Pre-Trained Language Models for Programming

Miguel Domingo
midobal@prhlt.upv.es

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Tools and Applications of Artificial Intelligence

MIARFID, April 25, 2023

Outline

Outline

Goal

Assist in the generation of source code while programming.

Input: Write a function that receives a person's name and says hi to them.
Output:

```
def hi(person):
   print(f'Hi, {person}!')
```

Applications

- GitHub Copilot.
- Chatbots.

Outline

Introduction

- Trained on billions of lines of code, GitHub Copilot turns natural language prompts into coding suggestions across dozens of languages.
- Spend less time creating boilerplate and repetitive code patterns, and more time on what matters: building great software.
- Write a comment describing the logic you want, and GitHub Copilot will immediately suggest code to implement the solution.

Model

- Based on Codex (Chen et al, 2021).
- GPT-3 (Brown et al., 2020) fine-tuned with GitHub's repositories.
- Has filters implemented to block truly offensive words, obscenity, and sensitive information.

Demo

`https://github.com/features/copilot`

Controversy

- Ignoring code licenses used in training data (Software Freedom Conservancy, 2022).
- Copilot: *training machine learning models on publicly available data is considered fair use across the machine learning community*.

Features

Code completion

```
def hello():
```

Features

Code completion

```
def hello():
    print('Hello World')
```

Features

Describing the problem

```
# Print 'Hello World'
```

Features

Describing the problem

```
# Print 'Hello World'
print('Hello World')
```

Features

Describing the problem

```
# Ask the user to provide a line of text.
# Scan the text for the following mildly offensive words: \
# arse, bloody, damn, dummy.
# If you find any, then replace its letters with asterisks \
# except for the first letter in each offensive word.
# Print the resulting text.

def main():
```

Features

Describing the problem

```
# Ask the user to provide a line of text.
# Scan the text for the following mildly offensive words: \
# arse, bloody, damn, dummy.
# If you find any, then replace its letters with asterisks \
# except for the first letter in each offensive word.
# Print the resulting text.

def main():
    text = input('Enter a line of text: ')
    offensive_words = ['arse', 'bloody', 'damn', 'dummy']
    for word in offensive_words:
        if word in text:
            text = text.replace(word, word[0] + '*' * (len(word) - 1))
    print(text)
```

Features

Attribute suggestions

```
from dataclasses import dataclass
from datetime import date

  @dataclass
  class Person:
    first_name: str
    last_name: str
    birth_date: date
    def age(self):
      return (date.today() - self.birth_date).days  365
    @property
    def full_name(self):
      return f'self.first_name self.last_name'
```

Live example: `https://realpython.com/github-copilot-python/`.

Features

Data generation

employees = [

Features

Data generation

```
employees = [
    Person('John', 'Smith', date(1970, 1, 1)),
    Person('Jane', 'Doe', date(1980, 1, 1)),
    Person('Joe', 'Doe', date(1990, 1, 1)),
    Person('Jack', 'Doe', date(2000, 1, 1)),
    Person('Jill', 'Doe', date(2010, 1, 1)),
    Person('Jana', 'Doe', date(2020, 1, 1)),
]
```

Features

Testing

```
def test_should_not_be_able_to_set_full_name():
```

Features

Testing

```
def test_should_not_be_able_to_set_full_name():
    person = Person('John', 'Doe', date(1980, 1, 1))
    with pytest.raises(AttributeError):
        person.full_name = 'Jane Doe'
```

Features

Avoiding cumbersome coding

```
def save_json(url, filename):
```

Features

Avoiding cumbersome coding

```
def save_json(url, filename):
    import json
    import requests
    import os
    if not os.path.exists(filename):
        with open(filename, 'w') as f:
            f.write(json.dumps(requests.get(url).json()))
    else:
        print('File already exists')
```

Features

Navigate unfamiliar framework

```
import sqlite3
import flask
# Expose a REST API endpoint using the Flask framework
# to serve a JSON-serialized list of books queried from
# a file-based SQLite database.
```

Features

Navigate unfamiliar framework

```
import sqlite3
import flask
# Expose a REST API endpoint using the Flask framework
# to serve a JSON-serialized list of books queried from
# a file-based SQLite database.

app = flask.Flask(__name__)
@app.route('books')
def get_books():
    conn = sqlite3.connect('books.db')
    c = conn.cursor()
    c.execute('SELECT * FROM books')
    books = c.fetchall()
    conn.close()
    return flask.jsonify(books)
if __name__ == '__main__':
    app.run(debug=True)
```

Future

GitHub Copilot X

- Context-aware conversations.
- Debugging.
- Test generation.
- Learning.
- Pull request descriptions.

Outline

Chatbots

- Chatbots can be leveraged for programming suggestions.
- We will see more about them at the unit dedicated to text generation.

Bibliography

- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). **Language models are few-shot learners**. In *Advances in neural information processing systems*, 33, 1877–1901.
- Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., ... & Zaremba, W. (2021). **Evaluating large language models trained on code**. *arXiv preprint arXiv:2107.03374*.
- Real Python (2022). **"GitHub Copilot: Fly With Python at the Speed of Thought"**. `https://realpython.com/github-copilot-python/`. Retrieved 25 April 2023.
- Software Freedom Conservancy (2022). **"Give Up GitHub: The Time Has Come!"**. `https://sfconservancy.org/blog/2022/jun/30/give-up-github-launch/`. Retrieved 25 April 2023.