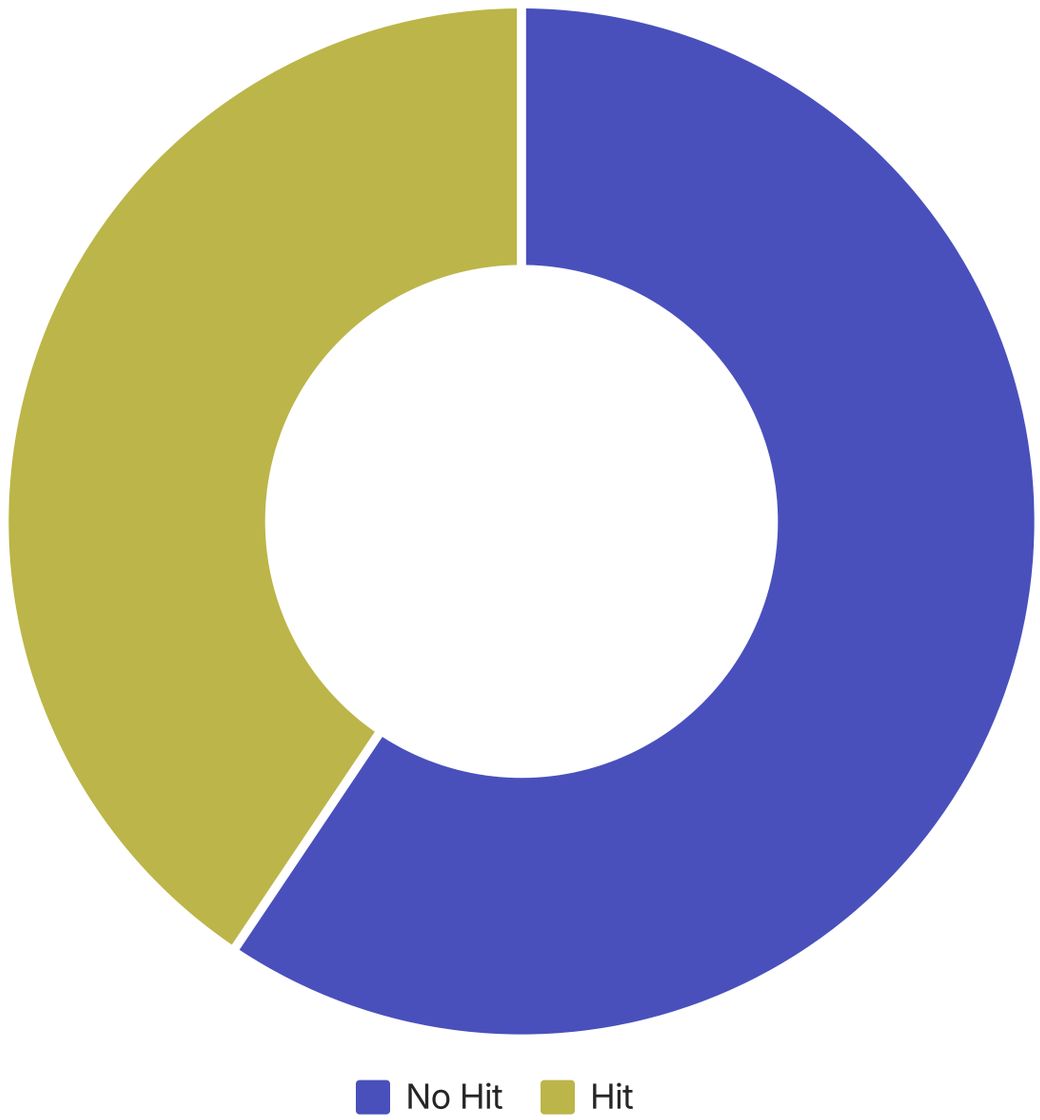
Week 3: Intro to Gemini and Ducky

by Alwin Lin

Recap from week 2:

The accuracy rate from last week



Recap from week 2:

Highlights from your feedbacks

Group 5 had the highest accuracy rate out of all

Group 4 reported to have detailed comments

Groups 3 & 7 was reported to have naming convention the likes of AI

Recap from week 2:

Most used:

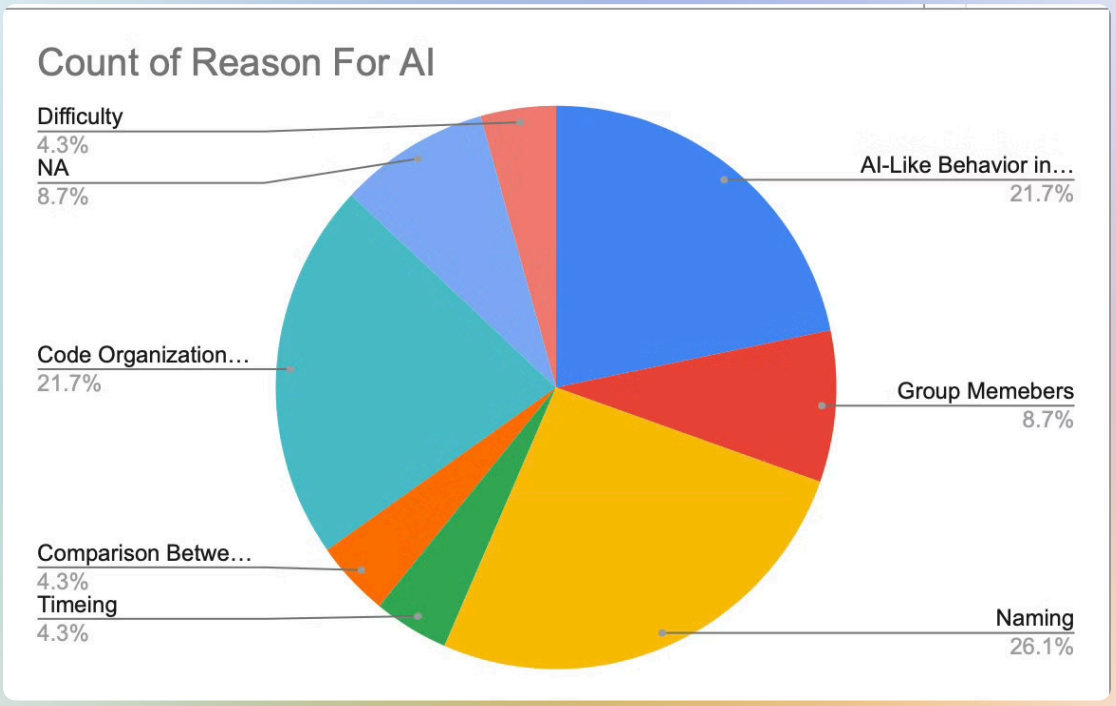
- AI-Like Behavior in Comments and Documentation (50%)
- Code Organization and Structure
- Naming

Most Reliable:

- The code is written in Python (100%)

Most unreliable:

- Naming Conventions (16.7%)

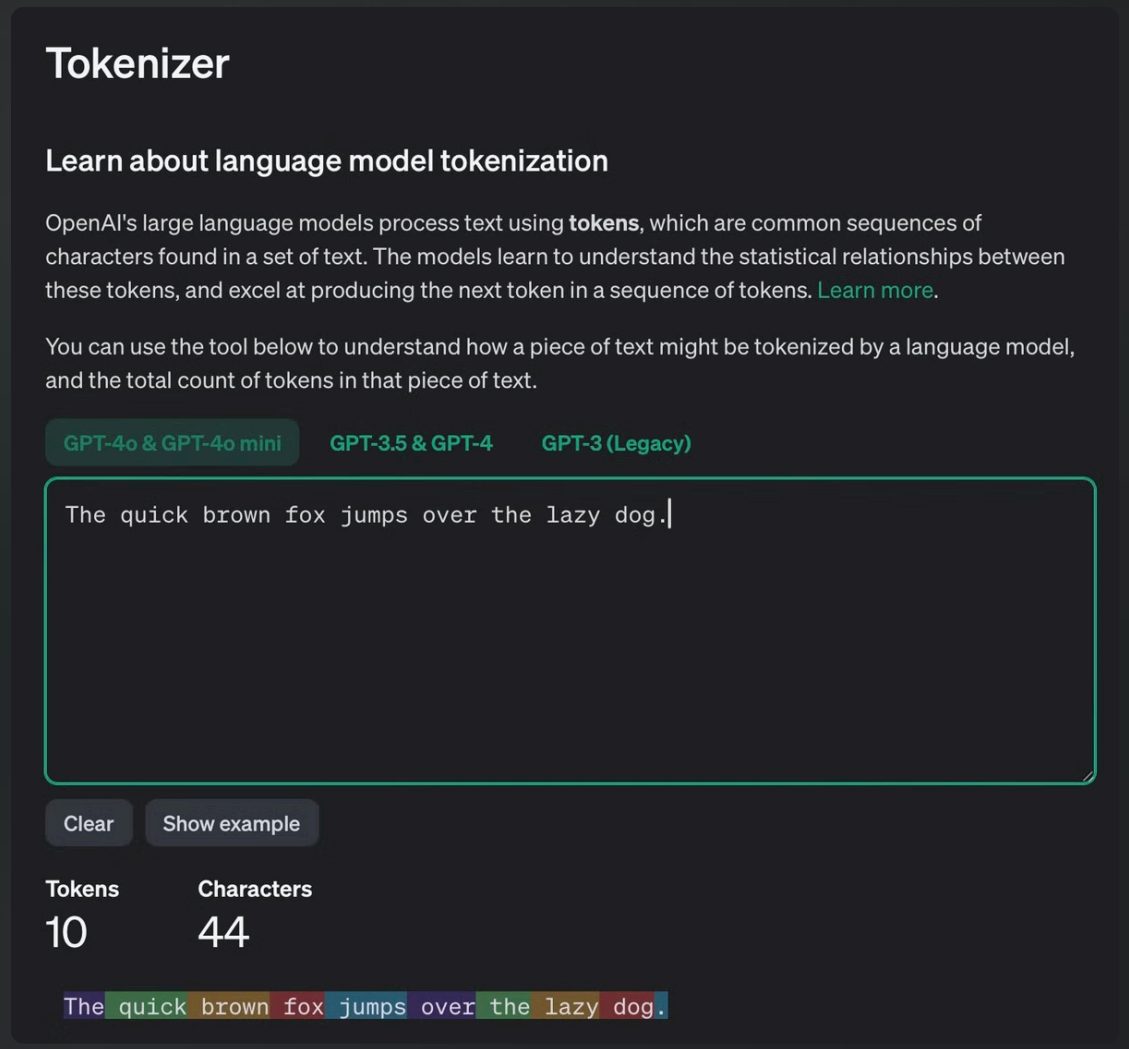


This week's agenda

- Technical 101 of LLMs
  - Tokens and Tokenization
  - Temperatures and Sampling
  - Environment Setup and API Basics
- Ducky AI Architecture Overview
- Documentation and Wrap up

What Are Tokens?

- Words and punctuation are broken into smaller units called tokens.
- Token limits dictate the maximum input/output size in queries.
- **Tokenizer Example:**
  - Sentence: "The quick brown fox jumps over the lazy dog."
  - Tokens: ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]



Why Tokens Matter?

Practical Implications:

- Longer inputs/output use more tokens, potentially exceeding limits.
- Efficient token use leads to clearer, cost-effective prompts.
- Different LLMs usually have different maximum token limits

Tokens price calculations:

- **GPT**
  - Input $10.00 / 1M tokens
  - Output $30.00 / 1M tokens
- **Gemini**
  - Input $0.075 / 1 million tokens
  - Output $0.30 / 1 million tokens

Property	Gemini 1.5 Flash	Gemini 1.5 Pro	Gemini 1.0 Pro Vision	Gemini 1.0 Pro
Total token limit (combined input and output) *	1,048,576 tokens	2,097,152 tokens	16,384 tokens	32,760 tokens
Output token limit *	8,192 tokens	8,192 tokens	2,048 tokens	8,192 tokens
Maximum number of images per request	3,000 images	3,000 images	16 images	N/A
Max base64 encoded image size	7 MB	7 MB	7 MB	N/A
Maximum PDF size	30 MB	30 MB	30 MB	N/A
Maximum number of video files per request	10 video files	10 video files	1 video file	N/A
Maximum video length (frames only)	~60 minutes of video	~60 minutes of video	2 minutes	N/A
Maximum video length (frames and audio)	~45 minutes of video	~45 minutes of video	N/A	N/A
Maximum number of audio files per request	1 audio file	1 audio file	N/A	N/A
Maximum audio length	~8.4 hours of audio	~8.4 hours of audio	N/A	N/A

** For all Gemini models, a token is equivalent to about 4 characters, so 100 tokens are about 60-80 English words. You can determine the total count of tokens in your requests using `countTokens`.*

Example:

- The following is the paper:
  - [Lost in the middle: How language models use long contexts]
  - Research paper
  - About 18 pages
- How many tokens?
  - A: 10K
  - B: 15K
  - C: 20K
  - D: 25K

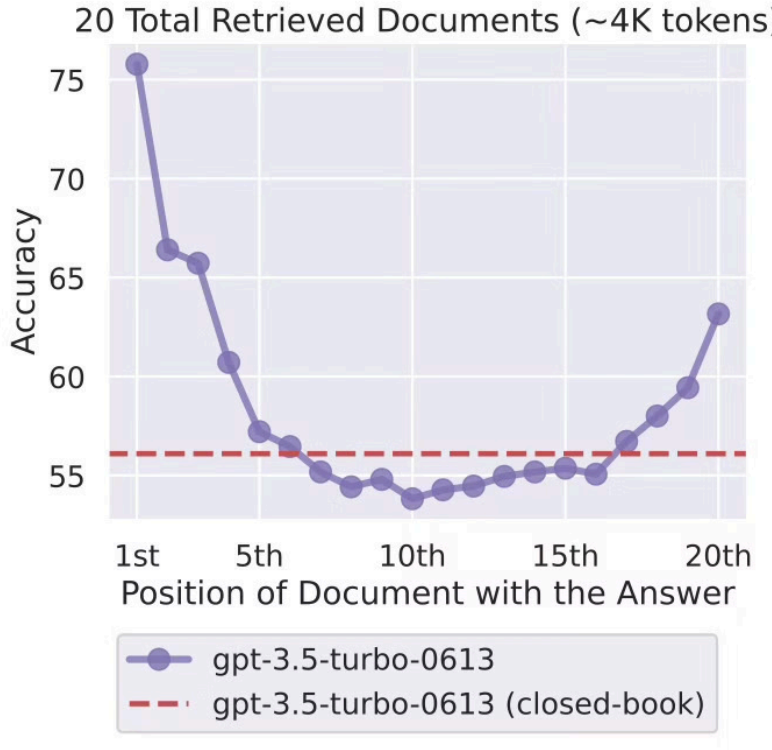


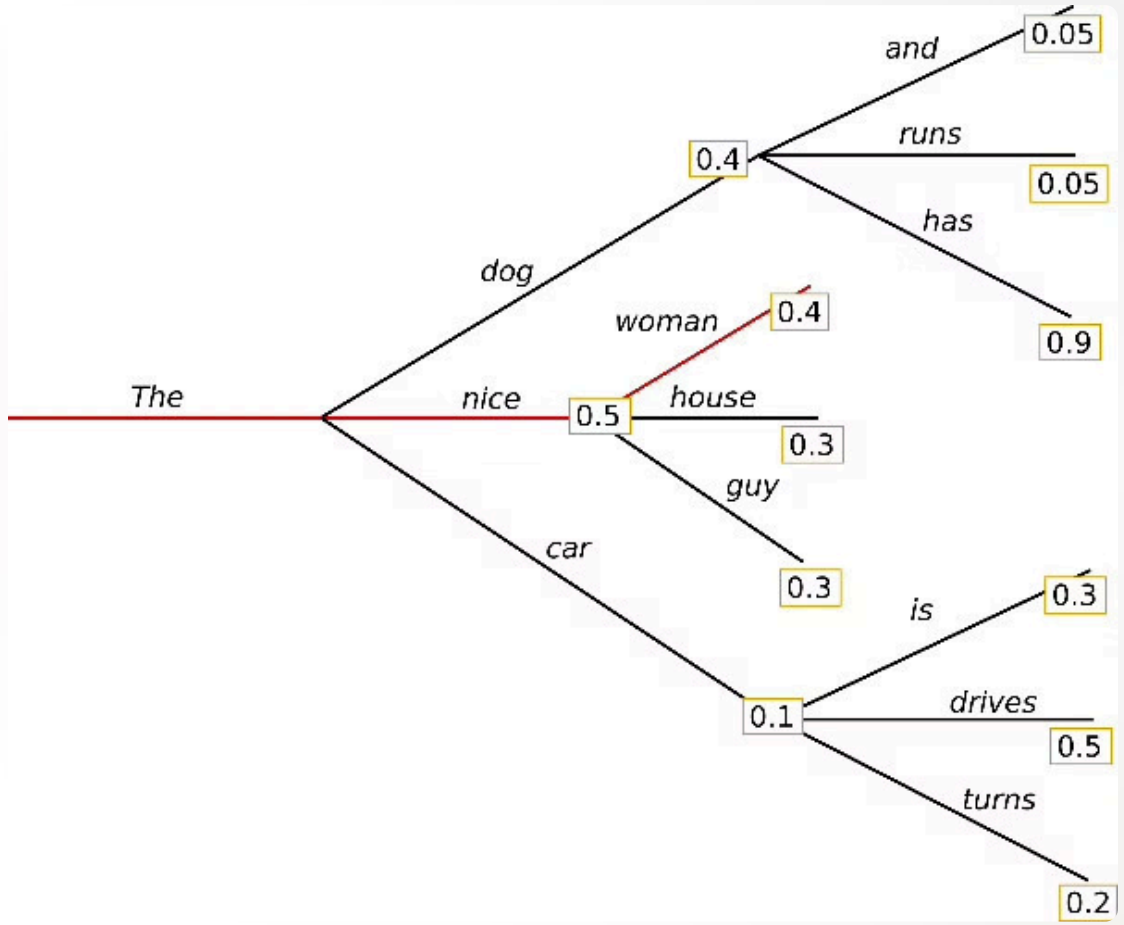
Figure 1: Changing the location of relevant information (in this case, the position of the passage that answers an input question) within the language model's input context results in a U-shaped performance curve—models are better at using relevant information that occurs at the very beginning (primacy bias) or end of its input context (recency bias), and performance degrades significantly when models must access and use information located in the middle of its input context.

Temperature and Randomness

- **What Is Temperature?**
  - A parameter controlling the randomness of AI outputs, goes 0~1
- **Why don't I find those slides in Chat?**
  - Most of the time it's hidden in either the API config or settings
- **Key Points**:
  - **Low Temperature (e.g., 0.2)**:
    - Focused, deterministic responses.
  - **High Temperature (e.g., 0.8)**:
    - Creative, diverse responses.

Sampling in Large Language Models

- **What is Sampling?**
  - How the model chooses the next token (word) in a sequence.
  - Allows the model to generate diverse and natural sounding text
- **Common Sampling methods**
  - Top-K: Top "k" results
  - Top-P: Top results that add up to P
  - Random: Well, random



Setup and API calls

Env Setup

```
!pip install -q -U google-generativeai
```

Sending your first API call

```
import google.generativeai as genai

genai.configure(api_key="YOUR_API_KEY")
model = genai.GenerativeModel("gemini-1.5-flash")
response = model.generate_content("Explain how AI works")
print(response.text)
```

A short intro to Ducky

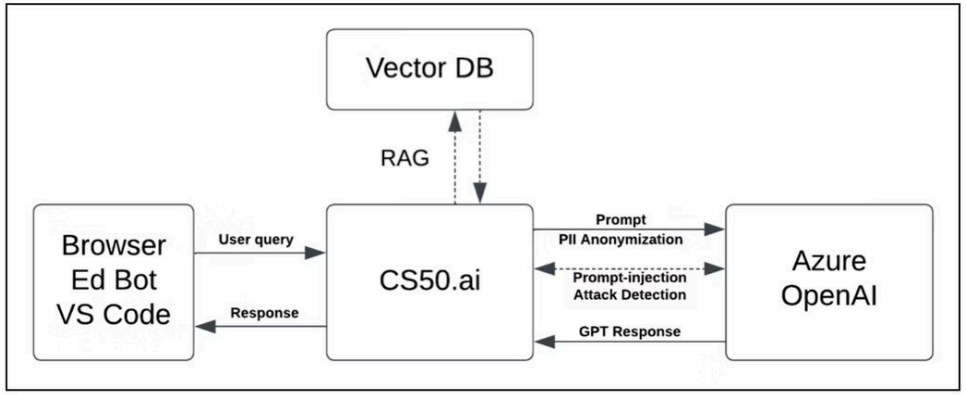


Figure 4: The system architecture of CS50.ai. GPT-4 generates responses to student queries and optionally employs a retrieval-augmented generation technique to improve response accuracy by incorporating facts from external sources.

Key Components

- Browser:
  - Takes and displays user query
- CS50.ai:
  - Security checks
  - Usage throttling
  - Passes query and relevant data to GPT
- Vector DB:
  - Stores embedding
  - Retrieves relevant data for RAG
- Azure OpenAI:
  - Data anonymization
  - Preventing Prompt injection attacks
  - GPT