AI For Coding

Yasen Kiprov

About me

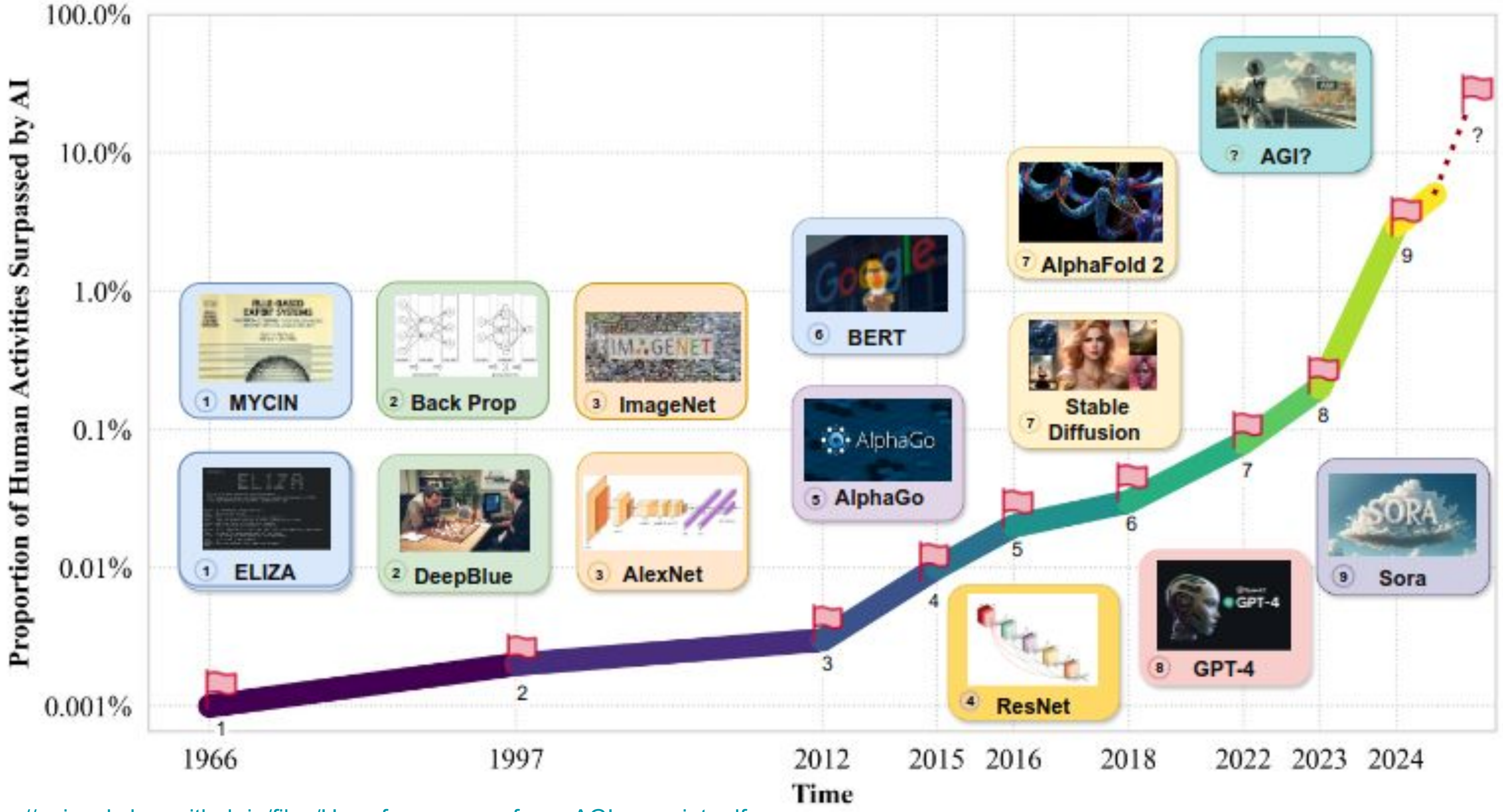
- Over 15 years in the field, AI, NLP
- AI team lead at SiteGround
- Co-organizer of PyData Sofia meetups

Interests: Education, Applied AI

Agenda

- Introduction
- AI for code generation
    - Applications
- AI assistants in your IDE
- Exercise

LLMs are Disruptive



Источник: https://agiworkshop.github.io/files/How_far_are_we_from_AGI_preprint.pdf

GPT History

- Attention Is All You Need (2017) - transformer
- BERT (2018) - training
- GPT-3 (2019) - scaling

Next word prediction

The cat

ChatGPT

- GPT tuned on dialogues, RLHF
- Dialogue:
  - System Prompt
  - Prompt
  - Completion

LLM Specifics

- Replies fast
- Writes thoroughly
- Can reply to everything
- Has tons of information
    - (Up to ~~2021~~ ~~2023~~ 2024)

- Bad critical thinker
- Easily misled
- Hallucinates

Does not realise what it’s saying

How to Write Prompts

- Command / question
- Context
  - Role of the AI assistant
  - Domain
  - Extra information
- Format
  - Length
  - Style / wording

How To Treat the Assistant

- Assistant / Intern
  - Give it tasks
  - Micro-manage
- Peer
  - Discuss
  - Pair with it
- Expert
  - Consult with it

Commercial Applications

- chatgpt.com
  - GPT 4o
- gemini.google.com
- claude.ai
  - Sonnet 3.5
- bing.com/chat
  - GPT 4 in Creative mode

Github Copilot

Open Source Models

- Llama Family, Codellama
- Mistral Family, Codestral
- DeepSeek Coder 2 (latest and greatest)
- OpenCode
- CodeQwen
- …

https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard

AI For Code Generation

Code Generation

- Similar to text generation
- Context is crucial
- Evaluation is different
  - Can cause damage
- State of art: GPT-4o
  - Open source models are close
- Side note: gamma.app

Pros and Cons

- Works surprisingly well
- Understands complex code instantly(demo)
- Takes huge contexts
- 55% increase in speed

- Decrease of quality
- Decrease of code reuse
- 30% of code is accepted as is
- Sometimes knowledge is not up to date

Dangerously misleads developers

Source: https://dev.to/jesterxl/github-copilot-research-finds-downward-pressure-on-code-quality-4m87

Prompt Engineering

- System instructions
- User message
- Context is important
    - Large context is OK
    - Too large context may be a problem

Concept: Micro-manage your bot

System instructions

- Persona

"You are an experienced python developer. Your job is to answer questions and write valid and concise code according to user's requirements."

- Generic Context

"You know the WordPress REST API. You can make direct API calls using …"

Prompt techniques

- Ask GPT for advice
- **Iterate**
- Be direct, write explicitly
- Describe desired outcome
  - Output format
- Use Chain of Thought

AI for code generation

- Get ideas / learn
- Debug
- Write “auxiliary code”
- Write one time scripts
- Write code/test/docs faster

Don’t trust chatGPT

Get Ideas / Learn

- Instant Stack Overflow
- Like Google (recently)

➢ How to do something

➢ What tools/libs to use

➢ Where to start

➢ Write some pseudocode

➢ What to look out for

Get Ideas / Learn

- Provide enough context
- Ask open questions
- Steer the dialogue
- Ask for sources and more information
- Ask for multiple opinions
- Ask to self-reflect

Debug / Understand Code

- Saves time
  - Helps when blocked
- Prompt structure:
  - Context -> code -> question
- Iterate to reach the right question
- Side effect - get refactoring ideas

“What does this code do”

“What is the problem with this code”

“This code … gives me this error …”

Debug / Understand Code

- Paste enough context
  - Paste code
  - Describe what’s missing
- Paste error without other instructions
- Steer away from impossible / improbable solutions

Write “Auxiliary Code”

- Not part of a large codebase
- Not your usual environment / language
- Easy to test
- Not critical code
  - Don’t bother for quality, maintainability, etc.
- SQL Queries
  - Provide schema as context

Write “Auxiliary Code”

- Instructions are crucial
- Provide code context, libraries, imports
- Write pseudo-code and TODOs
    - When you don’t know
    - When code is complex
    - When code is long & boring

Write One-time Scripts

- Disposable code
- Context - the input and desired output
- Iterate / run the scripts quickly
- Translate code from other languages
- (GPT-4o writes it by itself)

Write Code / Tests / Docs

- Context is crucial
- Prompt is crucial
- Tests and docs are easy
- Integrated code in large project is harder

Write Code / Tests / Docs

- Write smaller (local) chunks
- Automate context
- Save prompts
- IDE plugins:
  - Github Copilot
  - Continue.dev

Security Issues

- Be cautious what goes where
  - Don’t paste without thinking
  - Automate carefully
- Don’t trust ~~OpenAI~~ anyone
  - Critical code
  - Passwords, connection strings, tokens, private keys
  - Client info

AI in the IDE
Continue.dev

Continue

Plugin for **VS Code and JetBrains**

Alternative to **GitHub Copilot**

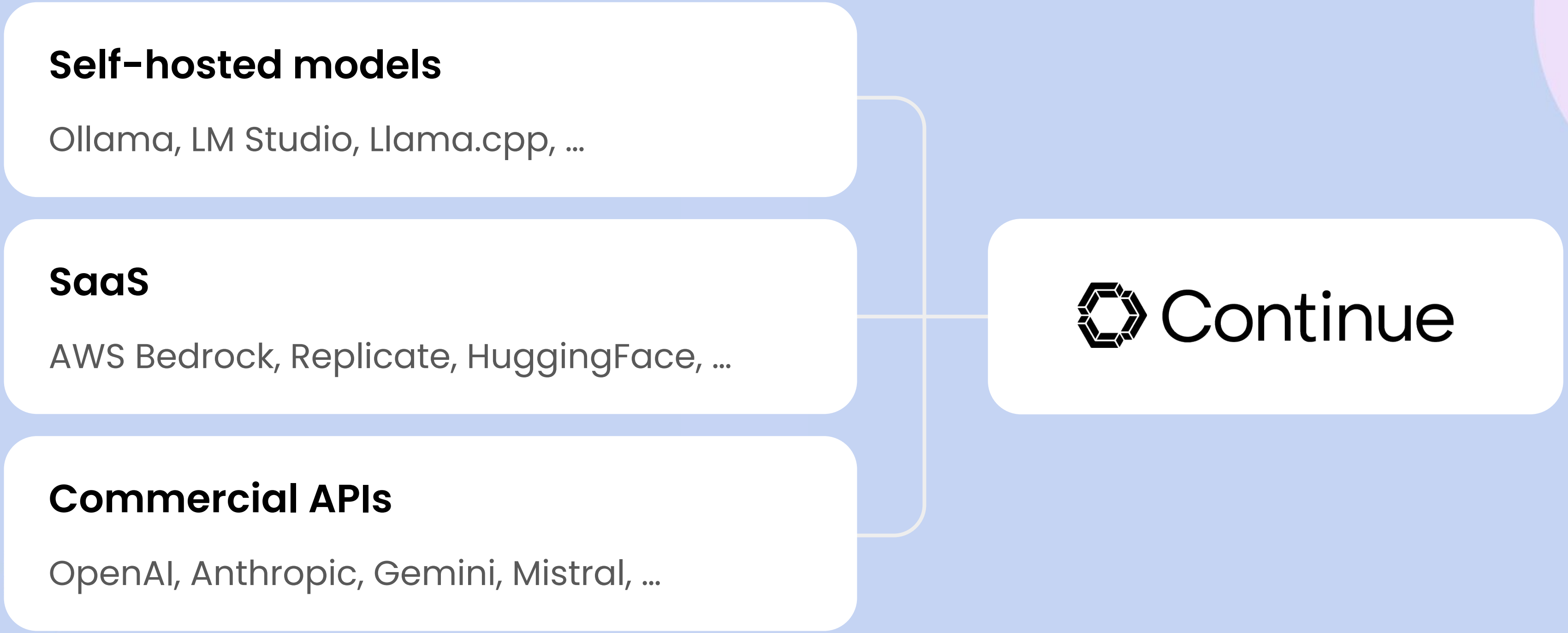
Free & Open

(In Beta for VS Code and Alpha for JetBrains)

Why Continue

- Rich functionality
- Viral community
- Open source
- Extensible

Model Providers


Self-hosted models
Ollama, LM Studio, Llama.cpp, ...
SaaS
AWS Bedrock, Replicate, HuggingFace, ...
Commercial APIs
OpenAI, Anthropic, Gemini, Mistral, ...
Continue

How We Integrate Continue

Continue

Custom config

- no telemetry
- system prompt

Context extensions

WIP: Experiments with **open source models**

Continue Demo

Exercises - ChatGPT and Your Projects

- Discuss ideas and approaches for a problem / task you have.
- Learn how to do something technically on your project.
- “Debug” a script - ask for description of what it does and how it can be improved. Ask for flaws in the code.
- Write something in pseudo-code or code skeleton with TODOs and finish it with chatGPT
- Experiment on the above, providing different contexts.