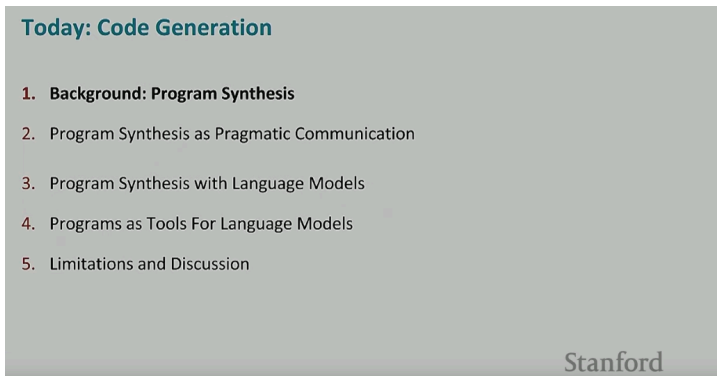
Lecture Notes

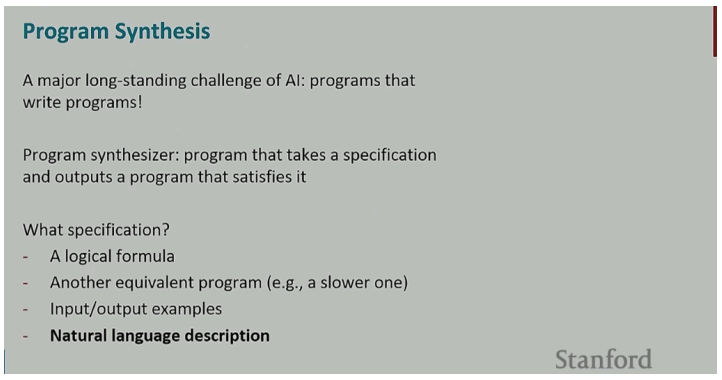
29 August 2024 08:03

Outline:

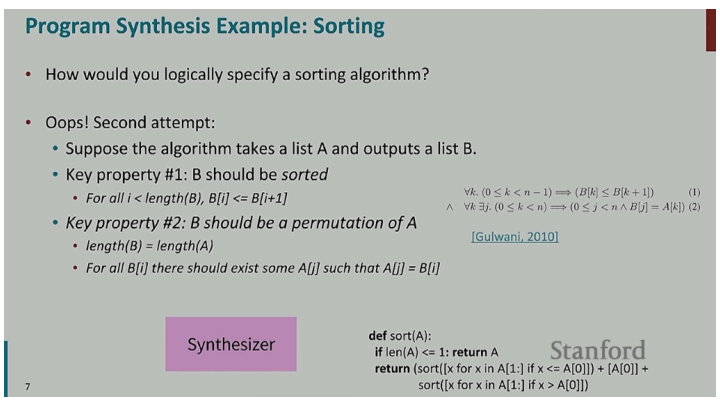


Program Synthesis:

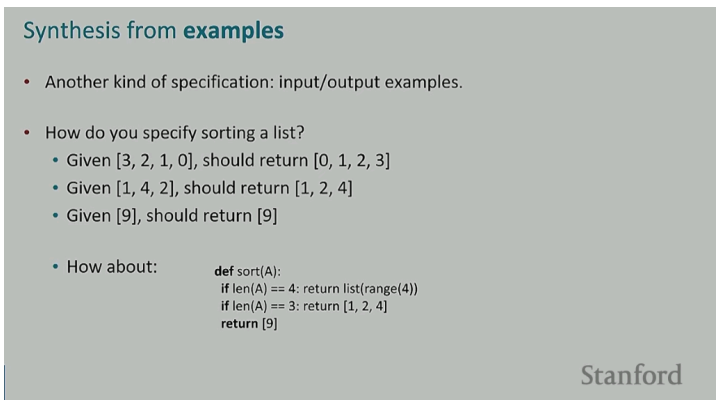
- It only make sense to use a program synthesizer if synthesizing the code is easier and reliable than writing the code ourselves
- Unlike language generation, code synthesis can be actually objectively tested.



- Examples of program synthesis using different inputs:
  - Using logical expression – difficult to specify



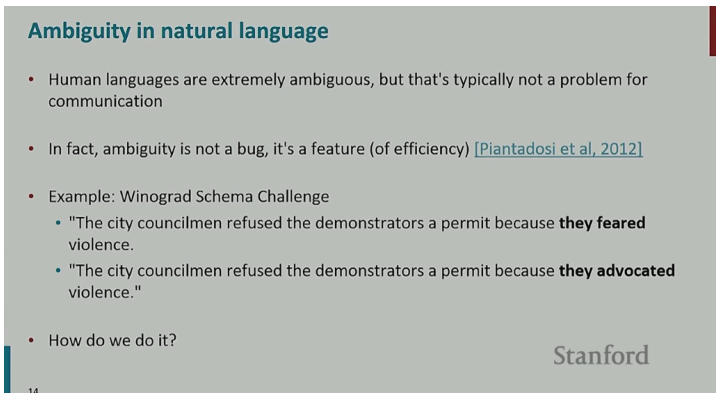
  - Input output examples – may still go wrong.
    - Examples can be ambiguous
    - For an example input – output, there might be N different programms that can fulfil the input – output pattern



Program Synthesis as Pragmatic Communication:

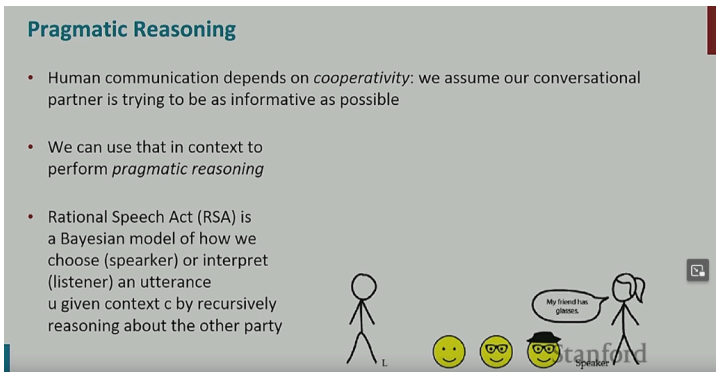
Ambiguity in Natural Languages:

- Try to determine what "they" means In below examples? -- easy for humans but can be difficult for a program

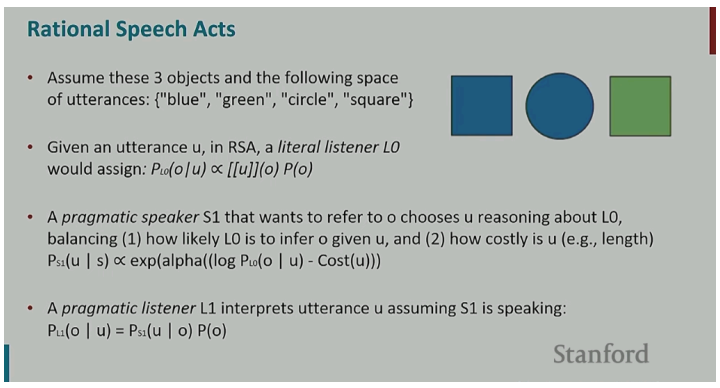


Pragmatic Reasoning:

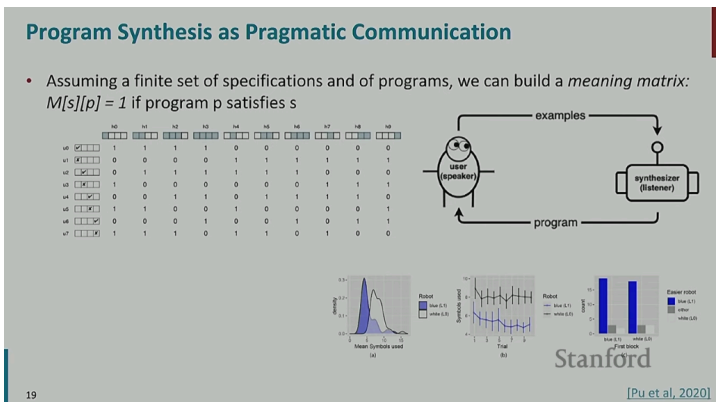
- We (listener) select the smiley in the middle when finding a "friend with glasses", I.e. given a context C (I.e. hat, glasses, not hat/glasses) and utterance U (my friend has glasses)



- Rational Speech Act: Given a utterance U a Listener assigns roughly uniform probability to the choices he has been given, e.g.



- Using Pragmatic communication for program synthesis
  - Assuming finite set of programs and examples that can be given to the synthesizer
  - Works well with finite sets of programs and examples.



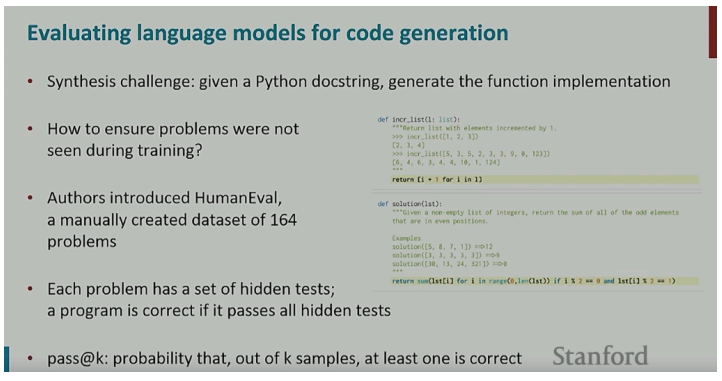
Large Language models can generate Code

- Idea is to use language modelling for code generation.
- LLMs are capable of predicting the next token given the context, so what if we train the GPT type of architectures on code database

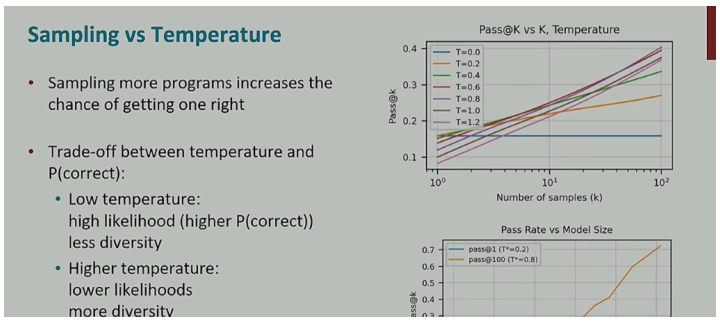
- The initial attempt was made by Github copilot. So What did the dataset look like?
  - Code snippets including comments, docstrings and additional natural language

- The initial efforts was to train a 12B LLM with same architecture of GPT3 and it succeeded

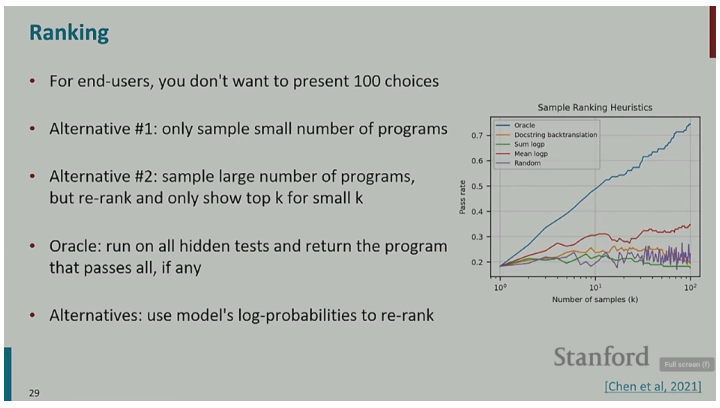
Evaluating the LLM Code generation



- Sampling – is generating N programs and selecting the one which passes the tests

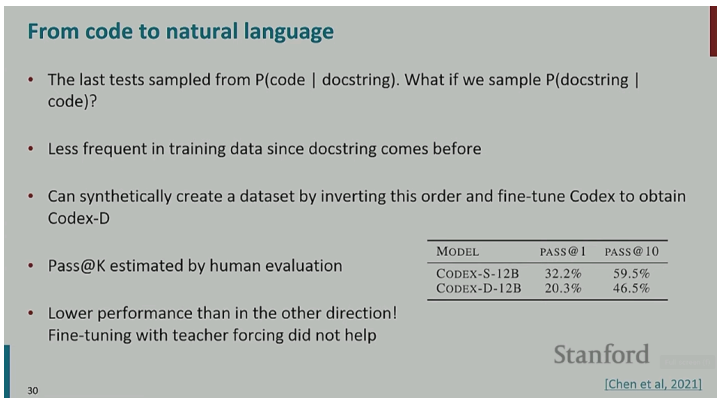


- Ranking – We can sample N programs but can't show user N programs as output. Hence we need some selection criteria

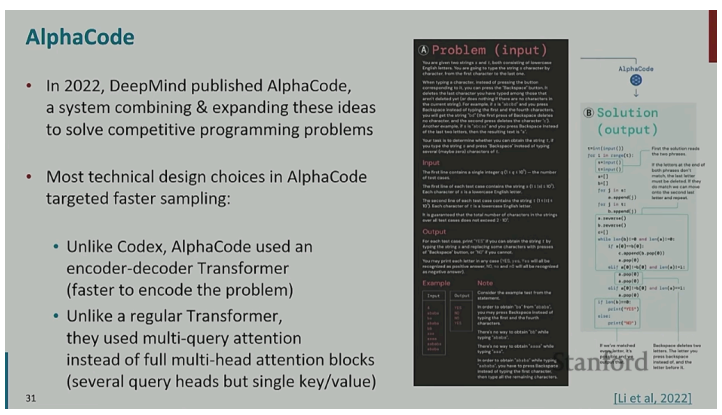


From Code to Natural Language

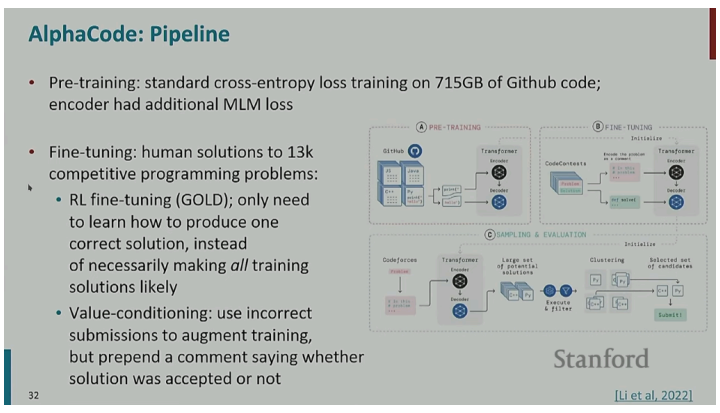
- What if we ask model to generate natural language given the code?
- Codex model was fine-tunedfor such a task and results were worse as compared to P(Code | Natural Language)



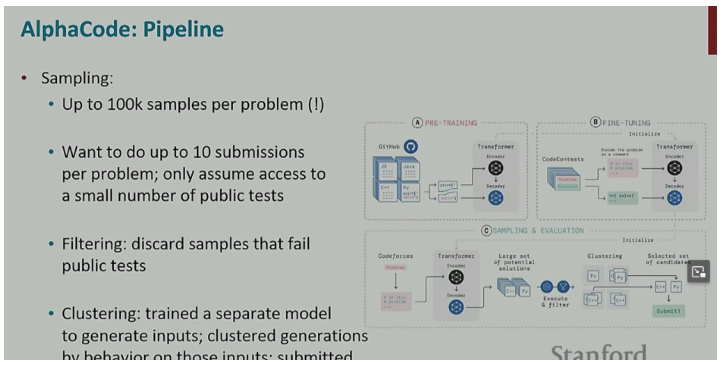
AlphaCode



- Changes from Codex:
  - Used encoder-decoder architecture instead of decoder only
  - Shared key-value heads and only query heads were different across layers
  - Addition of Masked language modelling loss
  - Usage of RL to learn how to produce only one correct solution
  - Inclusion of wrong outputs (with a comment that this ouput was wrong) in the training – called as value-conditioning



- Sampling: Unlike Codex they sampled 100k samples per problem!!!!
- Trained another model to generate inputs for the program to be tested, Grouped the programs based on their outputs



one exemplar from 10 largest clusters

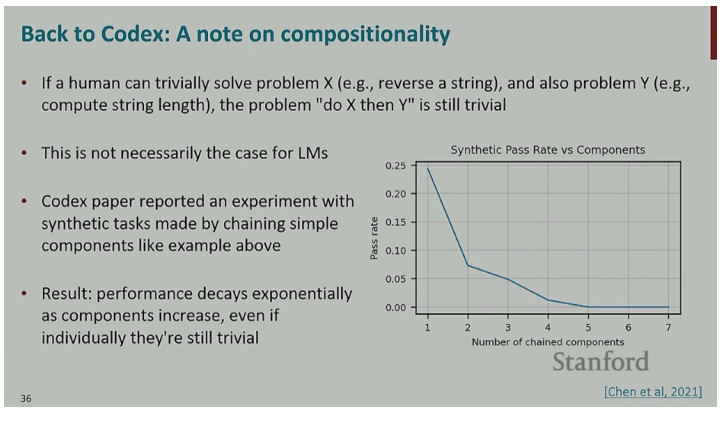
33

[Li et al, 2022]

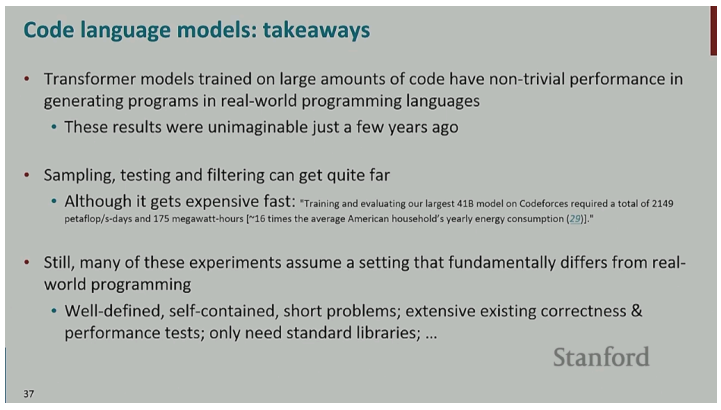
- Simple, if you generate-sample 10x more then you gotta to get few X factor better performance

A note on Compositionality

- Chaining of two simple problems creates difficulties for LLMs even when those problems are solvable for LLMs independently



Takeaway:



Limitations and Discussion: