Tokenizing with tiktoken and understanding the illusion of memory

Tokenization with Code

- Walkthrough of tokenizing text using Python and the tiktoken package
- Selected "Python Environments VMV" as the kernel
- Demonstrated encoding "Hi, my name is Ed" into tokens for GPT-4.1 Mini
- Tokens are just number IDs mapped from text fragments
- Decoded tokens back to text fragments, showing how each number maps to a word or part of a word
- Noted that some words (like "Ed") get their own token, others (like "banoffee") are split into parts
- Encouraged experimenting with finding words that have very low or very high token numbers

The Illusion of Memory in LLMs

- Explained that LLMs like GPT are stateless; each API call is independent
- Each call only knows about the input sequence it receives, not previous interactions
- To simulate memory, developers send the entire conversation history with each API call
- Messages are structured as a list of dictionaries with roles: "system", "user", and "assistant"
- Example: After introducing yourself as "Ed", asking "what's my name?" only works if the previous messages are included in the input
- The model predicts the next likely tokens based on the full input sequence, not any true internal memory

Key Takeaways

- LLMs do not remember past conversations unless the full context is provided with each call
- The "memory" effect is achieved by passing the whole conversation history every time
- This approach increases the number of tokens sent, which can slightly increase costs
- Cost per token is very low, but more context means more computation and higher (though still small) charges
- Understanding statelessness and context passing is critical for working with LLM APIs

Fun Experiment Suggested

- Try to find words with very low or very high token IDs using the tokenizer
- Observe how words are split or preserved as tokens