ESC190 Winter 2025 Lab 10: VIBE CODING

Vibe coding, according to Merriam-Webster, is a *recently-coined term for the practice of writing code, making web pages, or creating apps, by just telling an AI program what you want, and letting it create the product for you. In vibe coding the coder does not need to understand how or why the code works, and often will have to accept that a certain number of bugs and glitches will be present. The verb form of the word is vibe code.*. The term was coined by Andrej Karpathy, a UofT CS 2009 graduate.

You will learn how to use the OpenAI application programming interface (API), which allows applications to communicate with each other by sending and receiving data, to generate Python function code, execute the generated code, and test its correctness. You will analyze when the large language model (LLM) generates accurate code and when it fails.

1 Setup

Create a folder for the work in this lab named lab10.

Download the "notopenai client" from `https://www.cs.toronto.edu/~guerzhoy/190/notopenai.zip`. Unzip the folder and put it in the lab10 folder.

Create `lab10.py` and put it in the lab10 folder as well. Open lab10.py.

Obtain your own API key from `https://esc190-winter-2025-student-key-lookup.vercel.app/`. Do not share it with other students.

Copy the API key into the top of your lab10.py where `API_KEY = the api key you obtained`.

We are limited to a few calls a second total, so: be patient, and please don't overload the system.

2 Test calling the API

The `get_response(prompt)` function has been provided to you below. Try calling this function with the prompt being "What is engineering science?"

Print out the response.

```
from notopenai import NotOpenAI
import json

# Input the API key obtained for the lab
API_KEY = "..."
CLIENT = NotOpenAI(api_key=API_KEY)

def get_response(prompt):
    chat_completion = CLIENT.chat.completions.create(
    messages=[
        {
            "role": "user",
            "content": prompt,
        }
    ],
    model="gpt-3.5-turbo", # the GPT model to use
)
    response_str = chat_completion.choices[0].message.content
    return response_str
```

3 Prompt engineering

Write a prompt that asks the AI to generate a Python function for you that output the factorial of the input n. Extract the code portion from the response and print out the resulting code.

4 Test the code generated by AI

Given a few test cases in the following format:

```
    test_cases = [
        {"intput": 3, "expected_output": 6},
        {"intput": 4, "expected_output": 24},
    ]
```

Write a function `check_result(generated_code, test_cases)`, print out how many cases passed and failed.

You can use `exec(generated_code)` to define the function from a string, e.g.,

```
def check_result(generated_code, test_cases):
    """
    Check which test cases failed from the generated code
    generated_code - str
    test_caes - dict

    e.g., if the generated_code is:
    def fun(x):
        return x
    """
    exec(generated_code, globals())
                         # Execute the string that
                         # contains the function will make
                         # the function available to be called below
                         # print(fun(1)) # this will print 1 in the example above.
```

In your prompt, you might want to explicitly ask for the generated function to be named, e.g., `fun(x)`, and you might ask to generate the function only.

5 Test cases

Here is a task on which GPT-3.5 usually doesn't succeed:

```
s = '''Date,Character,Age,HeightCm,AppleCount,MoodRating
2025-01-15,Snow White,14,157.5,1,8.5
Doc,200,91.4,3,7.2
2025-01-16,Grumpy,199,89.0,0,3.4
2025-01-16,202,94.0,2,9.7
2025-01-17,Sleepy,202,90.2,1,6.3
Bashful,198,88.5,1,5.8
2025-01-18,Sneezy,197,92.3,2,7.4
2025-01-18,Dopey,195,87.1,4,8.9
2025-01-19,,42,175.6,0,2.1
Prince,25,185.3,2,9.5
2025-01-20,Huntsman,38,178.4,1,6.7
2025-01-20,250,92.0,3,7.3
```

```
2025-01-21,Forest Animals,5,30.5,4,9.2'''

#print(get_response('''Write Python code to parse a CSV string
                    formatted like the following. Result needs
                    to be a dictionary of dictionaries\n\n\n''' + s))
```

Note that here, some dates and character names are missing, and GPT-3.5 is not (usually) smart enough to figure out how to handle that. Write test cases for the function, experiment with different prompts on chatgpt.com, and demonstrate how you can generate code in our framework that passes the test cases.

6 Creating new data

We created a custom graph format in ESC190. Use ChatGPT to create a different graph in the same format, and verify that our algorithms still run on the newly-created graph.

Some ideas: input examples from lecture; input the definition of Graphs from lecture.

7 Working with graphs

Write a function to find the order in which a particular node would be printed using breadth-first travesal/depth-first traversal (both recursive and iterative), given that the traversal starts at a particular node. For example, if you start from `"YYZ"` and `"YUL"` would be printed 5-th, output 5.