Generative AI - Programming Assistant in the Classroom

Debasis Bhattacharya, JD, DBA
Professor
Bryson Uehara, Student
Jen Russo, Student
University of Hawaii Maui College
debasisb@hawaii.edu
maui.hawaii.edu/cybersecurity

Abstract

Generative AI (GenAI), as implemented in chatbots like ChatGPT, has greatly impacted higher education. This presentation will delve into the basics of large language models (LLM), prompt engineering, and the impact of these technologies in the classroom.

The presentation will provide a case study about how GenAI was used in an intermediate programming course at the University of Hawaii Maui College in Spring 2024.

The presentation will provide the latest updates in the core features and usage of popular AI tools such as Replit, ChatGPT from OpenAI, and Claude from Anthropic.

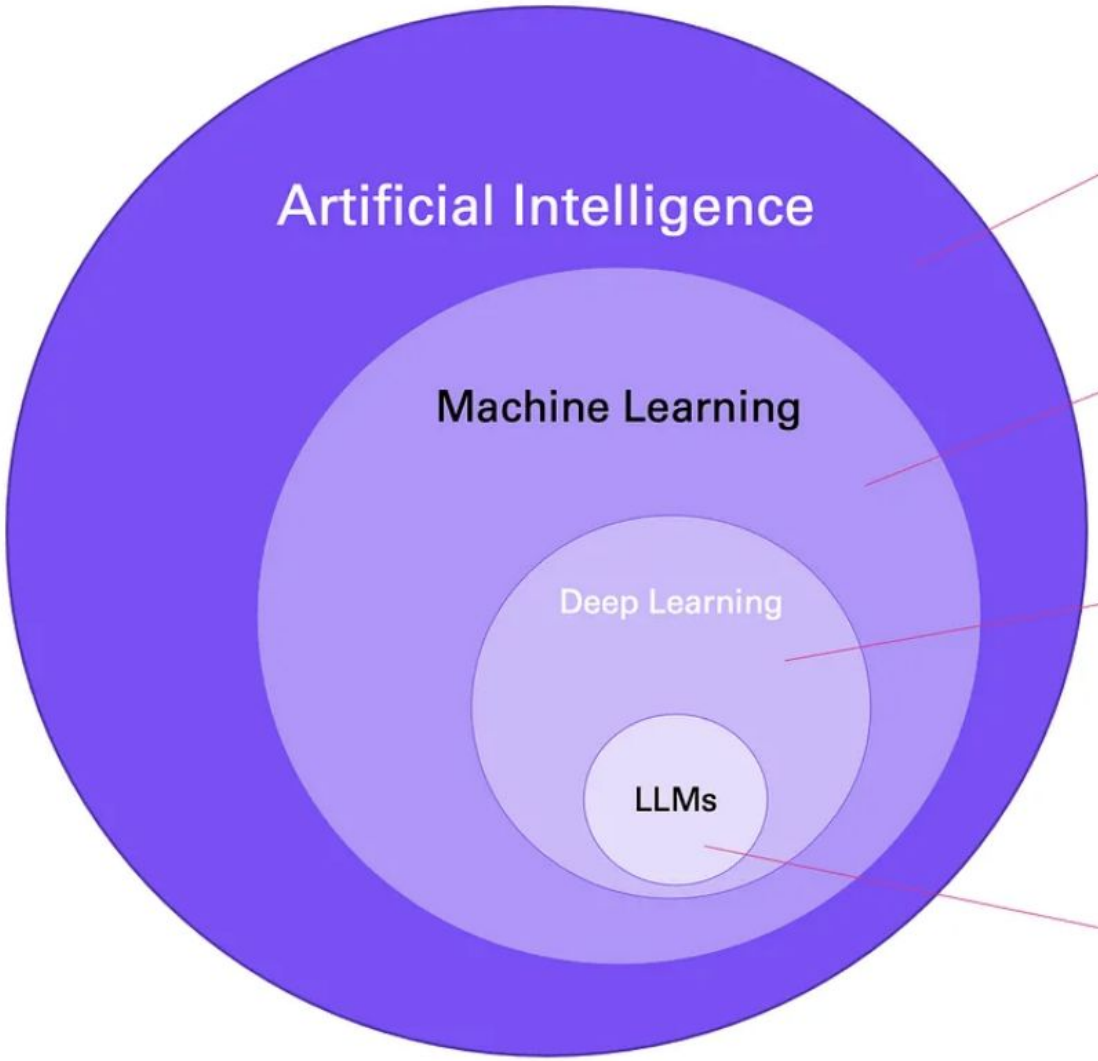
Participants with laptops can engage in hands-on activities.

Agenda

- Introductions - 5 minutes
- Exploring Large Learning Models (LLMs) - 10 min
- Emergence of Code LLMs - 10 min
- Case Study
  - Using LLMs in a Programming Class - 15 min
- Conclusions - 5 minutes

Introductions!

Exploring Large Language Models (LLMs)



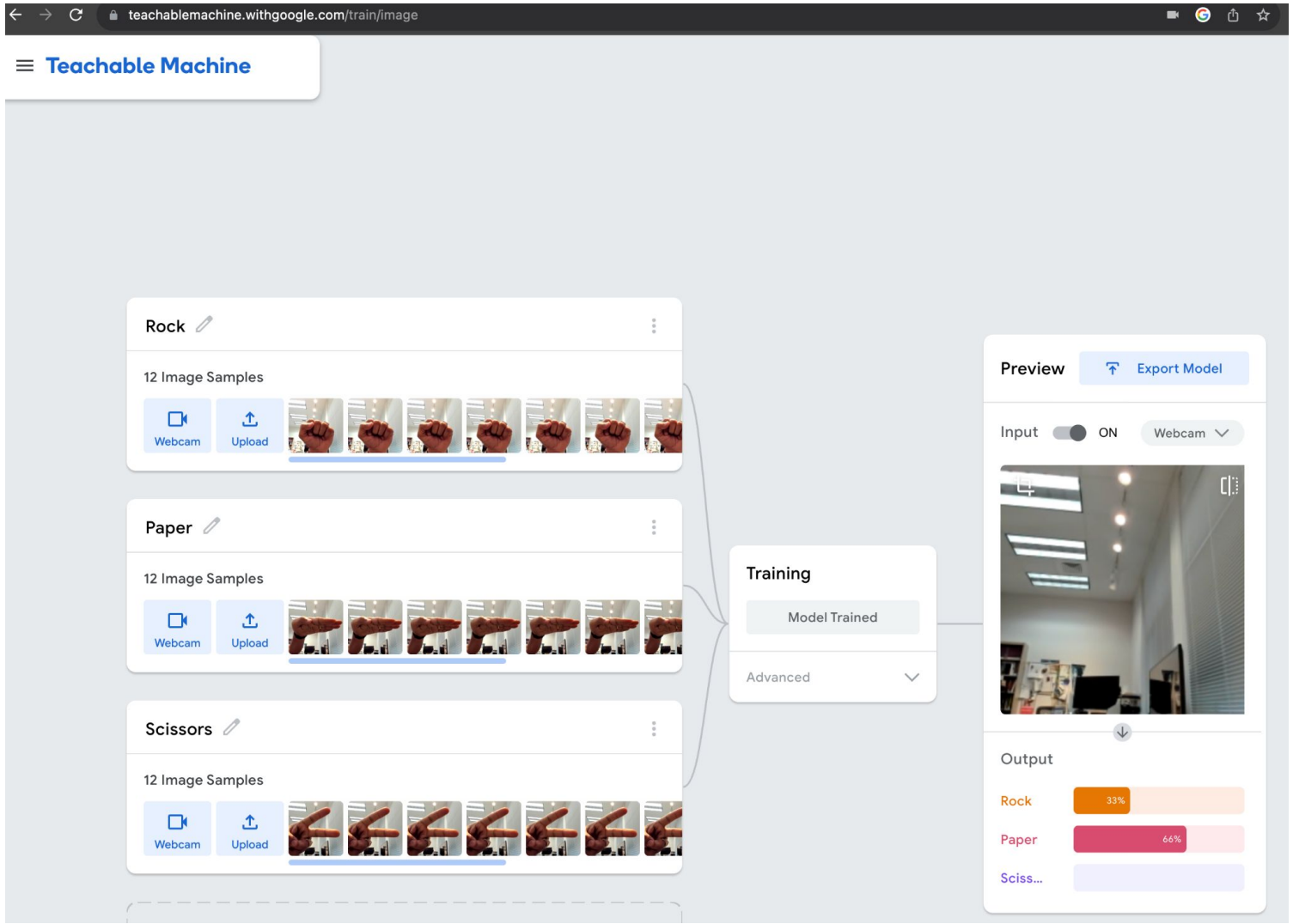
Intelligent Machines
Broadly defined

Pattern Recognition
Learning general patterns from data

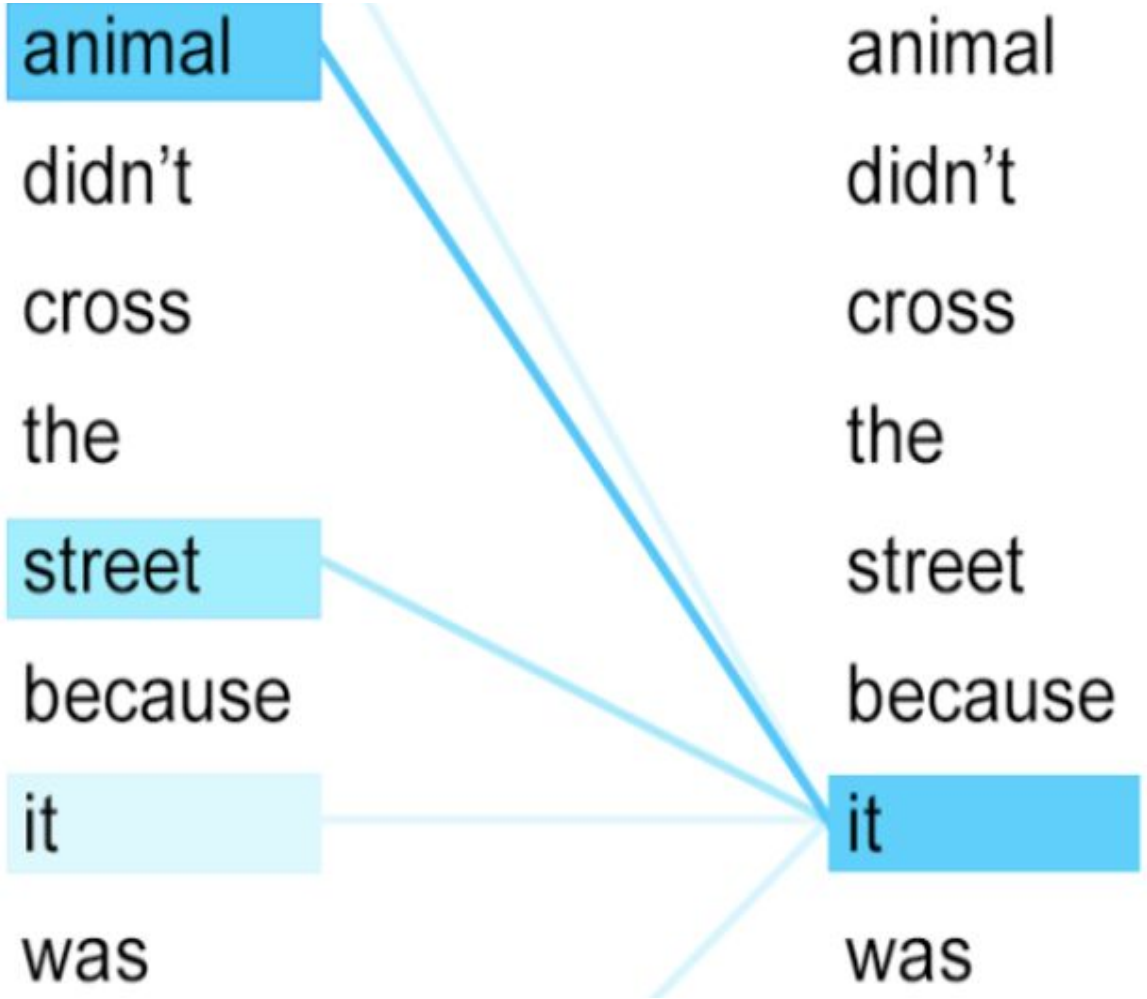
Neural Networks
Learning general patterns in **unstructured** data (i.e. images, text, audio, etc.)

Large Language Models
Learning to understand natural language (i.e. text)

Demo - Machine Learning - Teachable Machine Demo



https://teachablemachine.withgoogle.com/train/image



Attention Is All You Need paper on Transformers, Vasvani et al. (2017)

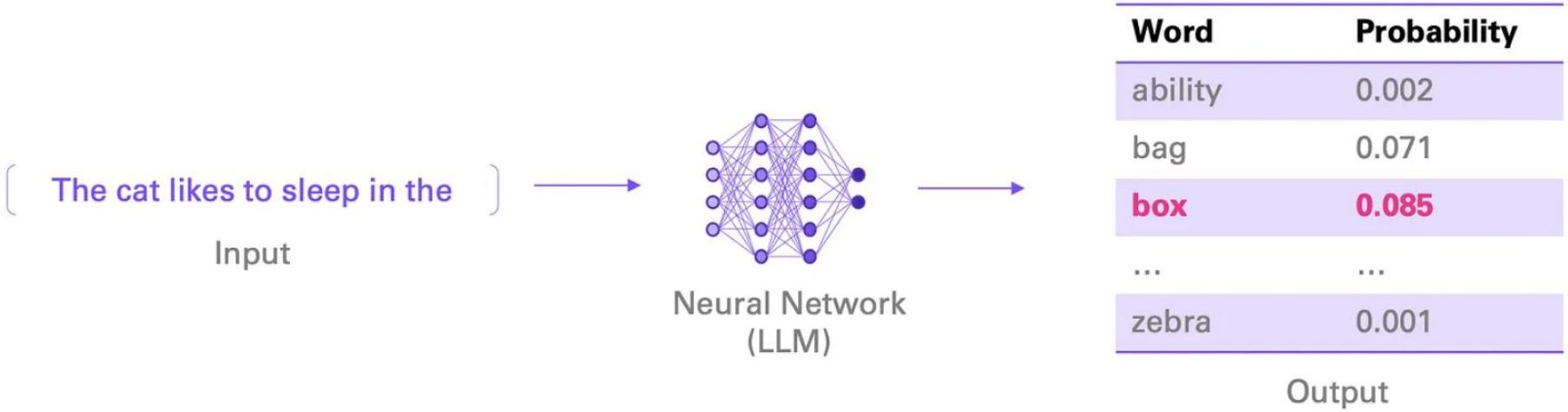
Language modeling

Imagine the following task: Predict the next word in a sequence

The cat likes to sleep in the ____ → What **word** comes next?

Now we have (say) ~50,000 classes (i.e. words)

Can we frame this as a ML problem? Yes, it's a classification task.



Language modeling is learning to predict the next word.

Source: Stouffelbauer, 2023

Massive training data

We can create vast amounts of sequences for training a language model

● Context ● Next Word ● Ignored

The cat likes to sleep in the

The cat likes to sleep in the

The cat likes to sleep in the

The cat likes to sleep in the

The cat likes to sleep in the

We do the same with much **longer sequences**. For example:

A language model is a probability distribution over sequences of words. [...] Given any sequence of words, the model predicts the next ...

Or also with **code**:

```
def square(number):
  """Calculates the square of a number."""
  return number ** 2
```

And as a result - the model becomes incredibly good at predicting the next word in any sequence.

Massive amounts of traning data can be created relatively easily.

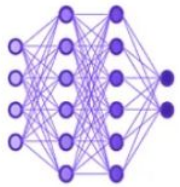
Source: Stouffelbauer, 2023

Natural language generation

After training: We can generate text by predicting one word at a time

A trained language model can

Input



LLM

Word	Probability
speak	0.065
generate	**0.072**
politics	0.001
…	…
walk	0.003

Output at step 1

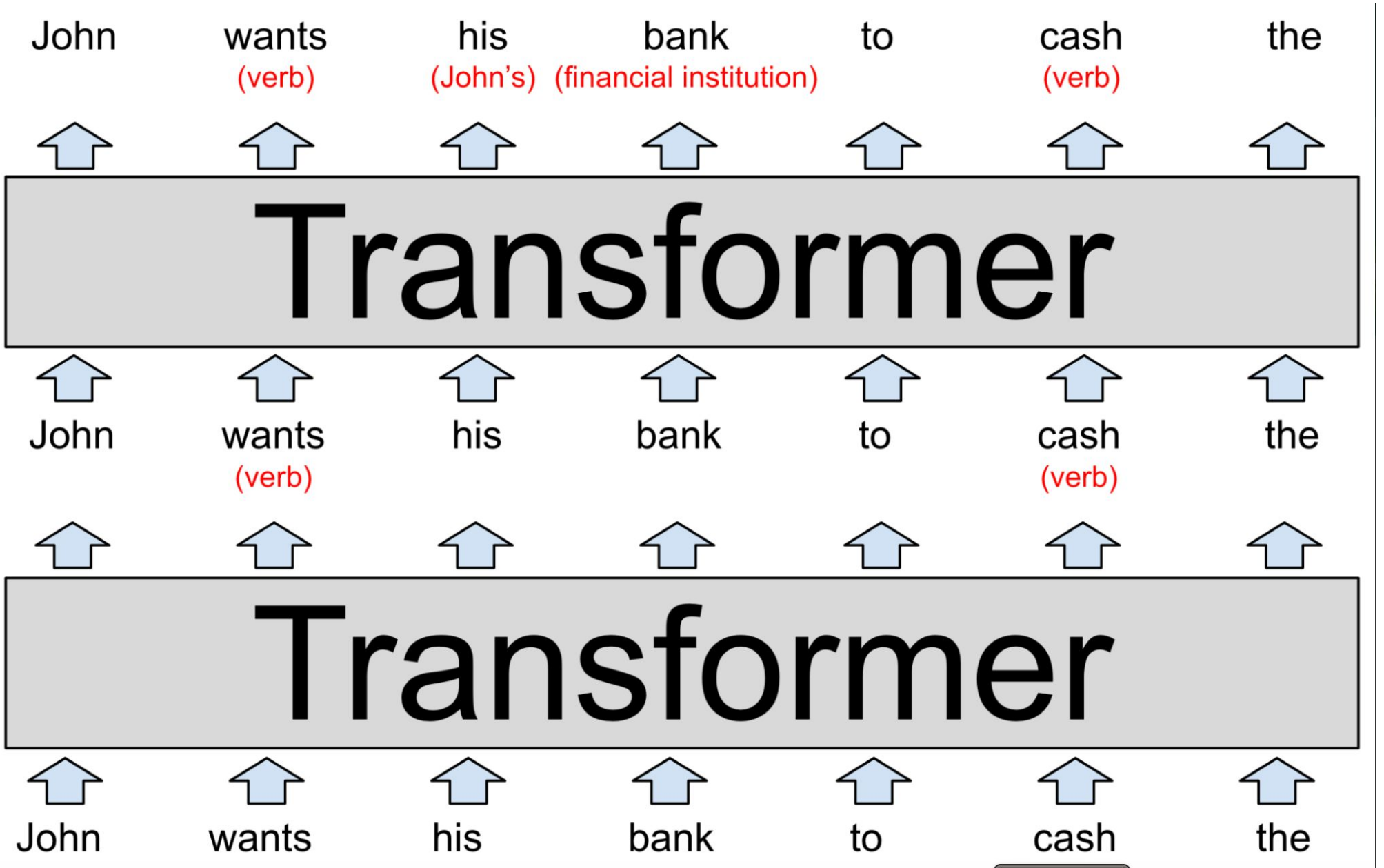
Word	Probability
ability	0.002
text	0.084
coherent	**0.085**
…	…
ideas	0.041

Output at step 2

LLMs are an example of what's called "Generative AI"

Source: Stouffelbauer, 2023

Transformer (the T in GPT) -> word vectors into predictions



What does **Generative Pre-trained Transformer (GPT)** mean

Generative

Means "**next word prediction.**"

As just described.

Pre-trained

The LLM is pretrained on massive amounts of text from the internet and other sources.

Transformer

The neural network architecture used (introduced in 2017).

Source: Stouffelbauer, 2023

Phases of training LLMs (GPT-3 & 4)

1. Pretraining

Massive amounts of data from the internet + books + etc.

Question: What is the problem with that?

Answer: We get a model that can babble on about anything, but it's probably not **aligned** with what we want it to do.

2. Instruction Fine-tuning

Teaching the model to respond to instructions.

Model learns to respond to instructions.

→ Helps **alignment**

"Alignment" is a hugely important research topic

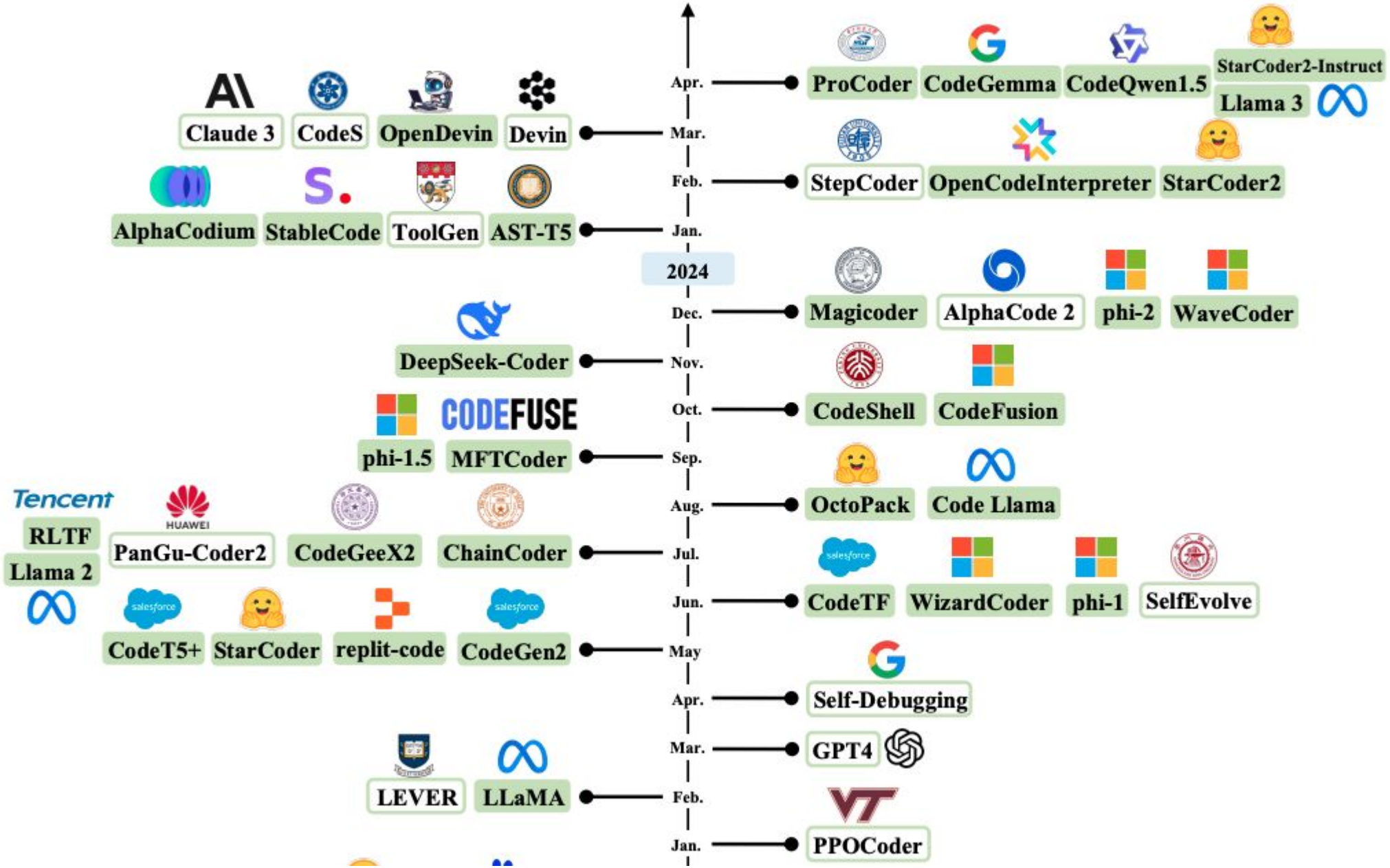
3. Reinforcement Learning from Human Feedback

Similar purpose to instruction tuning.

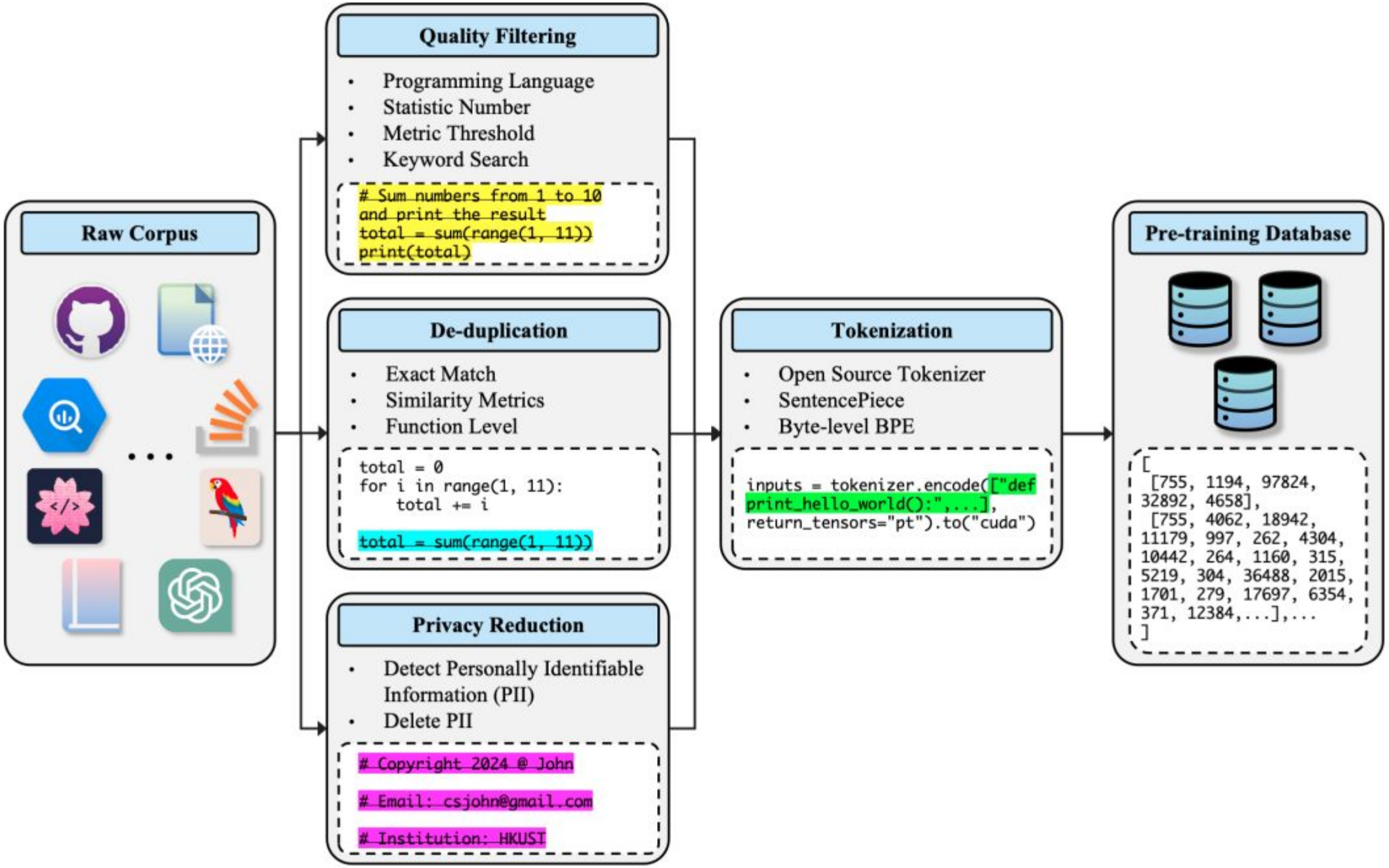
Helps produce output that is closer to what humans want or like.

Source: Stouffelbauer, 2023

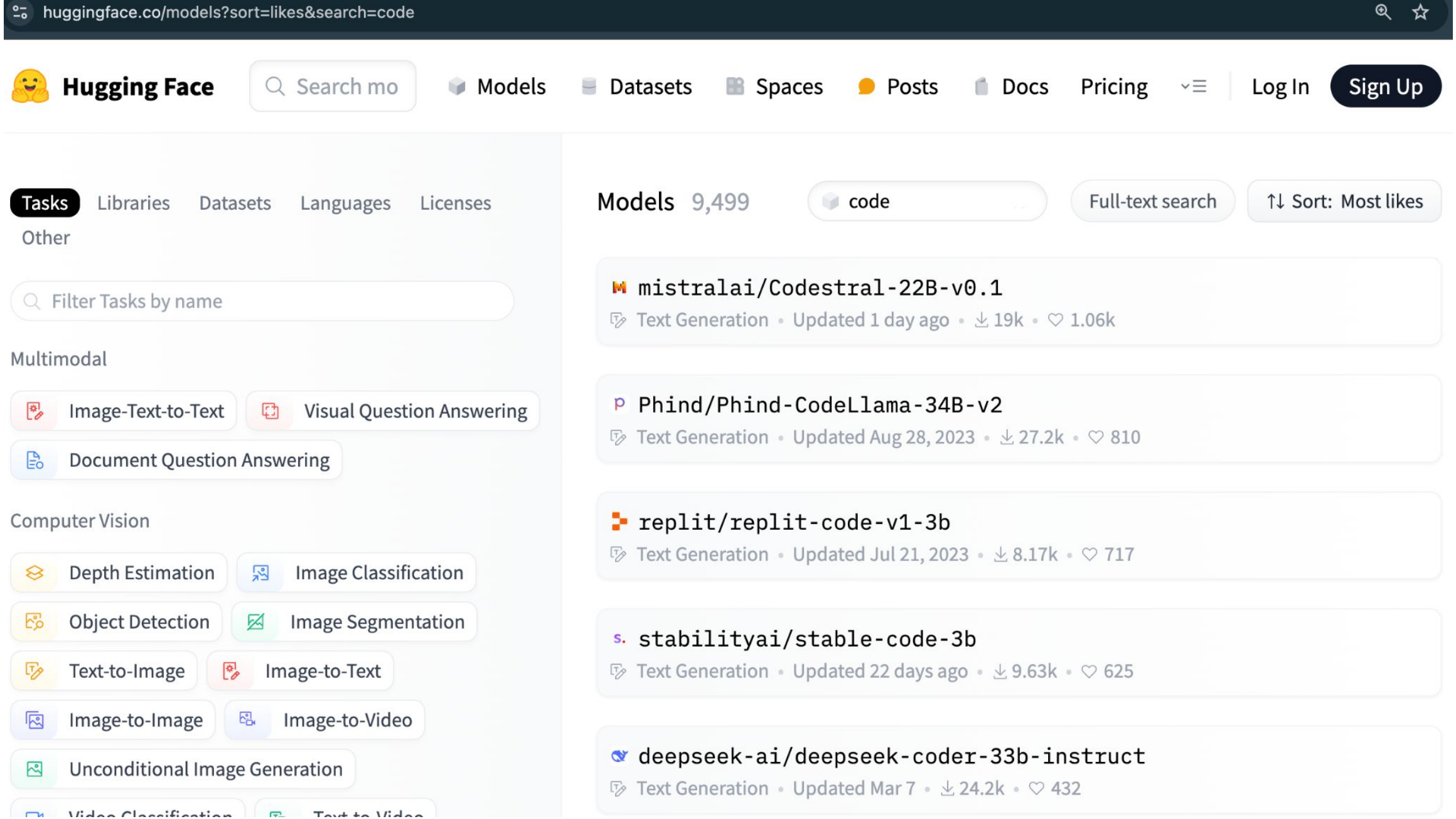
Emergence of Code LLMs



Source: Survey of LLMs for Code Generation - link



Source: Survey of LLMs for Code Generation - [link]



Source: https://huggingface.co/models?sort=likes&search=code

Model Description

`replit-code-v1-3b` is a 2.7B Causal Language Model focused on **Code Completion**. The model has been trained on a subset of the Stack Dedup v1.2 dataset.

The training mixture includes **20 different languages**, listed here in descending order of number of tokens:

`Markdown`, `Java`, `JavaScript`, `Python`, `TypeScript`, `PHP`, `SQL`, `JSX`, `reStructuredText`, `Rust`, `C`, `CSS`, `Go`, `C++`, `HTML`, `Vue`, `Ruby`, `Jupyter Notebook`, `R`, `Shell`

In total, the training dataset contains 175B tokens, which were repeated over 3 epochs -- in total, `replit-code-v1-3b` has been trained on **525B** tokens (~195 tokens per parameter).

Source- https://huggingface.co/replit/replit-code-v1-3b

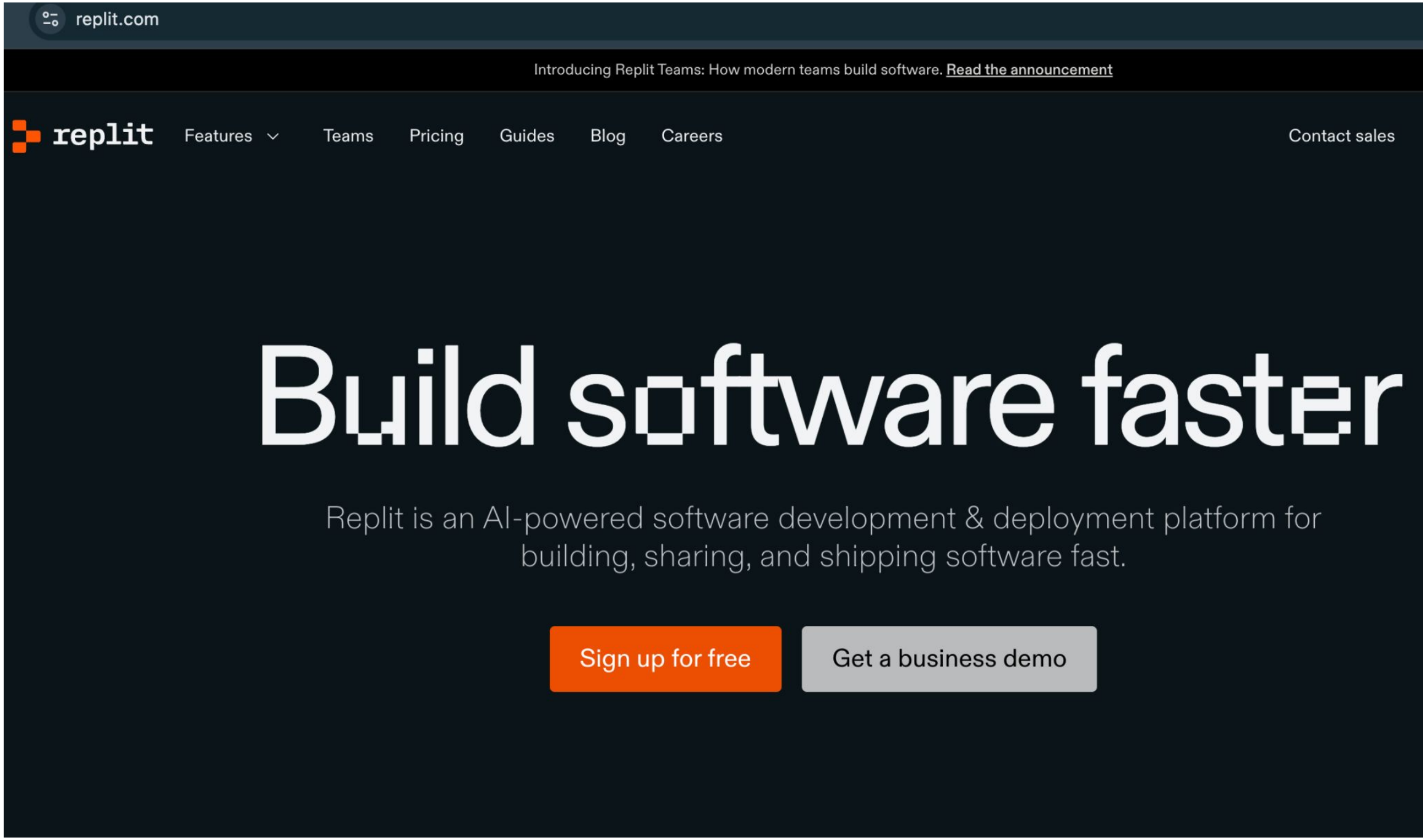
Dataset Summary

The Stack contains over 6TB of permissively-licensed source code files covering 358 programming languages. The dataset was created as part of the BigCode Project, an open scientific collaboration working on the responsible development of Large Language Models for Code (Code LLMs). The Stack serves as a pre-training dataset for Code LLMs, i.e., code-generating AI systems which enable the synthesis of programs from natural language descriptions as well as other from code snippets. **This is the near-deduplicated version with 3TB data.**

Supported Tasks and Leaderboards

The Stack is a pre-training dataset for creating code LLMs. Code LLMs can be used for a wide variety of downstream tasks such as code completion from natural language descriptions (HumanEval, MBPP), documentation generation for individual functions (CodeSearchNet), and auto-completion of code snippets (HumanEval-Infilling). However, these downstream evaluation benchmarks are outside the scope of The Stack.

Source: https://huggingface.co/datasets/bigcode/the-stack-dedup


replit.com
Introducing Replit Teams: How modern teams build software. Read the announcement
replit
Features
Teams
Pricing
Guides
Blog
Careers
Contact sales
Build software faster
Replit is an AI-powered software development & deployment platform for building, sharing, and shipping software fast.
Sign up for free
Get a business demo

Case Study : Use of Coding LLM in a Programming Course

ICS 385 - Intro to Web Programming

1. Intermediate web programming course

2. Topics
   - HTML/CSS
   - JavaScript
   - NodeJS and Express
   - EJS Templates
   - GitHub
   - RESTful APIs
   - MongoDB and Mongoose
   - Authentication and Security

3. Two student TAs
   - Took course last spring without AI

Insert in Syllabus...

Use of Artificial Intelligence (AI)

- AI tools, such as coding generators from Replit, Githib Copilot, and other similar tools are welcome to be used in all classroom and course assignments, unless when specifically prohibited from doing so. Students are expected to **cite and attribute their use of GenAI tools**, just like any other reference tool or source, when creating their homework or exam assignments.

Search & run commands ⌘ K

- .config
- public
- views
- date.js
- index.js

Packager files

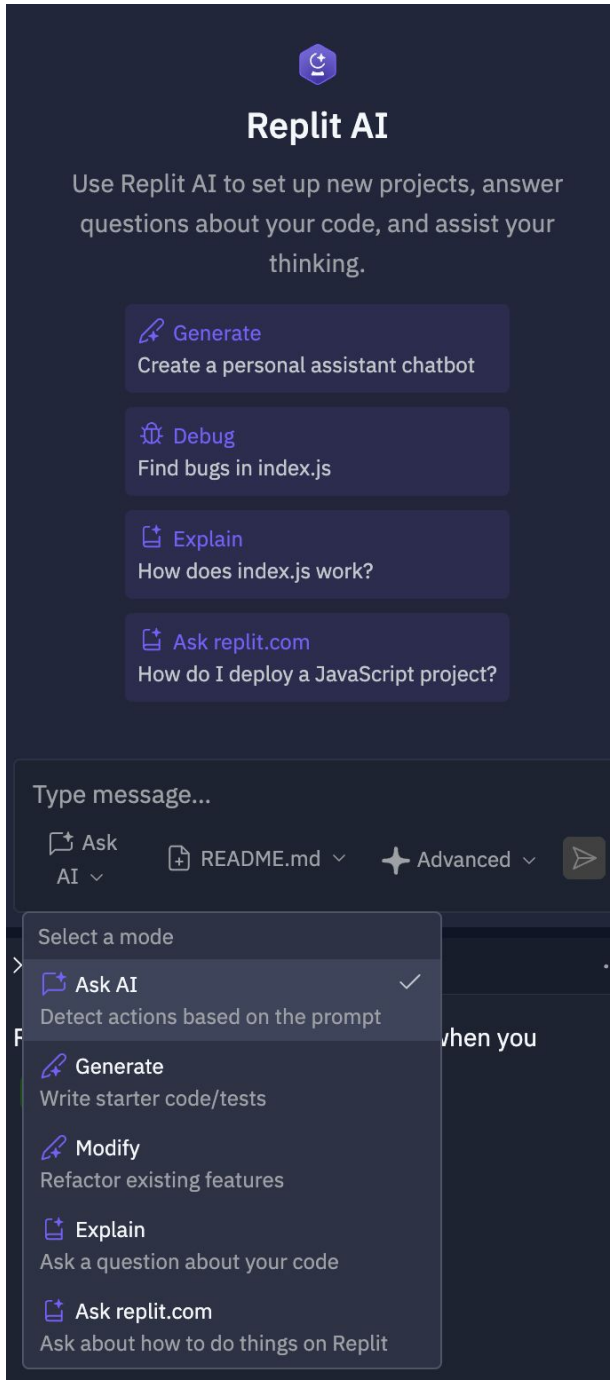
- package-lock.json
- package.json

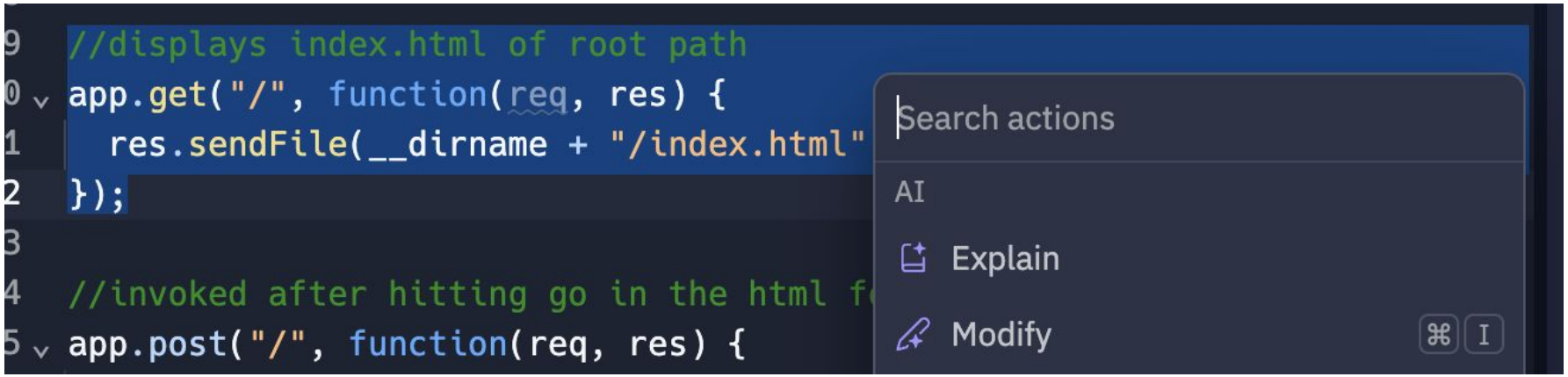
Config files

- .replit
- replit.nix

index.js

```
// This Express.js application creates two separate to-do lists, a fun to-do list and a weekend to-do list, each with pre-set arrays containing 2-3 items. The current date and a default to-do list with general items are shown on the main route ("/"). Users have the option of adding items to the work list (which is forwarded to the "/work" route) or the default list. Also, routes "/fun" and "/weekend" show listings designed for weekend chores and fun things to do.

// Preston Respicio
// 2/10/2024

const express = require("express");
const bodyParser = require("body-parser");

// create a date object that requires the date.js file
const date = require(__dirname + "/date.js");

const app = express();

// set an array for the default items in the list
let items = ["Wake up", "Brush teeth", "Get ready", "Start and warm up car", "Buy foods and drinks"];
// set an empty array for new work items
let workItems = ["Clock-in", "Curbside check-in service", "Wheelchair assistance service"];
// set an array for Fun and another for Weekend
let funItems = ["Watch anime episodes", "Play video games", "Walk the dog"];
```





AI just now Explain Advanced

This piece of code in the `index.js` file defines how your application responds when someone visits the root URL (`/`) of your website. Let's break it down into simpler parts:

1. `app.get("/", function(req, res) {...});` : This tells your web server (created with Express, a web framework for Node.js) to listen for GET requests on the root URL (`/`). A GET request is what browsers typically send when a user tries to visit a webpage.
2. `function(req, res) {...}` : This is a callback function that gets executed when a user hits the root URL. It takes two arguments:
   - `req` (short for request): This object contains all the information about the request made to the server, like any data the user sends.
   - `res` (short for response): This object is used to send back a response to the user. You fill it with what you want the user to receive.
3. `res.sendFile(__dirname + "/index.html")` : Inside the callback function, `res.sendFile` is used to send a file back to the user's browser. `__dirname` is a built-in variable in Node.js that gives you the absolute path of the directory containing the current executing script. So, `__dirname + "/index.html"` creates a path to the `index.html` file located in the same directory as your `index.js` script.

In summary, this piece of code serves the `index.html` page to users when they visit the root (`/`) of your website. It's like telling your web server, "Whenever someone comes to the entrance (`/`), show them the `index.html` page."

```
// Generate code to compute the volume of a sphere
```

Generate code to compute the volume of a sphere

Generate | Reject | Accept

```
// Function to compute the volume of a sphere given its radius
function computeSphereVolume(radius) {
  return (4/3) * Math.PI * Math.pow(radius, 3);
}
```

Write test scripts to test above code to compute the volume of a sphere

Generate | Reject | Accept

```
// Test script for computeSphereVolume function
function testComputeSphereVolume() {
  const radius = 5; // Example radius
  const expectedVolume = (4/3) * Math.PI * Math.pow(radius, 3);
  const calculatedVolume = computeSphereVolume(radius);

  if (calculatedVolume === expectedVolume) {
    console.log(`Test Passed: Volume for radius ${radius} is correctly
calculated as ${calculatedVolume}.`);
  } else {
    console.error(`Test Failed: Expected volume ${expectedVolume}, but
got ${calculatedVolume}.`);
  }
}

testComputeSphereVolume();
```

Conclusions!

Students

Pros

Design, write pseudocode

Explain code snippets

Generate Code, sometimes unfamiliar!

Debug code, explain bugs

Create Auto Test Scripts

Helps in documentation

Document AI section!

Cons

Generate Code, sometimes unfamiliar!

Code works but students don’t how...

Trial and Error Approach

Difficulty in Explaining Code

Students who have less programming experience have more difficulty with AI

Instructor and TAs

Pros

Provides another point of view for students to learn

Students can generate working code easier

Helps students who are unfamiliar with setup and installation over Zoom

Students can turn to help for LLMs before asking/emails

Students can focus on design and tests more than before!

Cons

Over reliance on LLMs vs. learning by writing code

Beginners take shortcuts

Cheating

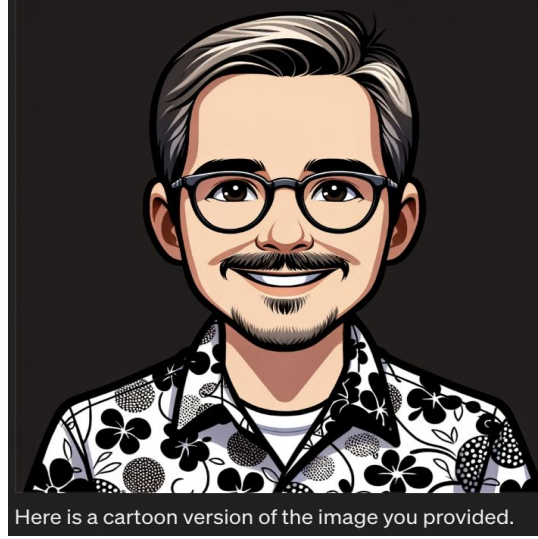
- Homework Assignments
- Exams

Harder to give partial credit

Unsure what students did vs. done by code generation!

Open book exams now have *images embedded in PDFs*

Questions?
Comments…
Discussions!
debasisb@hawaii.edu
maui.hawaii.edu/cybersecurity



References

Anthropic (2024). *Claude chatbot*. https://www.anthropic.com/

Chan, C.K.Y and Hu, W. (2023). *Students' Voices on Generative AI: Perceptions, Benefits, and Challenges in Higher Education*. Link.

Chan, C.K.Y and Lee, K. (2023). *The AI generation gap: Are Gen Z students more interested in adopting generative AI such as ChatGPT in teaching and learning than their Gen X and Millennial Generation teachers?* Link.

Entsminger, J., Esposito, M., Tse, T., and Jean, A. (2023). The Dark Side of Generative AI: Automating Inequality by Design. California Review Management. Link.

EU AI Act (2023). *Home Page*. Link.

Google (2024). *Gemini chatbot*. https://gemini.google.com/

References

Jiang, J. et al. (2024). A Survey of Large Language Models for Code Creation. https://arxiv.org/abs/2406.00515

Heaven, W.D. (January/February, 2024). *6 big questions for generative AI, MIT Technology Review*. Vol. 127(1), 31-37. Link.

Nightshade (2024). Home Page. https://nightshade.cs.uchicago.edu/whatis.html

Open AI (2024). *ChatGPT*. https://openai.com/

Stoffelbauer, A. (2023). *How Large Language Models Work.* Medium. Link.

University of Hawaii AI (2024). *Generative AI*. https://www.uhonline.hawaii.edu/ai

Vaswani, A. (2017). *Attention is all you need*. Link.

White House (2023). *Biden AI Executive Order*. Link