ISSN: 3139-1141

A Comparative Study of Tokenization Efficiency in English, Hindi, Marathi and Romanized Hinglish Across OpenAI Tokenizer Generations

Samruddhi Hemant Belsare
Master of Computer Applications, Zeal Institute of Business Administration, Computer Application and Research (ZIBACAR)
Savitribai Phule Pune University, Pune, India
samruddhibelsare11@gmail.com

Dr. Madhavi Shamkuwar
ZIBACAR, Pune, India (Co-author)
madhavi.shamkuwar@zealeducation.com

Abstract—

Large language models read text as small pieces called tokens. The number of tokens decides how much text fits in a model's context window and how much a token-billed service charges. Recent studies show that Indian languages need many more tokens than English, and that Hinglish, which is Hindi written in English letters, creates its own tokenization problems. However, to our knowledge, Marathi, Hindi and several spellings of Hinglish have not been compared side by side as tokenizers have improved. We built 99 parallel sentences in English, Marathi, Hindi and three Hinglish spellings, and counted their tokens with four OpenAI tokenizer encodings, from older to newer. On the oldest encodings, Marathi and Hindi needed 7.4 and 7.6 times as many tokens as English. On the newest, o200k_base, they needed 1.80 and 1.54 times as many. Hinglish needed 1.8 to 2.2 times as many tokens on the oldest encodings and 1.5 to 1.9 times on the newest, so it was much cheaper than Hindi on older tokenizers but not on the newest. Different Hinglish spellings of the same sentence changed token counts by 26% to 31% on average. These results show that the cost of writing in Indian languages depends strongly on the tokenizer and on the script and spelling a user chooses. The data and code are available at https://github.com/samruddhibelsare-max/hinglsh_tokenization_study.

Keywords— **Tokenization, large language models, Marathi, Hindi, Hinglish, token inflation, token fertility, multilingual NLP, context window**

I. INTRODUCTION

Before a large language model (LLM) reads any text, a tokenizer cuts it into small pieces called tokens. A token can be a whole word, part of a word, or a single character. The token count matters for two practical reasons. A model can only read a limited number of tokens at a time, called its context window, so more tokens per sentence means less content fits. Many commercial services also charge per token, so more tokens means a higher bill for the same message.

A tokenizer learns its vocabulary from a body of text, and when most of that text is English, other languages are cut into more pieces. Ahia et al. [1] tested a commercial service on 22 languages and found that speakers of many of them pay more and receive worse results. Petrov et al. [2] reported token counts that differ by up to fifteen times across languages for the same content. Later work reports the same pattern: Teklehaymanot and Nejdl [3] measured more than 200 languages with tiktoken and found higher token costs for non-Latin scripts, and Srivastava [4] reported about eight times the English token cost for Indian languages under cl100k_base. Velayuthan and Sarveswaran [5] studied Tamil, Sinhala and Hindi and found that choices made before the subword step limit how well tokenizers handle complex scripts.

In India, many people also write Hindi in English letters, known as Hinglish, for example "mujhe nahi pata". There is no fixed spelling, so the same words may appear as "nahi", "nahin" or "nhi". Because a tokenizer splits text by letter patterns, different spellings can give different token counts. A.Deshmukh and S.Deshmukh [6] showed that a tokenizer built for Hinglish reduced token counts by about a quarter compared with the tokenizer of GPT-2.

Srivastava [4] covers Hindi and Marathi in their native script but not romanized text, and we did not find a study that directly compares Marathi, Hindi and several spellings of Hinglish against English, using the same parallel sentences, across tokenizer generations. This paper does that with four OpenAI encodings, from older to newer. We do not train any model or tokenizer. We measure what existing tokenizers do and express the difference as relative cost and context-window capacity. Our contributions are:

- A parallel set of 99 sentences in English, Marathi, Hindi and three Hinglish spellings, covering news, conversation and technical text.
- A measurement of token inflation relative to English across four encoding generations, showing how the gap changed over time.

ISSN: 3139-1141

- A comparison of Hinglish with Devanagari Hindi, and a measurement of how much Hinglish spelling changes token counts.
- An estimate of the relative cost and context-window effect of these differences.

II. RELATED WORK

A. Cost and fairness across languages

Ahia et al. [1] linked token counts to benchmark results from a commercial service, while Petrov et al. [2] looked at tokenization before any model is run and also found large gaps for character-level and byte-level models. Teklehaymanot and Nejdl [3] used the same library as this study, tiktoken, and reported each language's cost relative to English. These studies measure many languages at once. Our study instead looks closely at Marathi, Hindi and romanized Hinglish.

B. Tokenizer quality

Rust et al. [7] compared multilingual models with dedicated monolingual models in nine languages and found that a dedicated tokenizer for a language matters about as much as the amount of pretraining data. This shows that tokenizer design affects results for a language, not just cost.

C. Indian languages

Dwivedi and Gopalan [8] evaluated tokenizers for Hindi and Marathi using intrinsic measures and downstream tasks, with attention to the Devanagari script. Velayuthan and Sarveswaran [5] studied Tamil, Sinhala and Hindi using compression and parity measures. Srivastava [4] compared six tokenizers on 997 FLORES-200 sentences in ten Indian languages, including Hindi and Marathi, and four comparison languages, and linked the high cost under cl100k_base to text being split into single bytes.

D. Hinglish and code-mixed text

Deshmukh and Deshmukh [6] studied tokenization of romanized Hindi-English text and compared a tokenizer trained on Hinglish with the tokenizer of GPT-2. Nayak and Joshi [9] built L3Cube-HingCorpus, a corpus of about 53 million Roman-script Hindi-English sentences from Twitter, and trained BERT-based models on it. Neither of these works compares Hinglish token counts directly with those of English, Hindi and Marathi.

E. Research gap

The closest work to ours is Srivastava [4], who measured token costs for Hindi, Marathi and other Indian languages written in their native scripts. That study did not include romanized text. Our study adds three things. First, we place three romanized Hinglish spellings alongside Marathi and Hindi in one parallel corpus. Second, we measure how much Hinglish spelling changes token counts. Third, we test whether Hinglish is cheaper than Hindi and how this changed across older and newer OpenAI encodings.

III. METHODOLOGY

A. Research Questions and Hypotheses

We ask four questions. RQ1: how much do Marathi, Hindi and Hinglish increase token counts compared with English? RQ2: how did this change from older to newer encodings? RQ3: how does Hinglish compare with Hindi, and how much does Hinglish spelling change token counts? RQ4: what are the relative cost and context-window effects? We test five hypotheses: H1, Marathi, Hindi and Hinglish need more tokens than English in every encoding; H2, Marathi inflation is at least as high as Hindi; H3, Hinglish inflation is lower than Hindi; H4, newer encodings reduce inflation; H5, Hinglish spelling changes token counts by more than 10% on average.

B. Corpus

The corpus has 99 parallel sentence sets, each with the same meaning in English, Marathi, Hindi and three romanized Hinglish spellings. There are 33 sentences each from news, everyday conversation and technical text. By English word count, 50 sentences are short (8 words or fewer), 28 medium (9 to 16) and 21 long (17 or more). Hinglish versions were written in different common romanization styles to capture spelling variation; the second style tends to use more English words and abbreviations. The sentence sets, including the Marathi, Hindi and Hinglish versions, were first drafted with an AI language model (Claude, Anthropic).

C. Tokenizers

We used four encodings from OpenAI's tiktoken library [10], listed from oldest to newest: r50k_base, p50k_base, cl100k_base and o200k_base. Special tokens were not added. Tokenization ran on Google Colab with tiktoken version 0.14.0 on 24 October 2026.

D. Metrics

For each sentence we recorded the number of tokens, words and UTF-8 bytes. We define:

- *inflation = tokens(L) / tokens(English)* (1)
- *fertility = tokens / words* (2)
- *bytes per token = UTF-8 bytes / tokens* (3)

Inflation is calculated for each sentence against the English version of the same sentence in the same encoding, and then averaged. An inflation of 1 means the same number of tokens as English.

E. Statistical Analysis

We report means with 95% confidence intervals from 5,000 bootstrap resamples of the sentences. Paired comparisons use the two-sided Wilcoxon signed-rank test on matched sentences. We ran 54 comparisons and applied Bonferroni correction. Two comparisons (r50k_base against p50k_base) could not be tested because both encodings gave identical counts for every sentence.

ISSN: 3139-1141

IV. RESULTS

A. Token inflation across encodings

Table I and Fig. 1 show mean inflation relative to English. On r50k_base and p50k_base, which gave identical counts on our sentences, Marathi needed 7.38 times and Hindi 7.61 times as many tokens as English. This fell to 4.79 and 4.99 on cl100k_base and to 1.80 and 1.54 on o200k_base, a reduction of 76% for Marathi and 80% for Hindi. All languages differed significantly from English in every encoding (adjusted $p < 0.001$).

TABLE I
MEAN TOKEN INFLATION VS ENGLISH (95% CI)

Language	r50k / p50k	cl100k	o200k
Marathi	7.38 [7.04, 7.72]	4.79 [4.58, 4.99]	1.80 [1.72, 1.87]
Hindi	7.61 [7.29, 7.95]	4.99 [4.80, 5.18]	1.54 [1.48, 1.61]
Hinglish v1	2.14 [2.06, 2.22]	1.96 [1.88, 2.04]	1.77 [1.70, 1.84]
Hinglish v2	1.79 [1.71, 1.87]	1.67 [1.60, 1.74]	1.53 [1.46, 1.60]
Hinglish v3	2.22 [2.13, 2.31]	2.04 [1.96, 2.13]	1.85 [1.78, 1.92]

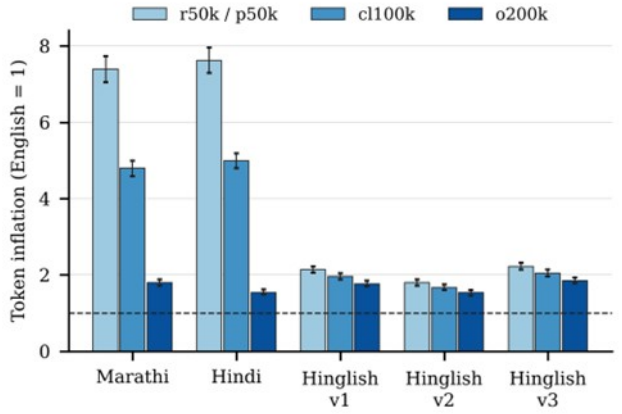


Fig. 1. Mean token inflation relative to English with 95% bootstrap confidence intervals. The dashed line marks English (1.0).

Hinglish changed much less across generations, from 2.14 to 1.77 for the first spelling (17%), 1.79 to 1.53 for the second (15%) and 2.22 to 1.85 for the third (17%). The gain from newer tokenizers was therefore concentrated in Devanagari text.

B. Fertility and bytes per token

Fertility followed the same pattern (Table II). Marathi needed 10.35 tokens per word on the oldest encodings and 2.50 on o200k_base. Bytes per token (Table III) explain why. A Devanagari character takes three bytes in UTF-8, and the oldest encodings averaged only 1.68 bytes per token for both Marathi and Hindi, so they were splitting characters into pieces. On o200k_base this rose to 6.96 for Marathi and 8.41 for Hindi. Hinglish stayed between 2.6 and 3.4 bytes per token throughout.

TABLE II
MEAN FERTILITY (TOKENS PER WORD)

Language	r50k / p50k	cl100k	o200k
Marathi	10.35	6.68	2.50
Hindi	7.92	5.19	1.61
Hinglish v1	2.23	2.04	1.83
Hinglish v2	1.88	1.75	1.59
Hinglish v3	2.30	2.11	1.91

TABLE III
MEAN BYTES PER TOKEN

Language	r50k / p50k	cl100k	o200k
Marathi	1.68	2.59	6.96
Hindi	1.68	2.56	8.41
Hinglish v1	2.63	2.89	3.22
Hinglish v2	2.92	3.12	3.41
Hinglish v3	2.56	2.80	3.09

C. Marathi, Hindi and Hinglish compared

Table IV compares languages directly. On the three older encodings, Marathi inflation was slightly lower than Hindi (for example 7.38 against 7.61), but the difference was not significant after correction. On o200k_base Marathi (1.80) was significantly higher than Hindi (1.54), because the newest encoding helped Hindi more than Marathi.

Hinglish was significantly cheaper than Hindi on all three older encodings: 1.7 to 2.2 times English against 5.0 to 7.6 for Hindi. On o200k_base the order changed. Hinglish v1 (1.77) and v3 (1.85) now needed significantly more tokens than Hindi (1.54), and v2 (1.53) did not differ from Hindi.

TABLE IV
ADJUSTED P-VALUES FOR SELECTED COMPARISONS

Comparison	r50k / p50k	cl100k	o200k
Marathi vs Hindi	1.00 (n.s.)	0.36 (n.s.)	<0.001 (Marathi higher)
Hinglish v1 vs Hindi	<0.001 (lower)	<0.001 (lower)	<0.001 (higher)
Hinglish v2 vs Hindi	<0.001 (lower)	<0.001 (lower)	1.00 (n.s.)
Hinglish v3 vs Hindi	<0.001 (lower)	<0.001 (lower)	<0.001 (higher)

D. Hinglish spelling variation

For the same sentence, the three Hinglish spellings differed in token count by 31.0% on average on r50k_base and p50k_base, 27.5% on cl100k_base and 25.9% on o200k_base, measured against the shortest spelling. All three pairs of spellings differed significantly in every encoding (adjusted $p < 0.001$). Spelling variation therefore changed Hinglish token counts by an amount comparable to or larger than the move from the oldest to the newest encoding did (15% to 17%), although it is much smaller than the effect on Hindi and Marathi. Because the second spelling also uses more English words, part of this difference reflects word choice and not spelling alone.

ISSN: 3139-1141

E. Relative cost and context capacity

When a service charges the same price per token, the cost of a message grows in proportion to its token count, so the inflation ratio is also the relative cost. On o200k_base a Marathi message costs about 1.8 times an English one and a Hindi message about 1.5 times, compared with about 7.4 and 7.6 times on r50k_base. Context capacity falls in the same proportion: a window that holds 100 English sentences holds roughly 56 Marathi or 65 Hindi sentences on o200k_base, and roughly 14 and 13 on r50k_base. These figures are approximate because they use mean ratios.

F. Summary of hypotheses

TABLE V
OUTCOME OF EACH HYPOTHESIS

Hypothesis	Outcome
H1 More tokens than English	Supported in all encodings
H2 Marathi ≥ Hindi	Supported on o200k_base only; no significant difference on older encodings
H3 Hinglish < Hindi	Supported on older encodings; not supported on o200k_base
H4 Newer encodings reduce inflation	Supported for Marathi and Hindi; small for Hinglish
H5 Spelling effect > 10%	Supported (25.9% to 31.0%)

V. DISCUSSION

The main finding is that the cost of writing in Devanagari fell sharply with the newest tokenizer. A likely reason is a larger vocabulary that includes more Devanagari pieces, but we did not examine the vocabularies, so we only report the measured effect. Because the older encodings split Devanagari into byte-level pieces, a Marathi or Hindi user of an older model paid several times what an English user paid for the same content. That gap has narrowed but not closed: Marathi still needs 1.8 times the tokens of English on o200k_base.

The sharp drop for Devanagari text agrees with Srivastava [4], who measured Hindi and Marathi on 997 FLORES-200 sentences. In that study Hindi fell from 4.08 to 1.34 times English and Marathi from 5.84 to 2.11 between cl100k_base and o200k_base, a reduction of about 67% and 64%. In our data the reductions were 69% for Hindi and 62% for Marathi. Both studies find Marathi higher than Hindi on o200k_base. On cl100k_base, however, that study found Marathi higher (5.84 against 4.08), while we found no significant difference (4.79 against 4.99). The two studies use different texts and ours is much smaller, so the difference may come from the sentences used.

The second finding is that the advantage of typing Hindi in Roman letters has mostly gone. On older encodings Hinglish used less than half the tokens of Devanagari Hindi, which gave users a practical reason to type in Roman letters. On o200k_base this reversed for two of the three spellings. A developer choosing between asking users to type in Devanagari or romanized form should check the specific tokenizer.

The third finding is that Marathi did not benefit as much as Hindi. Both were similar on older encodings, but on o200k_base Marathi was 17% higher than Hindi. We have not tested why this happens, for example whether it relates to the amount of Marathi text used to build each tokenizer.

A. Limitations

- The corpus is small (99 sentences) and was prepared with AI assistance and by a single author, so it may not cover all styles of Hinglish or all domains.
- We tested only OpenAI encodings, so the results may not apply to other model families.
- r50k_base and p50k_base gave identical counts, so we effectively compared three encodings.
- Inflation is measured against English in each encoding, so it shows the relative gap, not the absolute token count.
- The second Hinglish spelling also uses more English words, so spelling and word choice are mixed.
- Token count is not the same as model quality, and we did not test how well the models understand each language.
- The sentences were first drafted with AI assistance, so they may read differently from text written naturally by native speakers.
- Results are a snapshot taken on 24 October 2026 and tokenizers may change.

VI. CONCLUSION

We measured how four OpenAI tokenizer generations handle English, Marathi, Hindi and romanized Hinglish. Marathi and Hindi inflation fell from about 7.5 times English to 1.8 and 1.5 times, but Marathi remains higher than Hindi on the newest encoding. Hinglish was far cheaper than Hindi on older tokenizers and slightly more expensive on the newest, and its spelling changed token counts by more than a quarter on average. Future work should use a larger, verified corpus, tokenizers from other vendors, and Indic-focused tokenizers, and should add real prices to turn the relative cost into rupees.

ACKNOWLEDGMENT

The author thanks Dr. Madhavi Shamkuwar and ZIBACAR for guidance. AI tools (Claude, Anthropic) were used to draft the sentence corpus, help write the analysis code, and draft and edit the text. The author ran the experiments and takes responsibility for the content.

REFERENCES

[1] O. Ahia, S. Kumar, H. Gonen, J. Kasai, D. Mortensen, N. Smith, and Y. Tsvetkov, “Do all languages cost the same? Tokenization in the era of commercial language models,” in Proc. 2023 Conf. Empirical Methods in Natural Language Processing (EMNLP), Singapore, Dec. 2023, pp. 9904–9923, doi: 10.18653/v1/2023.emnlp-main.614.

[2] A. Petrov, E. La Malfa, P. Torr, and A. Bibi, “Language model tokenizers introduce unfairness between languages,” in Advances in Neural Information Processing Systems, vol. 36, 2023.

[3] H. K. Teklehaymanot and W. Nejdl, “Tokenization disparities as infrastructure bias: How subword systems create inequities in LLM access and efficiency,” arXiv:2510.12389, Oct. 2025.

ISSN: 3139-1141

[4] P. Srivastava, “The tokenizer tax: Quantifying and explaining the cross-lingual cost of subword tokenization for Indian languages,” arXiv:2607.24276, Jul. 2026.

[5] M. Velayuthan and K. Sarveswaran, “Egalitarian language representation in language models: It all begins with tokenizers,” in Proc. 31st Int. Conf. Computational Linguistics (COLING), 2025, pp. 5987–5996.

[6] A. A. Deshmukh and S. Deshmukh, “HingLM: Isolating the tokenizer’s effect for romanized Hindi–English code-mixed language,” NeurIPS 2026 Workshop on the Linguistic Principles for Foundation Models, 2026.

[7] P. Rust, J. Pfeiffer, I. Vulić, S. Ruder, and I. Gurevych, “How good is your tokenizer? On the monolingual performance of multilingual language models,” in Proc. 59th Annu. Meeting Assoc. Computational Linguistics (ACL-IJCNLP), Online, Aug. 2021, pp. 3118–3135, doi: 10.18653/v1/2021.acl-long.243.

[8] S. Dwivedi and K. Gopalan, “Comparative analysis of the intrinsic metrics for tokenizers and their effect on downstream tasks for Hindi and Marathi,” in Proc. 64th Annu. Meeting Assoc. Computational Linguistics (ACL), San Diego, CA, USA, Jul. 2026, pp. 22652–22663.

[9] R. Nayak and R. Joshi, “L3Cube-HingCorpus and HingBERT: A code mixed Hindi-English dataset and BERT language models,” in Proc. WILDRE-6 Workshop, Marseille, France, Jun. 2022, pp. 7–12.

[10] OpenAI, “tiktoken,” GitHub repository. [Online]. Available: https://github.com/openai/tiktoken (accessed Oct. 25, 2026).