Why some languages say more with less: A mathematical exploration of AI-Human Communication

Joshua Callary

April 13, 2026

Abstract

This essay explores how the choice of human language can influence the cost of AI-generated communication. By comparing languages mathematically through token counts, computational cost, and byte usage, it aims to determine which languages are the most efficient within current AI systems.

1 Introduction

Currently, more than 1 billion people use AI to help with work, study, and sometimes even everyday decisions [Kem25]. As more people around the world adopt AI, the demand for compute, storage, and energy keeps increasing. Currently this creates two main possibilities: either hardware deployment cannot keep up at a low cost, or some demographics are priced out of using AI at a competitive level. As different countries begin building their own technological sovereignty in AI, I started wondering whether some languages may have an advantage when it comes to token generation costs.

What is a token and tokeniser

This essay looks at compute, memory, and language through the idea of tokens.

A **token** is a small chunk of text that a language model reads and processes. A token can be a full word, part of a word, a number, or even punctuation. For example, the word "running" might be split into two tokens such as "run" and "ning". A **tokeniser** is the system that breaks raw text into these smaller pieces so the model can work with it.

This essay examines tokenisation and memory costs within AI systems that are still largely developed by English-speaking companies and standards, countries such as Russia and China are currently developing their own AI ecosystems, therefore calculations in this essay may change over time as they develop their own ecosystems further.

2 The Collection of Data

I collected a set of texts to produce rough estimates of language-related AI costs.

The five languages chosen for this essay are English, Russian, Arabic, Chinese, and Hindi. These languages were selected because they are widely spoken, come from different parts of the world, and belong to different language families. I expected that, if there were meaningful mathematical differences in token costs between languages, they would be easier to detect across a more diverse sample.

To test this, I selected texts from different literature associated with each language, together with one technical text to reduce bias. For each work, I used the native-language version as the base text rather than the English version, and selected roughly one paragraph from each source. Each passage was then translated into all other languages in the data set.

These translations were produced in several ways. I used official translations where possible, translated some sections myself in languages I know, and also used help from native speakers. A machine translator was only used when translation proved difficult.

In some cases, texts were first translated into English and then into the other target languages, since it is difficult to find translators who can work naturally and accurately translate across all five languages. However, whenever a direct translation was possible, it was used.

Token counts were measured using the OpenAI tokeniser[Ope26]. The main results are based on the tokenisation used for ChatGPT 5 and o3-style models. If we are to use older tokenisers the gap widens drastically more between languages. Each text is approximately 150–200 words once in English. The resulting token counts are shown in the table below.

Language	Alice	CNP	JTTW	TEOF	Godaan	Text
English	184	242	316	199	197	269
Russian	220	320	431	272	286	443
Arabic	196	340	493	207	321	415
Chinese	180	308	250	287	273	323
Hindi	212	364	469	305	230	537

Table 1: Token counts across languages and literary sources.

Alice in Wonderland = *Alice's Adventures in Wonderland*, by Lewis Carroll.
CNP = Преступление и наказание (Crime and Punishment), by Фёдор Достоевский (Fyodor Dostoevsky).
JTTW = 西游记 (Journey to the West), traditionally attributed to 吴承恩 (Wu Cheng'en).
TEOF = رسالة الغفران (The Epistle of Forgiveness), by أبو العلاء المعري (Abu al-Ala al-Ma'arri).
Godaan = गोदान, by प्रेमचंद (Premchand).
Text = A technical text about physics and mathematics.

*Chinese** = Chinese Simplified
*Arabic** = Arabic Standard

3 Output Token Results

$$\text{Range} = x_{\text{max}} - x_{\text{min}}$$

Language	Range
English	132
Russian	223
Arabic	297
Chinese	143
Hindi	325

Table 2: Range of token counts across texts for each language.

The ranges above are fairly large, which suggests that token counts vary significantly depending on the type of text being used. Because of this, a simple arithmetic mean may not fully reflect how efficient a language is in its own native creation. To account for this, I define a weighted average that gives extra importance to the native text while still keeping the translated texts in the calculation. Let $w =$ weight and $x =$ text token count.

$$\bar{x}_w = \frac{\sum w_i x_i}{\sum w_i}$$

$$w_{\text{native}} = 1, \qquad w_{\text{translated}} = \frac{2}{m}$$

$$\bar{x}_w = \frac{x_{\text{native}} + \frac{2}{m}\sum_{i=1}^{m} x_i^{(\text{translated})}}{1+2}$$

$$\bar{x}_w = \frac{x_{\text{native}} + \frac{2}{m}\sum_{i=1}^{m} x_i^{(\text{translated})}}{3}$$

$$\boxed{\bar{x}_w = \frac{1}{3}x_{\text{native}} + \frac{2}{3m}\sum_{i=1}^{m} x_i^{(\text{translated})}}$$

For this data set, each language has 6 separate texts: 1 native text and 5 translated texts. Therefore,

$$w_{\text{native}} = 1, \qquad w_{\text{translated}} = \frac{2}{5}$$

so for $m = 5$,

$$\bar{x}_w = \frac{1}{3}x_{\text{native}} + \frac{2}{15}(x_1 + x_2 + x_3 + x_4 + x_5)$$

Using this weighted mean gives the following results:

Language	$\bar{x}_w$	% of English
English	224.40	100.00%
Chinese	266.13	118.60%
Arabic	304.33	135.62%
Russian	326.93	145.69%
Hindi	328.27	146.29%

Table 3: Weighted token averages ordered by percentage relative to English.

Even after giving extra weight to each language's native text, English still has the lowest weighted token count in this data set. Chinese is the closest to English, while Arabic, Russian, and Hindi all are substantially higher. This suggests that the overall pattern is not just caused by translation effects alone, but reflects a broader difference in how efficiently these languages are tokenised under the chosen tokeniser.

4 Total Token and Byte Cost

4.1 Deriving the cumulative cost model from quadratic attention

In the original Transformer framework, self-attention is usually described as having quadratic complexity. If an input sequence has length n, then each token can attend to all n tokens, so the total number of possible interactions scales as

$$O(n^2).$$

[Vas+17]

As the sequence gets longer, the amount of attention computation rises very quickly. However, large language models do not produce all tokens at once. They generate text one token at a time using a causal decoder, meaning each new token can only attend to the tokens that came before it. So the amount of context grows step by step the first token attends to 1 token, the second to 2, the third to 3, and so on. For a sequence of length n, the total number of attention interactions is

$$1 + 2 + 3 + \cdots + n = \sum_{i=1}^{n} i.$$

Using the triangular number formula,

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

So instead of using the rough expression n^2, a more precise count for causal decoding is

$$\frac{n(n+1)}{2}.$$

Now let p be the number of input tokens in the prompt, and let r be the number of output tokens generated by the model.

For the input prompt, the context grows from length 1 up to length p. Therefore, the prompt-processing cost is

$$\sum_{i=1}^{p} i = \frac{p(p+1)}{2}.$$

After the prompt has been processed, the model begins generating output tokens one by one. The first output token attends to $p+1$ tokens, the second attends to $p+2$ tokens, and the rth output token attends to $p+r$ tokens. Therefore, the generation cost is

$$\sum_{j=1}^{r} (p+j).$$

Expanding this gives

$$\sum_{j=1}^{r} (p+j) = \sum_{j=1}^{r} p + \sum_{j=1}^{r} j = pr + \frac{r(r+1)}{2}.$$

So the total computational cost is

$$C(p,r) = \sum_{i=1}^{p} i + \sum_{j=1}^{r} (p+j).$$

Substituting the two parts gives

$$C(p,r) = \frac{p(p+1)}{2} + pr + \frac{r(r+1)}{2}.$$

Therefore the final cost model is

$$\boxed{C(p,r) = \frac{p(p+1)}{2} + pr + \frac{r(r+1)}{2}}$$

This model is useful because it does more than just say the cost is quadratic. It shows exactly where that cost comes from: first from reading the prompt, and then from generating the response while carrying the full earlier context forward each time.

At global scale, AI systems can process extremely large volumes of text each day. For example, Gemini models were processing 7 billion tokens per minute[Pic25]. From this point onward, I use a baseline of 1 trillion English tokens per day, and then scale the other languages using the percentage differences calculated earlier. This gives a simple way to estimate how token inflation changes the total attention cost.

Since $C(p,r)$ gives the cost of one full request, but the daily traffic is measured in total tokens rather than total requests, I divide by $p+r$ to convert the request cost into an average cost per token. I then multiply by the total number of daily tokens.

Language	Avg. Input	Avg. Output	Daily	Estimated C_{day}	Multi
English	256.00	512.00	1.00×10^{12}	3.85×10^{14}	$1.00\times$
Chinese	303.61	607.21	1.19×10^{12}	5.40×10^{14}	$1.40\times$
Arabic	347.19	694.37	1.36×10^{12}	7.07×10^{14}	$1.83\times$
Russian	372.97	745.94	1.46×10^{12}	8.16×10^{14}	$2.12\times$
Hindi	374.50	748.99	1.46×10^{12}	8.23×10^{14}	$2.14\times$

Table 4: Estimated daily attention using an average ordinary-chat request of 256 input tokens and 512 output tokens for English, scaled by relative token inflation across languages. The final column shows the multiplier relative to English [Cer24].

4.2 Bytes

Tokens are important, but bytes also matter. Even if two languages say the same thing, they may use a different number of characters, and each character may take up a different number of bytes in UTF-8. This affects storage, memory use, and data transfer[Uni25].

To keep things simple, I compare each language to a baseline English characters. I then estimate how many characters each language would need to express the same content, and multiply that by the typical number of bytes per character.

For this essay, I use the following approximate byte sizes[Uni25]:

$$b_{\text{English}} \approx 1, \quad b_{\text{Russian}} \approx 2, \quad b_{\text{Arabic}} \approx 2, \quad b_{\text{Hindi}} \approx 3, \quad b_{\text{Chinese}} \approx 3$$

Using this, the estimated byte count is

$$\text{Estimated Bytes} = (\text{equivalent number of characters}) \times (\text{bytes per character})$$

The Character factor is based off standard expansion and contraction tables[And26; Scr20].

Language	Char Factor	Bytes/Char	Estimated Bytes
English	1.000	1	1.000
Russian	1.150	2	2.300
Arabic	1.225	2	2.450
Hindi	1.250	3	3.750
Chinese	0.600	3	1.800

Table 5: Approximate byte requirements for equivalent content relative to 1 English character.

This table shows that byte cost does not always follow token cost exactly. For example, Chinese may use fewer characters, but each character usually takes more bytes than an English character. Russian and Arabic both use about 2 bytes per character in this simplified model, but Arabic is slightly higher overall because the model assumes it needs more characters to express the same meaning.

5 Cost Analysis

A combined compute-and-byte multiplier

Up to this point, I have looked at two different kinds of cost. The first is the compute cost, which comes from token generation and attention. The second is the byte cost, which comes from storage, memory, and networking.

To combine these into one final cost, let

$$C_L = \text{compute multiplier for language } L$$

and let

$$B_L = \text{byte multiplier for language } L.$$

Then we can introduce a parameter λ, which tells us how much byte-related cost matters compared to compute cost.

Using this, I define the raw combined score for language L as

$$R_L = C_L + \lambda B_L.$$

This works in a really simple way C_L resprents the compute part, while λB_L adds the byte part after weighting it by how important bytes are in the model.

For English, both multipliers are equal to 1, so the English raw score is

$$R_E = 1 + \lambda.$$

To keep English as the baseline, I divide every raw score by the English value. This gives the final combined multiplier

$$\boxed{M_L = \frac{R_L}{R_E} = \frac{C_L + \lambda B_L}{1 + \lambda}}.$$

The formula can be read quite simply:

- if $\lambda = 0$, only compute cost matters;
- if $\lambda > 0$, byte cost also contributes;
- as λ becomes larger, byte cost matters more.

Using the values from Tables 4 and 5, the combined multiplier for each language is:

Language	C_L	B_L	$M_L(\lambda)$
English	1.00	1.00	$\frac{1+\lambda}{1+\lambda} = 1$
Chinese	1.40	1.80	$\frac{1.40 + 1.80\lambda}{1+\lambda}$
Arabic	1.83	2.45	$\frac{1.83 + 2.45\lambda}{1+\lambda}$
Russian	2.12	2.30	$\frac{2.12 + 2.30\lambda}{1+\lambda}$
Hindi	2.14	3.75	$\frac{2.14 + 3.75\lambda}{1+\lambda}$

Table 6: Combined compute-and-byte multiplier relative to English.

As an example, we can take $\lambda = 0.10$. This means byte-related cost is treated as a small, but still non-zero, part of the total. The final multipliers are then:

Language	$M_L(0.10)$
English	1.00
Chinese	1.44
Arabic	1.89
Russian	2.14
Hindi	2.29

Table 7: Combined multiplier when $\lambda = 0.10$.

This result shows that once compute and byte costs are combined, English still remains the cheapest baseline in this model. Chinese stays the closest to English, while Arabic, Russian, and especially Hindi remain significantly more expensive. So even when byte cost is only given a small weight, the overall ranking does not change much but does continue to increase the gap to English.

6 What if someone can speak two languages?

A bilingual user might keep a sentence mostly in English, but replace a few expensive English words or phrases with shorter ones from another language. This could affect both the input and the output, because a shorter prompt may lead to a shorter reply, and mixed-language prompting may also change how the model responds.

Let the original English request have p_E input tokens and r_E output tokens. Using the cost model

$$C(p, r) = \frac{p(p+1)}{2} + pr + \frac{r(r+1)}{2},$$

the English-only cost is

$$C(p_E, r_E) = \frac{p_E(p_E+1)}{2} + p_E r_E + \frac{r_E(r_E+1)}{2}.$$

Now suppose that in the input, E_{in} English tokens are removed and replaced by L_{in} tokens from another language. Also let H_{in} be an extra input penalty caused by

switching, clarification, or repetition. Then the new input length is

$$p_{\text{mix}} = p_E - E_{\text{in}} + L_{\text{in}} + H_{\text{in}}.$$

Now also suppose that in the output, E_{out} English tokens are removed and replaced by L_{out} tokens from another language, with an extra output penalty of H_{out}. Then the new output length is

$$r_{\text{mix}} = r_E - E_{\text{out}} + L_{\text{out}} + H_{\text{out}}.$$

So the mixed-language cost becomes

$$C(p_{\text{mix}}, r_{\text{mix}}) = \frac{p_{\text{mix}}(p_{\text{mix}} + 1)}{2} + p_{\text{mix}} r_{\text{mix}} + \frac{r_{\text{mix}}(r_{\text{mix}} + 1)}{2}.$$

To compare this with the original English version, define the multiplier

$$M = \frac{C(p_{\text{mix}}, r_{\text{mix}})}{C(p_E, r_E)}.$$

The bilingual substitution is useful only when $M < 1$.

We can also simplify our cost model.

$$C(p, r) = \frac{(p + r)(p + r + 1)}{2}.$$

So in this simplified model, the cost depends only on the total number of tokens. Therefore, bilingual substitution helps exactly when

$$p_{\text{mix}} + r_{\text{mix}} < p_E + r_E.$$

Substituting the definitions of p_{mix} and r_{mix} gives

$$p_E - E_{\text{in}} + L_{\text{in}} + H_{\text{in}} + r_E - E_{\text{out}} + L_{\text{out}} + H_{\text{out}} < p_E + r_E.$$

After cancelling $p_E + r_E$ from both sides, this becomes

$$\boxed{E_{\text{in}} + E_{\text{out}} > L_{\text{in}} + L_{\text{out}} + H_{\text{in}} + H_{\text{out}}}.$$

So bilingual substitution is only worthwhile when the token savings from removing English words are larger than the extra tokens and penalties created by mixing languages.

The penalty varibale is not fixed in place, and it may decrease over time as tokenisers improve and multilingual models become more efficient. This makes bilingual prompting an interesting area for future study, especially if languages could be combined for more efficient AI-human communication. This might need to be an essay for next year!

7 Conclusion

This essay set out to test whether language can change the cost of using AI, and the results show that it can. Across the models used here, English consistently came out

as the most efficient, while Chinese was usually the closest competitor. Arabic, Russian, and Hindi generally had a higher overall cost.

This does not mean these languages are worse for communication. Instead, it suggests that current AI systems are still heavily shaped by the standards and infrastructure they were built on, which remain strongly English centered.

The mathematics shows that language choice can have real computational consequences. In the future, better multilingual tokenisers, locally trained models, or hybrid forms of AI-human communication may reduce these gaps. For now, English appears to keep a clear efficiency advantage in large-scale AI systems.

References

[al-33] Abu al-Ala al-Ma'arri. *The Epistle of Forgiveness*. Classical Arabic work;. Date is approximate. 1033.

[Che92] Wu Cheng'en. *Journey to the West*. Classical Chinese novel traditionally attributed to Wu Cheng'en; original title 西游记. 1592.

[Car65] Lewis Carroll. *Alice's Adventures in Wonderland*. London: Macmillan, 1865.

[Dos66] Fyodor Dostoevsky. *Crime and Punishment*. Originally published in Russian as *Преступление и наказание*. 1866.

[Pre36] Premchand. *Godaan*. 1936.

[Vas+17] Ashish Vaswani et al. "Attention Is All You Need." In: *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*. 2017. URL: https://papers.nips.cc/paper/7181-attention-is-all-you-need.

[Scr20] Scriptis. *Multilingual Desktop Publishing: Always an Adventure!* 2020. URL: https://www.scriptis.com/multilingual-desktop-publishing/ (visited on 04/10/2026).

[Cer24] Cerebrium. *Benchmarking vLLM, SGLang and TensorRT for Llama 3.1 API*. 2024. URL: https://cerebrium.ai/blog/benchmarking-vllm-sglang-tensorrt-for-llama-3-1-api (visited on 04/10/2026).

[Kem25] Simon Kemp. *Digital 2026: More than 1 Billion People Use AI*. 2025. URL: https://datareportal.com/reports/digital-2026-one-billion-people-using-ai (visited on 03/28/2026).

[Pic25] Sundar Pichai. *Q3 earnings call: Remarks from our CEO*. The Keyword, Google blog. Oct. 2025. URL: https://blog.google/company-news/inside-google/message-ceo/alphabet-earnings-q3-2025/ (visited on 04/10/2026).

[Uni25] Unicode Consortium. *Unicode Character Code Charts*. 2025. URL: https://www.unicode.org/charts/ (visited on 04/10/2026).

[And26] Andiamo. *Expansion and Contraction Factors*. 2026. URL: https://www.andiamo.co.uk/resources/expansion-and-contraction-factors/ (visited on 04/10/2026).

[Ope26] OpenAI. *Tokenizer - OpenAI API*. 2026. URL: https://platform.openai.com/tokenizer (visited on 04/01/2026).