Lec 17 Code Generation

Task Objective

Generate executable code!

In : specs — types, constraints, natural langs, code context,
images, unit tests, ...
Out: code — completion, implementation, ...

Models : Copilot, Claude 3, GPT4

Challenges

- Strict grammar
- Code semantics
- Executable
- Developed incrementally

Datasets

→ HumanEval with 164 python test examples
- only uses python standard lib

→ CoNaLa / ODEX — scrapped from stackoverflow
└ execution-based eval
- Unit testing tricky for things like plotting

→ ARCADE (data sci notebooks)
- incremental coding
{ Data leak: model does worse on newly created test data

→ SWEBench — GH issue + codebase ⇒ PR

→ Design2Code — website ⇒ HTML/CSS/JS
- need multimodal

Evaluations

▷ Pass@k — generate k samples, does at least one pass?
(has expectation stuff to account for variance)

▷ BLUE, CodeBLUE
+ doesn't require unit tests
+ evaluates style
- may return low score for alternative solution
CodeBLUE : AST overlap, control flow overlap

▷ BERTScore, CodeBERTScore
+ good correlation with execution success & human judgement

▷ Visual Similarity
└ high level visual emb
└ low level element similarity

Models / Methods

▷ LM trained on code
└ try lower temp

→ Code infilling — context should be before & after
but can mask out cursor & pull out after

▷ More Context

→ Copilot Strategy
- relative file path
- language
- 20 mostly recent files in that lang
- similar / imported files
- metadata

▷ Retrieval based code gen
→ Get similar code/documentation via retrieval

▷ Execution Feedback
→ Generate samples, then MBR on execution output

→ InterCode — feed error messages back in

▷ Code synthesis from in-out pairs

→ FlashFill — in MS Excel, tries to find mapping pattern
btwn columns
→ Terpret

* Usually these are done on domain specific langs

Code LMs

▷ Codex — continue training GPT3 on GH data

▷ StarCoder2 — open, by Big Science
└ Mostly LLaMa style
└ Reconfigured for long context
└ Data: the stack, GH issues, PRs, docs, notebooks, LLVM IRs, ...
Preprocess: add file & repo metadata 50% of time
└ Infilling

* The Stack 2 — tries to detect license

▷ Code LLaMa
└ Continued from LLaMa2 but for longer context
└ Code instruction tuning

▷ DeepSeek Coder
└ Preprocess: include dependencies (external libs)