CS 696 Applied Large Langauge Models
Spring Semester, 2025
Doc 5 Tokens
Jan 23, 2025

LLM News - Cleaning Robot

https://twitter.com/JannikGrothusen/status/1852790503823057073

GPT-4o Powered Robot

- 4 days to build

Afternoon Project - Four Days of AI Lies

https://nemo.foo/blog/day-4-of-an-afternoon-project

Side Project - Hardware, Networking

Desktop device using Zig protocol

AI is great for generating ideas or drafting code, but it doesn’t understand

> AI isn’t a co-pilot; it’s a junior dev faking competence. Trust it at your own risk.
>
> Like Icarus, my codebase is irrecoverable. A tangled heap of wing fragments and melted wax, dripping with half-baked ideas and unsupervised AI chaos.

Debugging my own failures has always been one of the best ways to understand a new language or concept. Relying on AI to “fix” things for me short-circuited that learning process. As a result, I’m left with no deeper understanding of Zig than when I started.

DeepSeek-R1 LLM

Functionality - about the same as Open AI 4o

From Chinese company

- US trade restrictions do not allow high-end Nvidia hardware to be sold to China

Estimates

- Needed %6 of computing resources to train than 4o
- They probably had contraband Nvidia hardware

Open source, liberal license

Published detailed Tech Reports

Nvidia lost just under $600 Billion in market value

UCB graduate student reproduced R1's key technologies for $30

- Complex reasoning in small models (1.5B parameters)
- https://xyzlabs.substack.com/p/berkeley-researchers-replicate-deepseek

Bitter (AI) Lesson

Rich Sutton UT Austin, 2019

https://www.cs.utexas.edu/~eunsol/courses/data/bitter_lesson.pdf

Biggest lesson from 70 years of AI research

General methods that leverage computation are ultimately the most effective, and by a large margin

Chess

- Researchers tried using human knowledge to create chess programs
- Just using deep search was more effective

Go

- Researchers tried using human knowledge to create go programs
- Deep search and self-play were more effective

Speech Recognition

- Forget grammar etc, use Markov models, deep learning, LLMs

Computer Vision

- Deep learning

Bitter (AI) Lesson Rich Sutton UT Austin, 2019

Bitter lesson

Building in how we think we think does not work in the long run

Eventual success is tinged with bitterness,

because it is success over a favored, human-centric approach

Two methods that seem to scale arbitrarily are search and learning

Jevons Paradox

When a resource becomes more efficient to use

We expect total resource consumption to decrease

But as the price falls, total consumption will rise

Coal

Watt steam engine was greatly more efficient than the one it replaced

Coal consumption increased

Water

Developing a reservoir to reduce the risk of water shortage

Can actually increase the risk

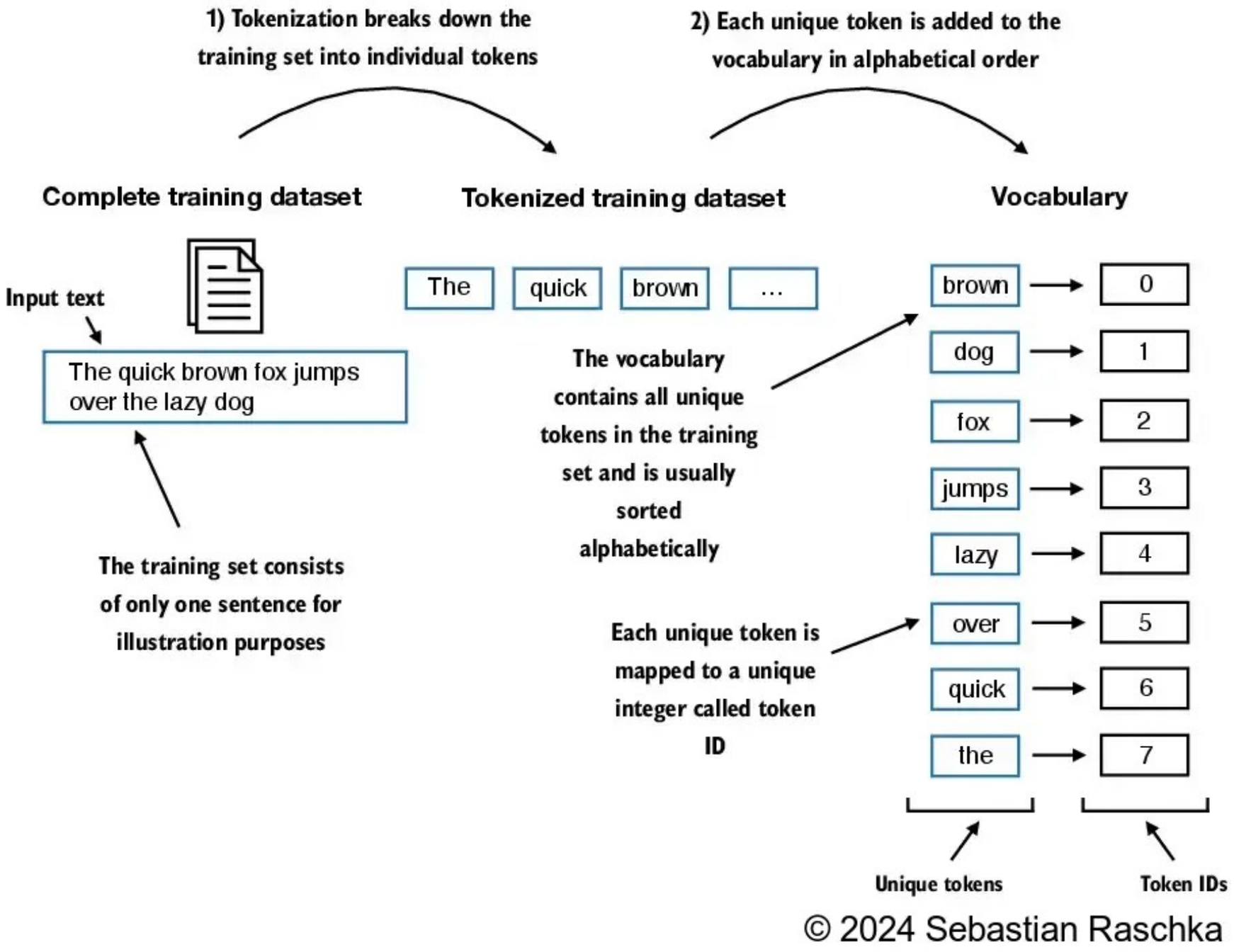
Input

LLMs cannot process raw data directly

- Text
- Sound
- Images
- Video

Text

- Convert text into numbers 0 through n


1) Tokenization breaks down the training set into individual tokens
2) Each unique token is added to the vocabulary in alphabetical order
Complete training dataset
Tokenized training dataset
Vocabulary
Input text
The quick brown fox jumps over the lazy dog
The training set consists of only one sentence for illustration purposes
The
quick
brown
...
The vocabulary contains all unique tokens in the training set and is usually sorted alphabetically
Each unique token is mapped to a unique integer called token ID
brown → 0
dog → 1
fox → 2
jumps → 3
lazy → 4
over → 5
quick → 6
the → 7
Unique tokens
Token IDs
© 2024 Sebastian Raschka

Overview

Create a token dictionary mapping tokens to integers (token ids)

- Collect all words in your possible input
- Tokenize them
- Add special tokens
- Sort the tokens & assign them integers 0 - n

LLM Input

- Tokenize the input
- Use the token dictionary to convert tokens to integers (token ids)
- Sequence of integers is the input to the model
- Output is sequence of token ids
- Use a token dictionary to convert token IDs to text

Each Model has its own Tokenizer

Example string: "antidisestablishmentarianism"

r50k_base: 5 tokens (gpt2)
token integers: [415, 29207, 44390, 3699, 1042]
token bytes: [b'ant', b'idis', b'establishment', b'arian', b'ism']

cl100k_base: 6 tokens (gpt-4, gpt-4-turbo, gpt-3.5-turbo)
token integers: [519, 85342, 34500, 479, 8997, 2191]
token bytes: [b'ant', b'idis', b'establish', b'ment', b'arian', b'ism']

o200k_base: 6 tokens (gpt-40, gpt-40-mini)
token integers: [493, 129901, 376, 160388, 21203, 2367]
token bytes: [b'ant', b'idis', b'est', b'ablishment', b'arian', b'ism']

Example string: "Goodbye, and thanks for all the fish!"

r50k_base: 10 tokens (gpt2)
token integers: [10248, 16390, 11, 290, 5176, 329, 477, 262, 5916, 0]
token bytes: [b'Good', b'bye', b',', b' and', b' thanks', b' for', b' all', b' the', b' fish', b'!']

cl100k_base: 10 tokens (gpt-4, gpt-4-turbo, gpt-3.5-turbo)
token integers: [15571, 29474, 11, 323, 9523, 369, 682, 279, 7795, 0]
token bytes: [b'Good', b'bye', b',', b' and', b' thanks', b' for', b' all', b' the', b' fish', b'!']

o200k_base: 10 tokens (gpt-40, gpt-40-mini)
token integers: [17212, 43571, 11, 326, 11707, 395, 722, 290, 13897, 0]
token bytes: [b'Good', b'bye', b',', b' and', b' thanks', b' for', b' all', b' the', b' fish', b'!']

Root Words

Example string: "fun funny funnier funniest funnel fundamentalist functional"

cl100k_base: 10 tokens (gpt-4, gpt-4-turbo, gpt-3.5-turbo)
token integers: [12158, 15526, 2523, 90083, 2523, 85880, 61319, 16188, 380, 16003]
token bytes: [b'fun', b' funny', b' fun', b'nier', b' fun', b'niest', b' funnel', b' fundamental', b'ist', b' functional']

o200k_base: 9 tokens (gpt-40, gpt-40-mini)
token integers: [18142, 21217, 2827, 25238, 174952, 50625, 18864, 421, 20483]
token bytes: [b'fun', b' funny', b' fun', b'nier', b' funniest', b' funnel', b' fundamental', b'ist', b' functional']

Non-English

Example string: "नमस्ते, आप कैसे हैं"

r50k_base: 31 tokens (gpt2)
token integers: [11976, 101, 11976, 106, 11976, 116, 24231, 235, 11976, 97, 24231, 229, 11, 28225, 228, 11976, 103, 28225, 243, 24231, 230, 11976, 116, 24231, 229, 28225, 117, 24231, 230, 11976, 224]

o200k_base: 8 tokens (gpt-40, gpt-40-mini)
token integers: [998, 1637, 14681, 628, 11, 9717, 63073, 6374]

Token Limits & Cost - Open AI

Model	Input Context Window - Tokens	Output limit - Tokens	Speed Tokens per second	Cost per Million Tokens
GPT-3.5 Turbo	16,385	4,096	121.5	Input $0.50 Output $1.50
GPT-4 Turbo	128,000	4,096	39.3	Input $10 Output $30
GPT-4o	128,000	16,384	134.9	Input $2.50 Output $10
o1-preview	128,000	32,000	151.3	Input $15 Output $60

The Verdict.txt - Chapter two

20479 Characters
3,747 Words

r50k_base: 5145 tokens

p50k_base: 5145 tokens

cl100k_base: 4943 tokens

o200k_base: 4836 tokens

tiktoken https://github.com/openai/tiktoken

Fast BPE tokeniser for use with OpenAI's models

pip install tiktoken

```
import tiktoken
enc = tiktoken.get_encoding("o200k_base")
assert enc.decode(enc.encode("hello world")) == "hello world"

# To get the tokeniser corresponding to a specific model in the OpenAI API:
enc = tiktoken.encoding_for_model("gpt-4o")
```

Source Code for Token Examples

```
import tiktoken

def compare_encodings(example_string: str) -> None:
    """Prints a comparison of three string encodings."""
    # print the example string
    print(f'\nExample string: “{example_string}"')

    # for each encoding, print the # of tokens, the token integers, and the token bytes
for encoding_name in ["r50k_base", "cl100k_base", "o200k_base"]:
    encoding = tiktoken.get_encoding(encoding_name)
    token_integers = encoding.encode(example_string)
    num_tokens = len(token_integers)
    token_bytes = [encoding.decode_single_token_bytes(token) for token in token_integers]
    print()
    print(f"{encoding_name}: {num_tokens} tokens")
    print(f"token integers: {token_integers}")
    print(f"token bytes: {token_bytes}”)
```

Downloading a Model

```
from transformers import AutoModelForCausalLM, AutoTokenizer

# Load model and tokenizer
model = AutoModelForCausalLM.from_pretrained(
    "microsoft/Phi-3-mini-4k-instruct",
    attn_implementation='eager',
    torch_dtype="auto",
    trust_remote_code=True,
)
tokenizer = AutoTokenizer.from_pretrained("microsoft/Phi-3-mini-4k-instruct")
```

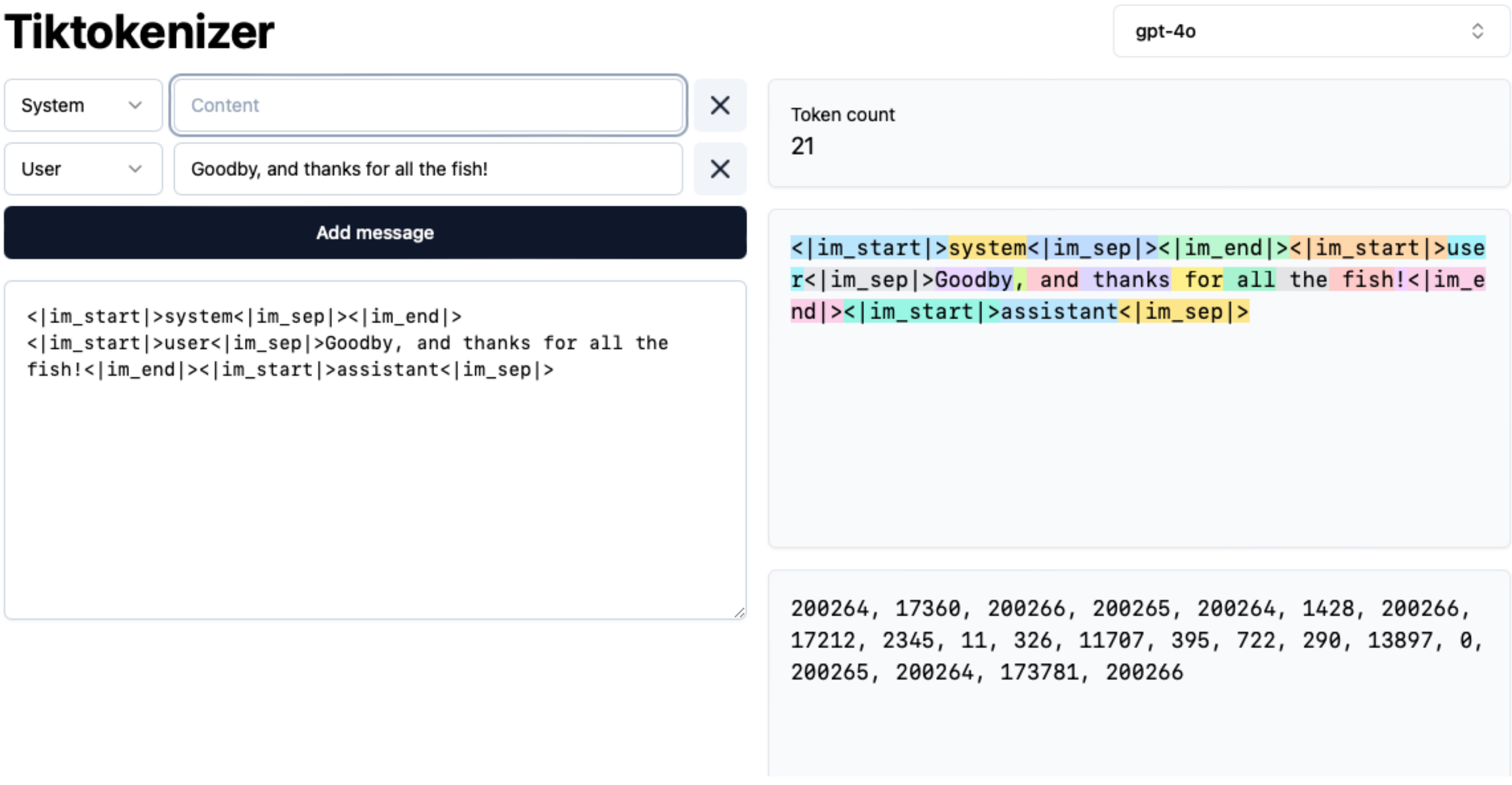
A Pipeline

```
from transformers import pipeline

# Create a pipeline
generator = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    return_full_text=True,
    max_new_tokens=500,
    do_sample=False
)
```

https://tiktokenizer.vercel.app/

Online tokenizer



Tokenizer Issues

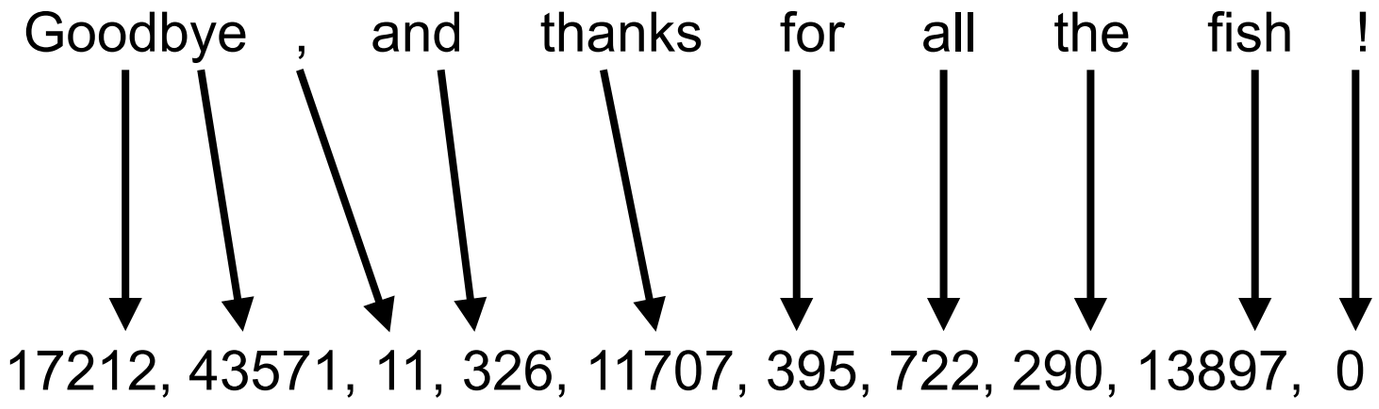
What is a token

Special Characters

Missing words in the token dictionary

<|unk|>

The Mapping


Goodbye , and thanks for all the fish !
17212, 43571, 11, 326, 11707, 395, 722, 290, 13897, 0

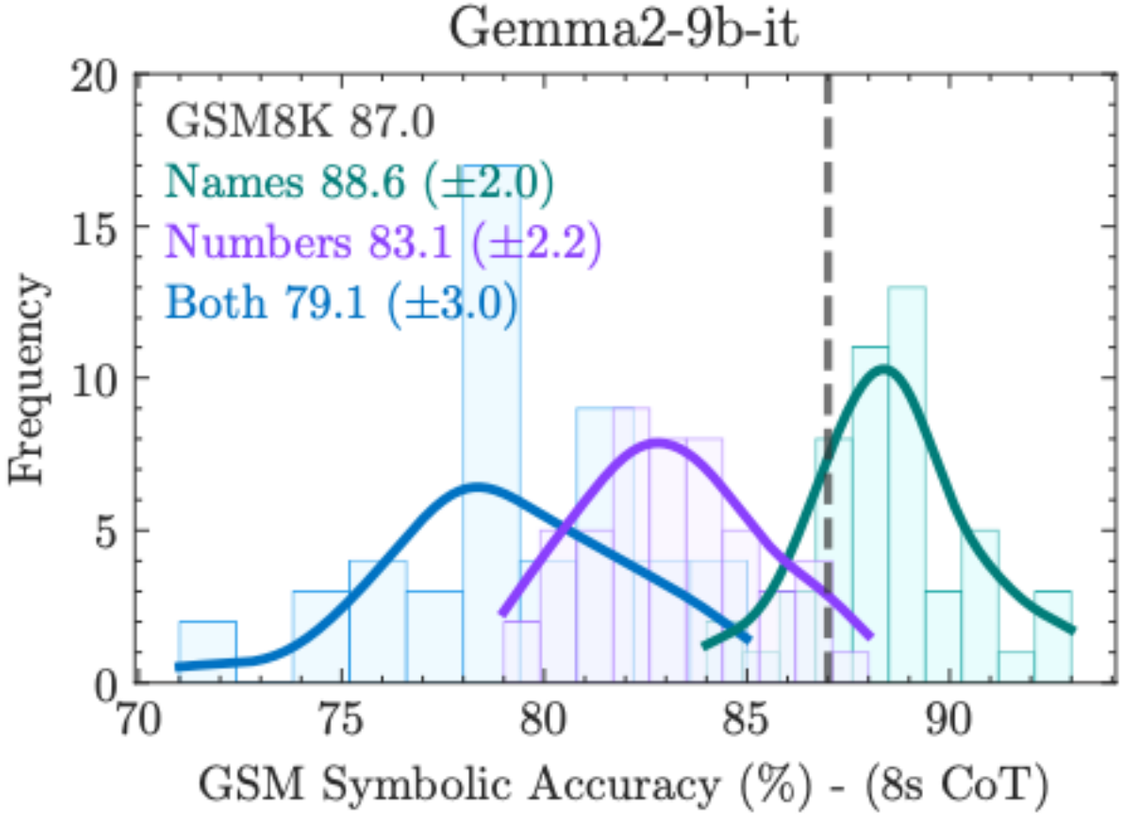
Limitations of Math Reasoning in LLMs

Paper from Apple Research

LLMs exhibit noticeable variance when responding to different instantiations of the same question

Performance of all models declines when only the numerical values are altered

Performance significantly deteriorates as the number of clauses in a question increases.



Limitations of Math Reasoning in LLMs

GSM-Symbolic-M1:

To make a call from a phone booth, you must pay $0.6 for each minute of your call.
How much would a 60-minute call cost?

GSM-Symbolic:

After 10 minutes, that price drops to $0.5 per minute. How much would a 60-minute call cost?

GSM-Symbolic-P1:

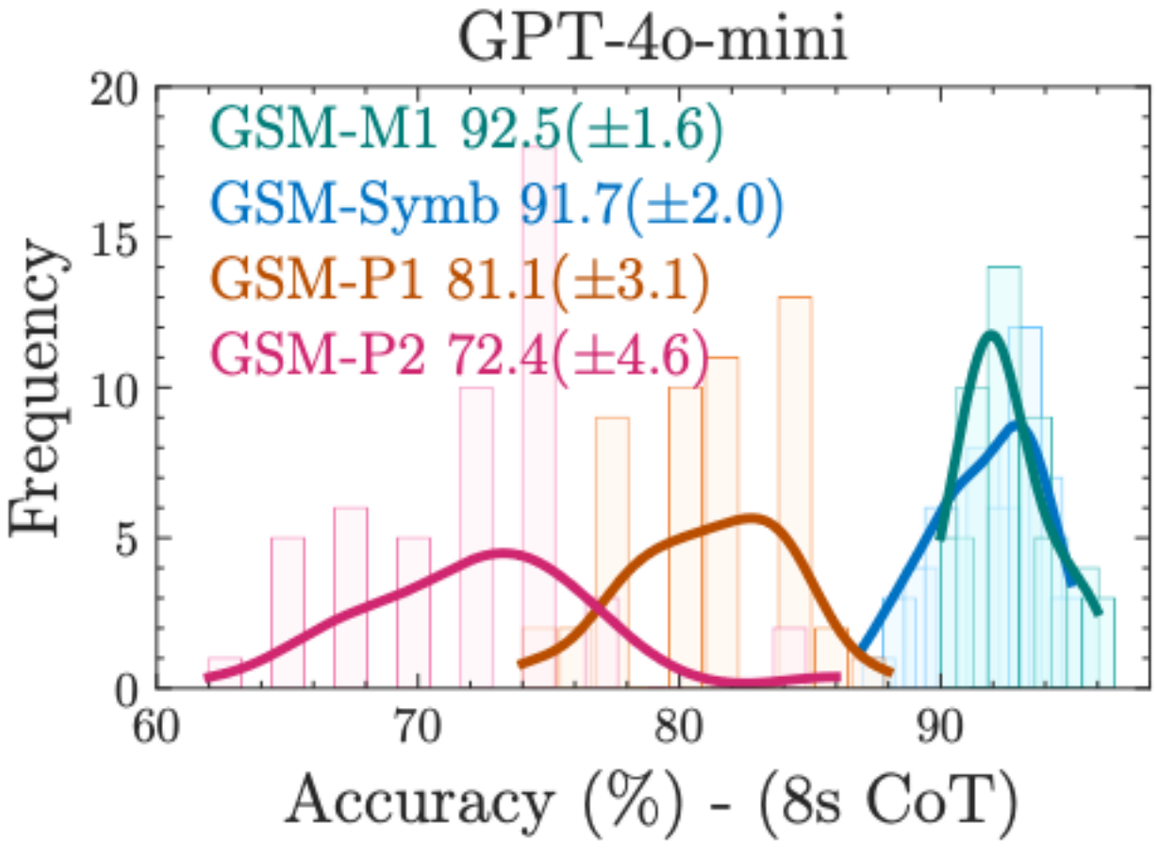
To make a call from a hotel room phone, you must pay $0.6 for each minute of your call.
After 10 minutes, that price drops to $0.5 per minute.
After 25 minutes from the start of the call, the price drops even more to $0.3 per minute.
How much would a 60-minute call cost?

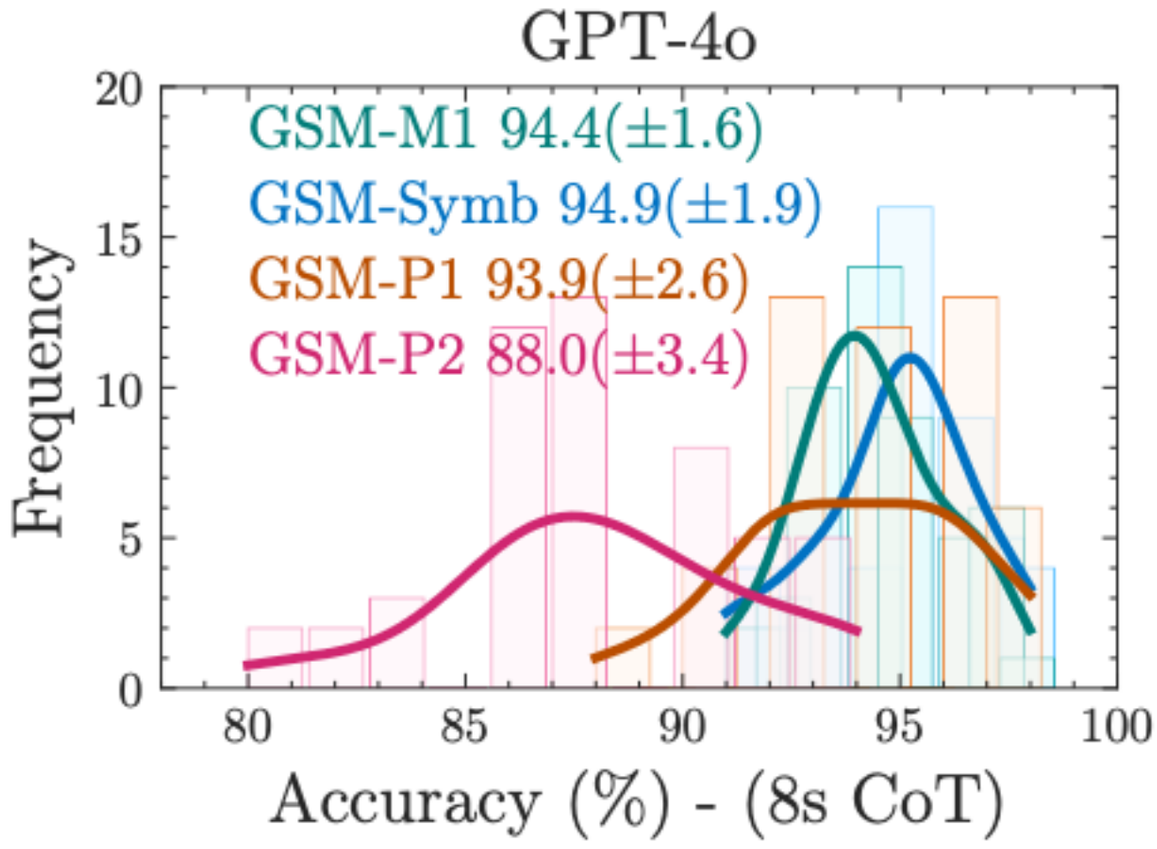
GSM-Symbolic-P2:

To make a call from a hotel room phone, you must pay $0.6 for each minute of your call.
After 10 minutes, the price drops to $0.5 per minute.
After 25 minutes from the start of the call, the price drops even more to $0.3 per minute.
If your total bill is more than $10, you get a 25% discount.
How much would a 60-minute call cost?

Limitations of Math Reasoning in LLMs

Performance variation on increasing complex questions





Filler Tokens

Let's Think Dot by Dot:

Hidden Computation in Transformer Language Models

Transformers can use meaningless filler tokens (e.g., '.....') in place of a chain of thought to solve two hard algorithmic tasks