of an LLM on multiple NLP tasks.

- **Number of models and tasks:** In this paper, we have experimented with only a limited number of different LLMs (4) and also a limited number of NLP tasks (7). Although, we have chosen 7 tasks of very different nature and considered 1 representative dataset for each task, there is still some scope of extending the experiments to cover more LLMs, more tasks, and more datasets per task.
- **Closed-source models:** We have not explored closed source models like OpenAI GPT because all our penalty functions need access to embedding matrix (for AS, UT, PD) as well as next token probability assigned to a certain token given its left context (for CP).
- **Combination of multiple penalty functions:** We have not yet explored how multiple penalty functions can be combined to produce a better combined penalty function. We plan to take this up as a future work.

References

- Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. [Phi-3 technical report: A highly capable language model locally on your phone](#). *Preprint*, arXiv:2404.14219.
- Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveiling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, and 1 others. 2024. Tokenizer choice for llm training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Pavel Chizhov, Catherine Arnett, Elizaveta Korotkova, and Ivan Yamshchikov. 2024. BPE gets picky: Efficient vocabulary refinement during tokenizer training. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16587–16604.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising

- difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, Adriane Boyd, and 1 others. 2020. spacy: Industrial-strength natural language processing in python.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. 2023. [Mistral 7b](#). *Preprint*, arXiv:2310.06825.
- Andrej Karpathy. 2024. [Let’s build the gpt tokenizer](#).
- Sander Land and Max Bartolo. 2024. Fishing for magikarp: Automatically detecting under-trained tokens in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11631–11646.
- Haoran Lian, Yizhe Xiong, Zijia Lin, Jianwei Niu, Shasha Mo, Hui Chen, Peng Liu, and Guiguang Ding. 2025a. LBPE: Long-token-first tokenization to improve large language models. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.
- Haoran Lian, Yizhe Xiong, Jianwei Niu, Shasha Mo, Zhenpeng Su, Zijia Lin, Hui Chen, Jungong Han, and Guiguang Ding. 2025b. Scaffold-BPE: Enhancing byte pair encoding for large language models with simple and effective scaffold token removal. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24539–24548.
- Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE.
- Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Ma Jie, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, Stefano Soatto, and 1 others. 2021. Structured prediction as translation between augmented natural languages. In *International Conference on Learning Representations*, pages 1–26. International Conference on Learning Representations, ICLR.
- 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, and 25 others. 2025. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Nitin Ramrakhiani, Vasudeva Varma, Girish Keshav Palshikar, and Sachin Pawar. 2025. Gauging, enriching and applying geography knowledge in pre-trained language models. *Information Processing & Management*, 62(1):103892.
- Matthew Renze and Erhan Guven. 2024. The effect of sampling temperature on problem solving in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7346–7356.
- Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.
- Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019*.
- Dixuan Wang, Yanda Li, Junyuan Jiang, Zepeng Ding, Guochao Jiang, Jiaqing Liang, and Deqing Yang. 2024. Tokenization matters! degrading large language models through challenging their tokenization. *arXiv preprint arXiv:2405.17067*.

A NLP Tasks and Datasets

In this section, we provide a more detailed description of the NLP tasks and the corresponding datasets used in our experiments.

- **Text Classification:** The task is to assign a suitable label to a given input text. We use the test partition of TREC ([Voorhees and Tice, 2000](#)) dataset (#instances=500).

- **Text Pair Classification:** The task is to assign a suitable label to a given pair of two texts. We use the train partition of RTE (Wang et al., 2019) dataset (#instances=2490) where for a given pair of premise and hypothesis, the task is to identify whether the premise entails the hypothesis or not.
- **Passage-based QA:** The task is to answer a question based on a specific passage of information. We use the validation partition of boolq dataset (Clark et al., 2019) (#instances=3200).
- **Factual QA:** In this task, a factual question is to be answered without any reference passage and only based on the pre-training knowledge of an LLM. We use Geography dataset proposed by Ramrakhiyani et al. (2025) (#instances=2879).
- **Named Entity Recognition (NER):** The task is to identify named entities in a given text. We use the test partition of CoNLL 2003 (Sang and De Meulder, 2003) dataset (#instances=1525). We only consider 3 entity types (PER, ORG, and LOC) and discard the sentences having no verb or less than 5 words to retain only meaningful sentences. Unlike all the other tasks where we use an LLM in zero-shot manner, for this NER task we use few-shot examples in TANL format (Paolini et al., 2021). The LLM output is considered to be *correct* if and only if all the entities in the input are identified correctly (both mention boundaries as well as types).
- **Math Reasoning:** The task is to solve a mathematical reasoning question using a multi-step approach. We use the test partition of the Grade School Math (GSM) dataset (Cobbe et al., 2021) (#instances=1319). We consider the solution to be *correct* if the final answer matches with that of the gold-standard answer, irrespective of the intermediate steps.
- **Code Generation:** The task to generate suitable python code for the given problem. We used the sanitized (hand-verified) version of the MBPP (Mostly Basic Python Problems) dataset (Austin et al., 2021) (#instances=427). Each instance in the dataset is also accompanied by 3 test cases. We consider the generated code to be *correct* if it passes these 3 test cases.

B Implementation Details

For obtaining natural words, we use spaCy tokenizer from the model en_core_web_trf-3.8.0.

We use the following 4 models for all our experiments having 8 billion or less parameters because of limited hardware available with us.

- **Phi:** Phi-3.5-mini-instruct⁷ (3.8 billion parameters)
- **Mistral:** Mistral-7B-Instruct-v0.3⁸ (7 billion parameters)
- **Qwen:** Qwen2.5-7B-Instruct⁹ (7 billion parameters)
- **Llama:** Meta-Llama-3-8B-Instruct¹⁰ (8 billion parameters)

Hyperparameters: We only used these LLMs in inference mode and uniformly used the temperature setting of 0 for all the 4 LLMs across all the 7 tasks to ensure more consistent and repeatable responses (based on the recommendation by Renze and Guven (2024)).

GPU size: All the experiments were run on an Nvidia Tesla V100 with 32GB GPU RAM.

Statistical tests: We used implementations of the Student’s t-test and Mann Whitney U test available as part of the stats package of scipy^{11 12}.

C Prompts used for the NLP tasks

Table 5 shows the detailed prompts used for each NLP task in our experiments. Zero-shot prompting is used for all the tasks except for NER where we use 8 few-shot examples.

D Examples of Penalty Functions

Table 6 shows 2 example sentences from the Geography dataset. For each example, it shows how each LLM tokenizes the input sentence (in terms of natural words that are split and also the values for each proposed penalty function.

⁷<https://huggingface.co/microsoft/Phi-3.5-mini-instruct>

⁸<https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3>

⁹<https://huggingface.co/Qwen/Qwen2.5-7B-Instruct>

¹⁰<https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct>

¹¹https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html

¹²<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mannwhitneyu.html>

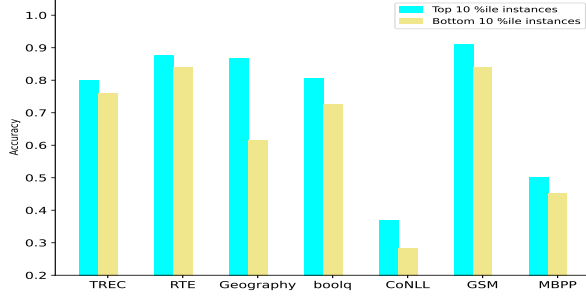


Figure 2: Accuracy difference between top and bottom deciles of the instances as per CP (top_3) for Phi

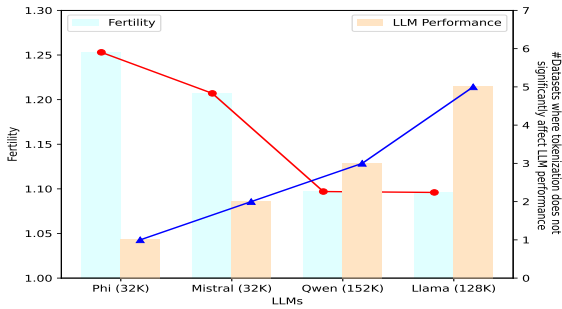


Figure 3: Comparing LLMs across multiple aspects: fertility (average number of tokens per natural word across all the datasets (Ali et al., 2024)) and performance (number of datasets where the LLM is not significantly affected by tokenization, as per CP (top3) penalty function). The numbers in bracket after each LLM indicate its vocabulary size.

E Comparing Contextual Penalty (CP) with Perplexity

Although, the formula for computing the tokenization penalty CP (Eq. 4) looks similar to *perplexity*, there are some key differences. Perplexity quantifies how much an LLM is *perplexed* (surprised) to observe a certain text whereas CP quantifies *goodness* of tokenization for a certain text for a specific LLM. Therefore, CP only considers those *natural words* which are split in the tokenization process and computes the penalty for each such word (which is further aggregated using some aggregation function to compute overall penalty for the text). Those natural words which are not split, contribute zero penalty as per CP . Moreover, CP also weighs importance of each natural word based on its POS tag. On the other hand, perplexity computation considers *all* tokens irrespective of whether a token corresponds to a natural word or just a part of it. It neither considers the concept of natural words nor POS tagging information. Overall, perplexity and CP both try to quantify how well an LLM can comprehend certain text but CP specifically focuses on the tokenization aspect. Similar to the tokenization penalty functions, perplexity may also affect the LLM’s performance on various tasks which we show in Table 9 using the similar statistical significance test.

The difference between CP and perplexity can also be seen in the results of the statistical test (Tables 3 and 9) where CP seems to have a significant effect on the performance for the TREC dataset but perplexity does not. On the other hand, for RTE-Llama combination, perplexity has a significant effect on performance but CP does not.

F Background concepts

F.1 Isolation Forest (IF) (Liu et al., 2008)

It is an anomaly detection technique based on the idea that anomalies are data points or instances that are few and different. It works by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of that feature. This process is repeated to build an ensemble of isolation trees. The anomaly score for each instance is computed as follows:

1. Each instance is passed through the trees, and the *path length* (number of splits required to isolate the instance) is recorded.

2. *Anomalies* tend to have shorter path lengths because they are easier to isolate.
3. The anomaly score for an instance is calculated based on the average path length across all trees. A higher score (closer to 1) indicates a higher likelihood of that instance being an anomaly.

In our case, each token is represented by a feature vector (as per the output embedding matrix of an LLM). Isolation Forest is then run over all the tokens in this space and anomaly score is computed for each token. The tokens with higher anomaly score are supposed to be more anomalous than other normal tokens with lower anomaly scores.

F.2 Byte Pair Encoding (BPE) (Sennrich et al., 2016)

It is a simple and effective text tokenization algorithm used for building the vocabulary for most LLMs. It compresses text by iteratively replacing the most frequent pair of adjacent symbols (characters or character sequences) with a new symbol. It starts with a sequence of characters, counts all adjacent symbol pairs, and then replaces the most frequent pair with a new token. This process is repeated for a fixed number of steps or until no more pairs are left. This results in a vocabulary of subword units that balances between characters and full words, making it especially useful for handling rare or unknown words in NLP models.

boolq dataset: Answer the Question below as Yes or No, based on the following Passage. Question: {question} Passage: {passage} Answer:
Geography dataset: Answer for the following question in one word. {question} Answer:
RTE dataset: The Entailment relation holds between a Premise and a Hypothesis when the Hypothesis can be inferred to be true if the Premise is true. For the following pair of PREMISE and HYPOTHESIS, identify whether Entailment relation holds or not. Do not generate any extra text. Only answer as "Yes" (if Entailment holds) or "No" (if Entailment does not hold). PREMISE: {premise} HYPOTHESIS: {hypothesis} Entailment:
TREC dataset: There are 6 different possible answer types for any Question as follows: 1. ABBR: an abbreviation 2. ENTY: an entity 3. DESC: a description or abstract concept 4. HUM: a human being 5. LOC: a geographical location 6. NUM: a numeric value What is the suitable answer type from the above list for the following Question? Question: {question} Answer type:
CoNLL dataset: You are given a SENTENCE, your task is to identify the entites ['person', 'organisation', 'location'] present in the SENTENCE. Avoid mentioning any other entites in the response. Only identify the entites provided in the list. Some examples of your task are given below- SENTENCE: Enn Markvart , chairman of the National Election Commission said 96 members of parliament cast votes . OUTPUT: [Enn Markvart person] , chairman of the [National Election Commission organization] said 96 members of parliament cast votes . SENTENCE: He did not elaborate . OUTPUT: He did not elaborate {6 more few-shot examples} Now, identify the entites from- SENTENCE: {sentence}
GSM dataset: Solve the following mathematical Question by using step-by-step reasoning. Write the final answer following the prefix "Final Answer:" Question: {question}
MBPP dataset: {problem} Use the following function name: {function_name} Generate only executable python code and nothing else.

Table 5: Prompts used for various NLP tasks

Sentence 1: <i>In which country, is the Raysko Praskalo waterfall located?</i>
Gold answer: <i>Bulgaria</i>
LLM: Phi, Generated answer: <i>Slovenia</i> , Split natural words: <i>_Ray, sko, _Pr, ask, alo, _water, fall</i> Tokenization Penalties: AS: 0.644, UT: 0.405, PD: 0.978, CP: 1.72
LLM: Mistral, Generated answer: <i>Russia</i> , Split natural words: <i>_R, ays, ko, _Pr, ask, alo, _water, fall</i> Tokenization Penalties: AS: 0.342, UT: 0.466, PD: 1.022, CP: 1.87
LLM: Qwen, Generated answer: <i>Bosnia and Herzegovina</i> , Split natural words: <i>_Rays, ko, _Pr, ask, alo</i> Tokenization Penalties: AS: 0.387, UT: 0.527, PD: 0.982, CP: 0.9999
LLM: Llama, Generated answer: <i>Serbia</i> , Split natural words: <i>_Rays, ko, _Pr, ask, alo</i> Tokenization Penalties: AS: 0.55, UT: 0.47, PD: 1.00, CP: 0.712
Sentence 2: <i>In which country, is the city of Helsinki located?</i>
Gold answer: <i>Finland</i>
LLM: Phi, Generated answer: <i>Finland</i> , Split natural words: <i>_Hels, ink, i</i> Tokenization Penalties: AS: 0.597, UT: 0.341, PD: 0.957, CP: 0.404
LLM: Mistral, Generated answer: <i>Finland</i> , Split natural words: <i>_H, els, ink, i</i> Tokenization Penalties: AS: 0.308, UT: 0.362, PD: 0.994, CP: 0.587
LLM: Qwen, Generated answer: <i>Finland</i> , Split natural words: NONE Tokenization Penalties: AS: 0, UT: 0, PD: 0, CP: 0
LLM: Llama, Generated answer: <i>Finland</i> , Split natural words: NONE Tokenization Penalties: AS: 0, UT: 0, PD: 0, CP: 0

Table 6: Example sentences from the Geography and their tokenization penalties using max aggregation. All 4 LLMs generate incorrect response for Sentence 1 whereas all the 4 LLMs generated a correct response for Sentence 2. Some key observations: (i) The penalty scores are NOT comparable across the penalty functions and also across the LLMs. (ii) The penalty scores for a specific penalty function and a specific LLM are comparable across multiple sentences (inputs). In this example, it can be observed that penalty for Sentence 2 is consistently lower than the penalty for Sentence 1 for each penalty function and for each LLM. (iii) As Qwen has much larger vocabulary size as compared to Phi and Mistral, it tends to produce lesser number of tokens.

Dataset	Model	Acc	B1	B2	AS		UT		PD		CP			
					sum	max	sum	max	sum	max	sum	avg	max	top3
TREC	Phi	.796	.001	.082	.058	.160	.185	.260	.033	.019	.027	.344	.034	.038
	Mistral	.710	.290	.084	.004	.001	.141	.079	.087	.082	.037	.025	.039	.029
	Qwen	.776	.318	.081	.087	.096	.102	.095	.088	.082	.059	.048	.061	.058
	Llama	.754	.229	.161	.129	.122	.200	.186	.129	.120	.129	.069	.127	.119
RTE	Phi	.865	.328	.389	.433	.871	.383	.161	.412	.507	.184	.164	.321	.282
	Mistral	.818	.874	.976	.978	.922	.955	.346	.982	.530	.833	.659	.625	.409
	Qwen	.895	.584	.906	.914	.486	.915	.778	.932	.924	.754	.587	.750	.536
	Llama	.748	.999	.999	.999	.984	.999	.985	.999	.999	.962	.266	.567	.390
boolq	Phi	.757	.169	.166	.164	.019	.238	.966	.151	.366	.002	.000	.000	.000
	Mistral	.755	.296	.168	.149	.038	.193	.742	.151	.149	.026	.006	.065	.015
	Qwen	.773	.695	.606	.594	.529	.642	.946	.618	.971	.077	.048	.005	.009
	Llama	.798	.658	.540	.470	.226	.608	.949	.562	.897	.090	.073	.012	.008
Geography	Phi	.751	.400	.002	.020	.289	.100	.671	.006	.061	.000	.000	.000	.000
	Mistral	.790	.783	.000	.001	.175	.029	.725	.000	.000	.000	.000	.000	.000
	Qwen	.726	.930	.000	.000	.000	.006	.002	.000	.000	.000	.000	.000	.000
	Llama	.699	.978	.992	.997	.942	.977	.778	.857	.365	.425	.664	.025	.002
CoNLL	Phi	.280	.000	.000	.000	.003	.000	.000	.000	.053	.000	.363	.000	.013
	Mistral	.336	.000	.000	.000	.000	.000	.000	.000	.000	.000	.424	.000	.000
	Qwen	.481	.000	.000	.000	.000	.000	.000	.000	.000	.000	.067	.000	.013
	Llama	.389	.000	.000	.000	.000	.000	.000	.000	.000	.000	.095	.003	.014
GSM	Phi	.867	.000	.075	.080	.509	.066	.192	.076	.089	.004	.677	.060	.011
	Mistral	.640	.000	.117	.181	.848	.147	.554	.097	.274	.201	.996	.539	.269
	Qwen	.916	.000	.625	.703	.723	.604	.378	.682	.635	.427	.830	.352	.247
	Llama	.813	.000	.205	.249	.185	.246	.295	.164	.070	.121	.771	.137	.097
MBPP	Phi	.553	.270	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.001
	Mistral	.372	.007	.000	.000	.003	.000	.000	.000	.003	.002	.006	.021	.046
	Qwen	.595	.117	.054	.055	.064	.060	.057	.039	.043	.056	.073	.058	.061
	Llama	.532	.062	.014	.013	.019	.014	.023	.009	.011	.018	.019	.026	.031
#settings@5% significance			10	10	11	11	9	8	12	10	15	11	16	18
#settings@10% significance			11	15	15	13	12	11	16	16	19	16	20	21

Table 7: Statistical significance results (p-values) for various Dataset-LLM settings using **Mann-Whitney U Test**. The settings with significance at 5% & 10% are shown in **green** and **orange**, respectively. (Penalty functions – B1: Baseline of no. of tokens, B2: Baseline of no. of natural words that are split, AS: Penalty based on token anomaly scores, UT: Penalty based on distance from unused tokens, PD: Penalty based on pairwise distance between tokens, CP: Contextual penalty)

Dataset	Model	Acc	CP (with POS multiplier)				CP (without POS multiplier)			
			sum	avg	max	top3	sum	avg	max	top3
TREC	Phi	.796	.068	.746	.040	.043	.060	.593	.037	.065
	Mistral	.710	.110	.040	.033	.028	.083	.020	.026	.027
	Qwen	.776	.037	.019	.039	.033	.024	.009	.025	.021
	Llama	.754	.079	.026	.076	.069	.109	.019	.101	.094
RTE	Phi	.865	.109	.150	.263	.242	.102	.140	.266	.290
	Mistral	.818	.830	.632	.560	.459	.830	.644	.428	.389
	Qwen	.895	.781	.542	.670	.574	.762	.503	.585	.552
	Llama	.748	.971	.291	.639	.469	.995	.462	.899	.774
boolq	Phi	.757	.011	.001	.000	.000	.008	.000	.003	.000
	Mistral	.755	.045	.006	.015	.012	.035	.003	.011	.008
	Qwen	.773	.059	.037	.009	.018	.137	.070	.084	.178
	Llama	.798	.107	.122	.018	.012	.050	.062	.114	.057
Geography	Phi	.751	.000	.000	.000	.000	.000	.000	.000	.000
	Mistral	.790	.000	.000	.000	.000	.000	.000	.000	.000
	Qwen	.726	.000	.000	.000	.000	.000	.000	.000	.000
	Llama	.699	.756	.645	.207	.078	.734	.607	.186	.077
CoNLL	Phi	.280	.000	.336	.001	.012	.000	.454	.265	.464
	Mistral	.336	.000	.358	.000	.000	.000	.515	.000	.000
	Qwen	.481	.000	.114	.000	.030	.000	.116	.012	.158
	Llama	.389	.000	.312	.000	.011	.000	.243	.000	.005
GSM	Phi	.867	.009	.666	.205	.040	.006	.590	.211	.056
	Mistral	.640	.119	.998	.434	.265	.039	.991	.191	.095
	Qwen	.916	.426	.929	.493	.230	.491	.937	.504	.283
	Llama	.813	.134	.954	.198	.116	.099	.970	.122	.076
MBPP	Phi	.553	.000	.000	.000	.002	.000	.000	.000	.001
	Mistral	.372	.004	.015	.015	.037	.002	.009	.010	.025
	Qwen	.595	.116	.114	.134	.150	.082	.080	.096	.106
	Llama	.532	.010	.022	.041	.061	.005	.011	.022	.037
#settings@5% significance			14	12	17	17	15	11	14	12
#settings@10% significance			17	12	18	20	20	14	16	19

Table 8: Ablation study for using POS importance weight (wt_p) in the Contextual Penalty (CP) function. Statistical significance results (p-values) for various Dataset-LLM settings using **Student’s t-test** are shown. The settings with significance at 5% & 10% are shown in **green** and **orange**, respectively.

Dataset	Model	Acc	Perplexity
TREC	Phi	.796	.161
	Mistral	.710	.767
	Qwen	.776	.118
	Llama	.754	.337
RTE	Phi	.865	.122
	Mistral	.818	.205
	Qwen	.895	.723
	Llama	.748	.001
boolq	Phi	.757	.000
	Mistral	.755	.007
	Qwen	.773	.009
	Llama	.798	.000
Geography	Phi	.751	.000
	Mistral	.790	.000
	Qwen	.726	.000
	Llama	.699	.000
CoNLL	Phi	.280	.635
	Mistral	.336	.999
	Qwen	.481	.877
	Llama	.389	.994
GSM	Phi	.867	.000
	Mistral	.640	.002
	Qwen	.916	.025
	Llama	.813	.257
MBPP	Phi	.553	.000
	Mistral	.372	.276
	Qwen	.595	.000
	Llama	.532	.000
#settings@5% significance			15
#settings@10% significance			15

Table 9: Statistical significance results (p-values) for various Dataset-LLM settings using **Student’s t-test** for Perplexity