

OpenAPI autocompletion with Large Language Models

The background of the slide features a cosmic scene with a bright, star-like nebula in the center, surrounded by a field of distant stars and a dark, swirling galaxy structure on the right side.

Bohdan Petryshyn

Supervised by Assoc. Professor Mantas Lukoševičius

Terms - Code completion

- Symbol-level autocompletion vs Snippet-level autocompletion

```
7 import http.server
6 import socketserver
5 from http import HTTPStatus
4
3
2 class Handler(http.server.SimpleHTTPRequestHandler):
1 |     def do_GET(self):
8 |         self.send_response(HTTPStatus.OK)
           self.send_header('Content-type', 'text/html')
           self.end_headers()
           self.wfile.write(b'Hello, world!')
1
2 httpd = socketserver.TCPServer(('', 8022), Handler)
3 httpd.serve_forever()
```

Terms - OpenAPI specification

- Definition formats: JSON, YAML
- Definition versions: 2.0, 3.0, 3.1

```
1  openapi: "3.0.0"
2  info:
3    version: 1.0.0
4    title: Swagger Petstore
5    license:
6      name: MIT
7  servers:
8    - url: http://petstore.swagger.io/v1
9  paths:
10    /pets:
11      get:
12        summary: List all pets
13        operationId: listPets
14        tags:
15          - pets
16        parameters:
17          - name: limit...
18        responses:
19          '200':
20            description: A paged array of pets
```

Problem



- Existing code generation solutions demonstrate good performance in common languages like Python. Their performance in more narrow-purpose languages like OpenAPI or GraphQL is less than satisfactory.
- OpenAPI generation can't be evaluated using existing natural language or code generation methods. Developing an evaluation framework would stimulate improvement of solutions in this field.

Existing evaluation methods

- Machine translation metrics:
 - BLEU
 - METEOR
- Levenshtein distance
- Code generation metrics

Evaluation framework. Metrics

- Correctness - how often the generated OpenAPI definition is semantically identical to the original one.
- Validity - how often the generated OpenAPI definition is syntactically valid.
- Speed - how fast the solution generates the OpenAPI definition.

Evaluation framework. Implementation

- Automated evaluation framework
- 100 randomly selected 10-line snippets are masked from 10 definitions
- Semantic OpenAPI difference is considered during evaluation.
 - Invalid result == test failed
 - Significant difference == test failed
 - Insignificant difference:
 - Description changed
 - Example changed (structure is the same)

Evaluation framework - Definitions

- Criterias for definitions for the evaluation framework
 - Most recent definitions
 - Not found in The Stack dataset
 - Human-written
 - 3,000+ lines. Preferring 20,000+ lines
 - Different fields
 - Different companies

Solution prototype. Platforms and tools



- Foundational model - Code Llama from Meta
- Infrastructure platform - Hugging Face Inference Endpoints
- Client-side programming language - JavaScript

Results. Hardware used



Model	GPU	GPU RAM	CPU	RAM	Hourly price (as of the time of writing)
Code Llama 7B	1x Nvidia A10G	24 GB	6 vCPU	28 GB	1.3 USD
Code Llama 13B	4x Nvidia A10G	96 GB	46 vCPU	175 GB	7 USD

Results. GitHub Copilot

Results

- Correctness: **29%**
- Validity: **68%**
- Speed: **19** char/s



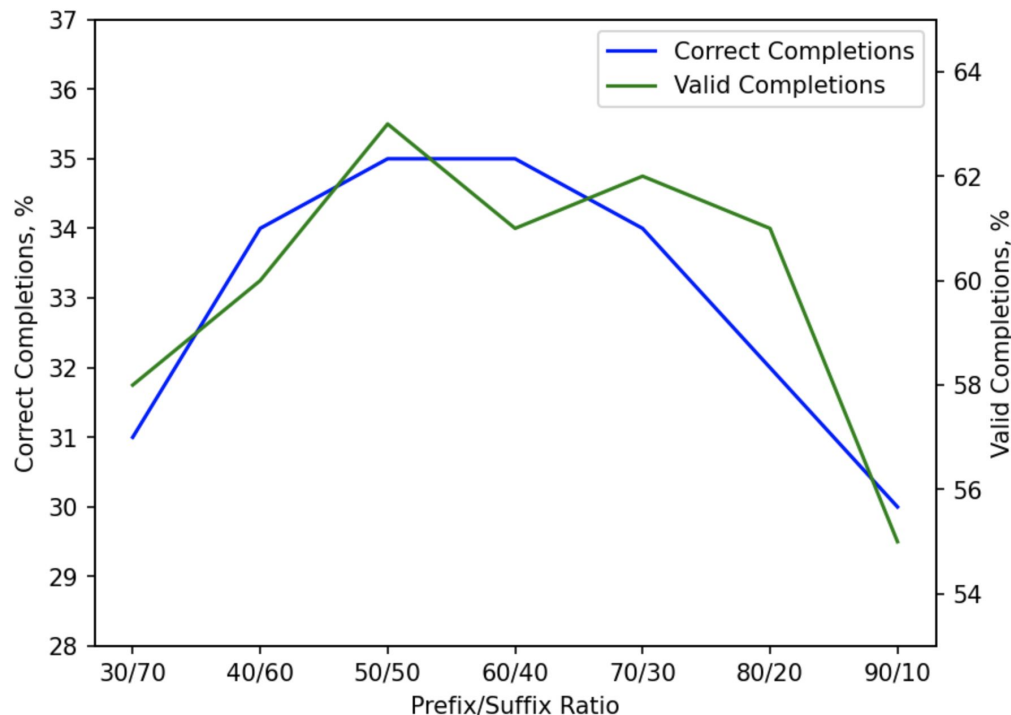
GitHub
Copilot

Results. Optimal prefix-to-suffix ratio

Optimal ratio: 50/50

Results:

- Correctness: **36%**
- Validity: **61%**
- Speed: **81 char/s**



Results. Optimal context size

ktu

1922

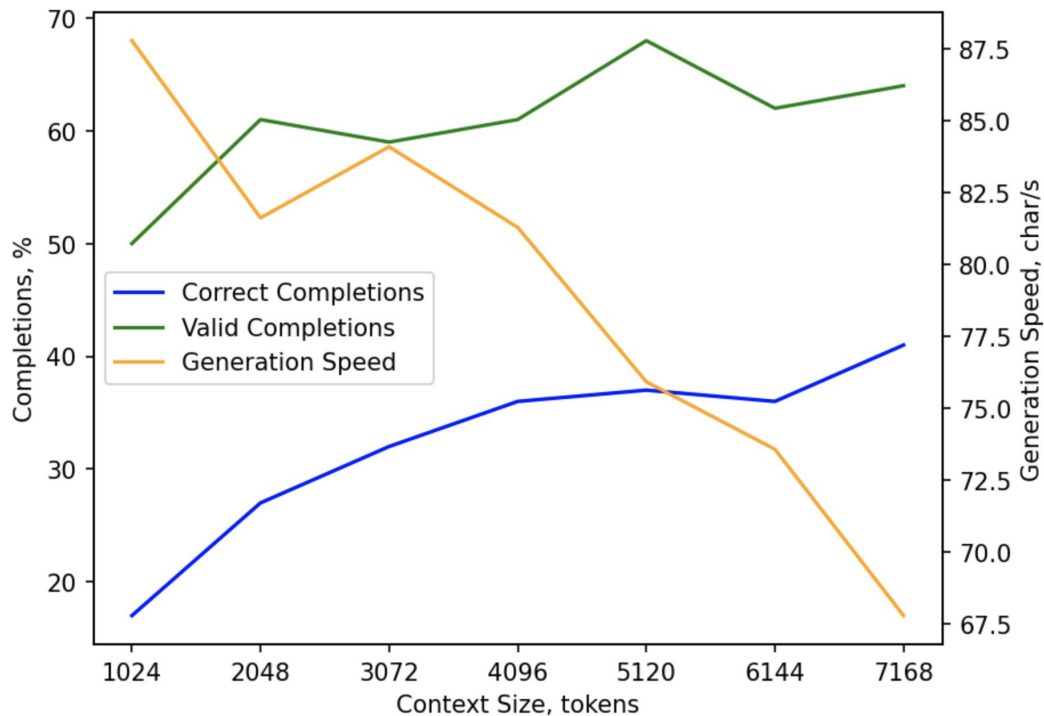
Optimal size: 4096 tokens

Results:

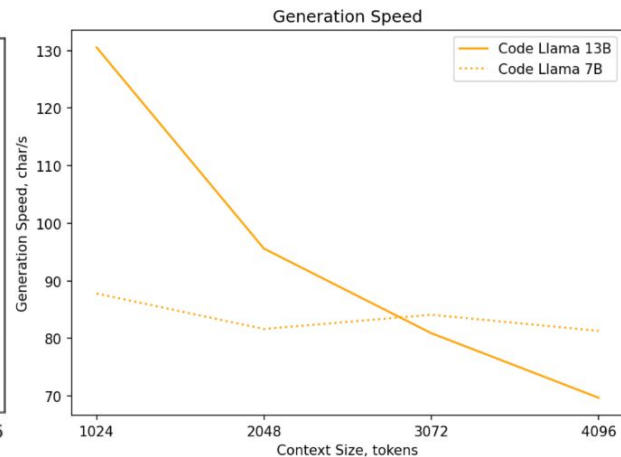
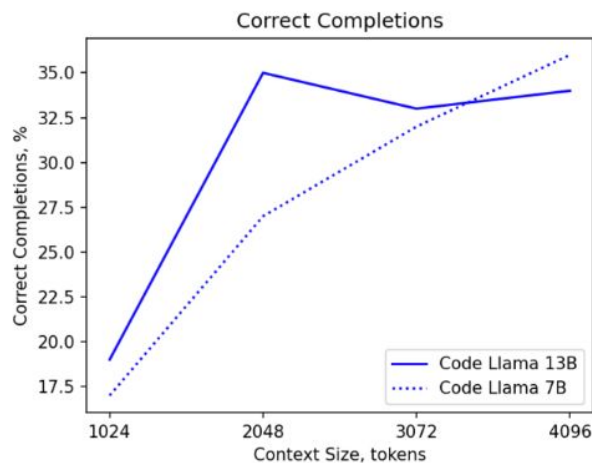
- Correctness: **36%**
- Validity: **61%**
- Speed: **81 char/s**

Best:

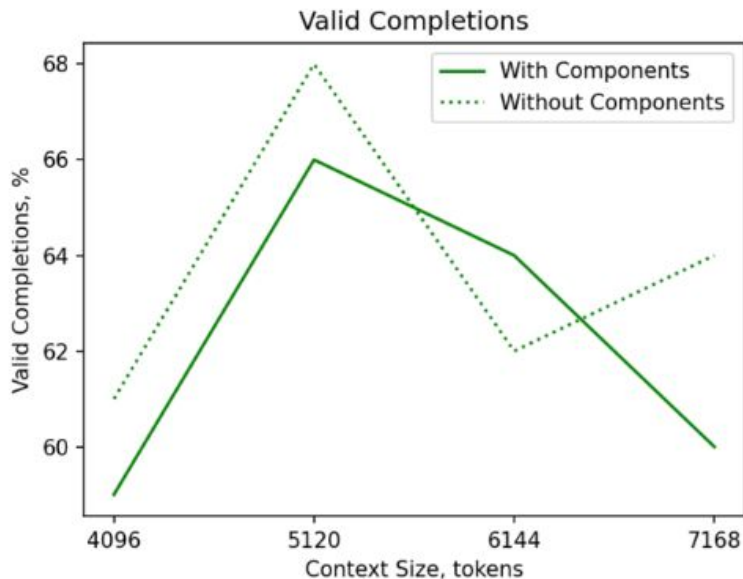
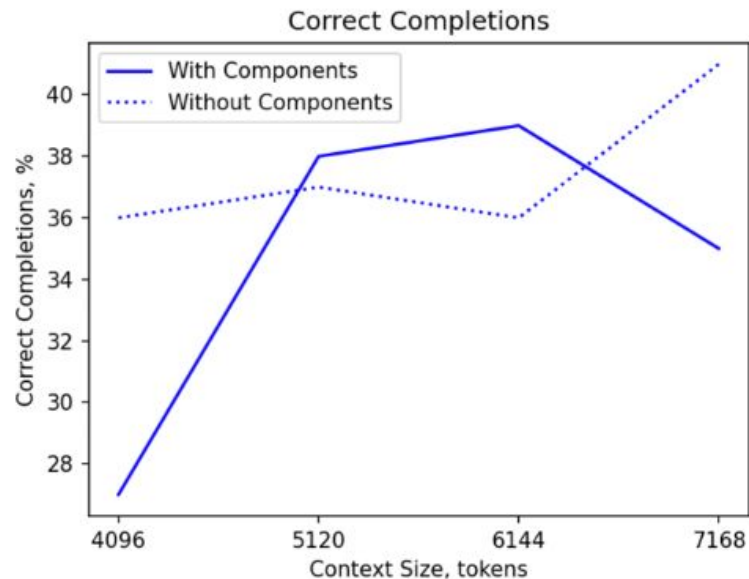
- Correctness: **41%**
- Validity: **64%**
- Speed: **67 char/s**



Results. Optimal model size



Results. OpenAPI metadata in prompt



Results. YAML vs JSON

ktu

1922

Results:

- Correctness: **27% (-8%)**
- Validity: **56% (-5%)**
- Speed: **72 char/s (-11%)**

JSON vs YAML size:

- Characters: **+ 43.5%**
- Tokens: **51.5%**

```
paths:
  /pets:
    get:
      summary: List all pets
      operationId: listPets
      tags:
        - pets
      parameters:
        - name: limit
          in: query
          description: How many items to return at one time
          required: false
          schema:
            type: integer
            maximum: 100
            format: int32
```

```
"paths": {
  "/pets": {
    "get": {
      "summary": "List all pets",
      "operationId": "listPets",
      "tags": [
        "pets"
      ],
      "parameters": [
        {
          "name": "limit",
          "in": "query",
          "description": "How many items to return at one time (max 100)",
          "required": false,
          "schema": {
            "type": "integer",
            "maximum": 100,
            "format": "int32"
          }
        }
      ]
    }
  }
}
```

```
<s> paths:<0x0A> /pets:<0x0A> get:<0x0A>
summary: List all pets<0x0A> operationId:
listPets<0x0A> tags:<0x0A> - pets
<0x0A> parameters:<0x0A> - name: limit
<0x0A> in: query<0x0A>
description: How many items to return at one time
(max 100)<0x0A> required: false<0x0A>
schema:<0x0A> type: integer
<0x0A> maximum: 100<0x0A>
format: int32
```

359

Characters

101

Tokens

```
<s> "paths": {<0x0A> "/pets": {<0x0A> "
get": {<0x0A> "summary": "List all pets",
<0x0A> "operationId": "listPets",<0x0A>
"tags": [<0x0A> "pets"<0x0A>
],<0x0A> "parameters": [<0x0A>
{<0x0A> "name": "limit",<0x0A>
"in": "query",<0x0A>
description": "How many items to return at one time
(max 100)",<0x0A> "required": false,<0x0A>
<0x0A> "schema": {<0x0A>
type": "integer",<0x0A> "maximum": 100
,<0x0A> "format": "int32"<0x0A>
}<0x0A> }<0x0A> ],
```

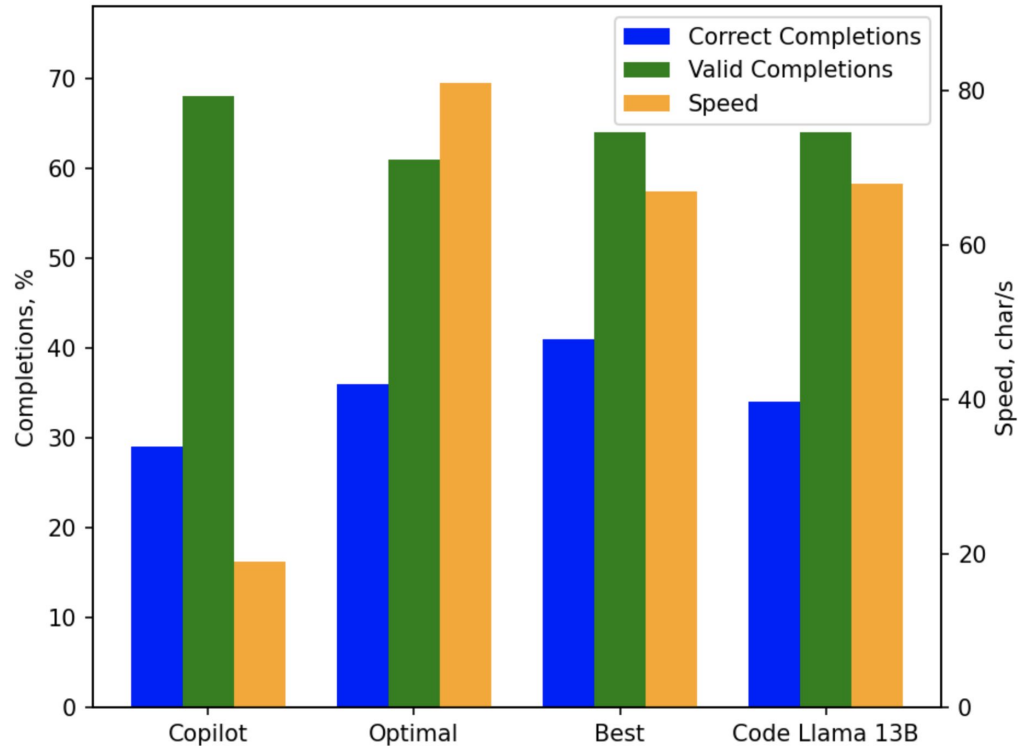
515

Characters

153

Tokens

Results. Comparison



Next steps

- Code Llama fine tuning
 - Most of the definitions were already used for training
 - Dataset size >>> Learned parameters size
- Build VSCode extension prototype
 - Would help to evaluate usability