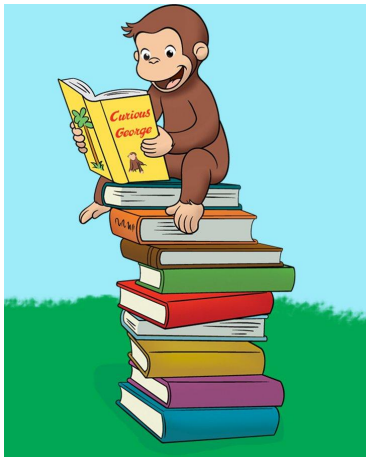
Curious Monkey Chat



Curious George is a smart monkey. He cannot pronounce human words though :-(But he has a computer and loves the internet! One day, he read the “Infinite monkey theorem” story. He is so excited! He has a brilliant idea. With the feedback from his friends, he can learn how to communicate in human language from randomness on the keyboard!

George wants to build a chat bot that can learn! He names it as CMC, which stands for Curious Monkey Chat.

- George loads sentences that he found in his favorite books into this chatbot.
- George will ask his friends to connect to the Curious Monkey ChatBot from their computers.
- When a friend sends a message, the bot will pick one of the sentences as the response.
- Then, his friends can click like/unlike buttons to inform the bot whether they like the response or not. Of course, his friends may not click the button. They can just keep sending more messages.
- If a friend likes the response of a given message, then the bot will remember to return the same response next time. If a friend doesn’t like the response of a given message, then the bot will pick other sentences as the response next time.

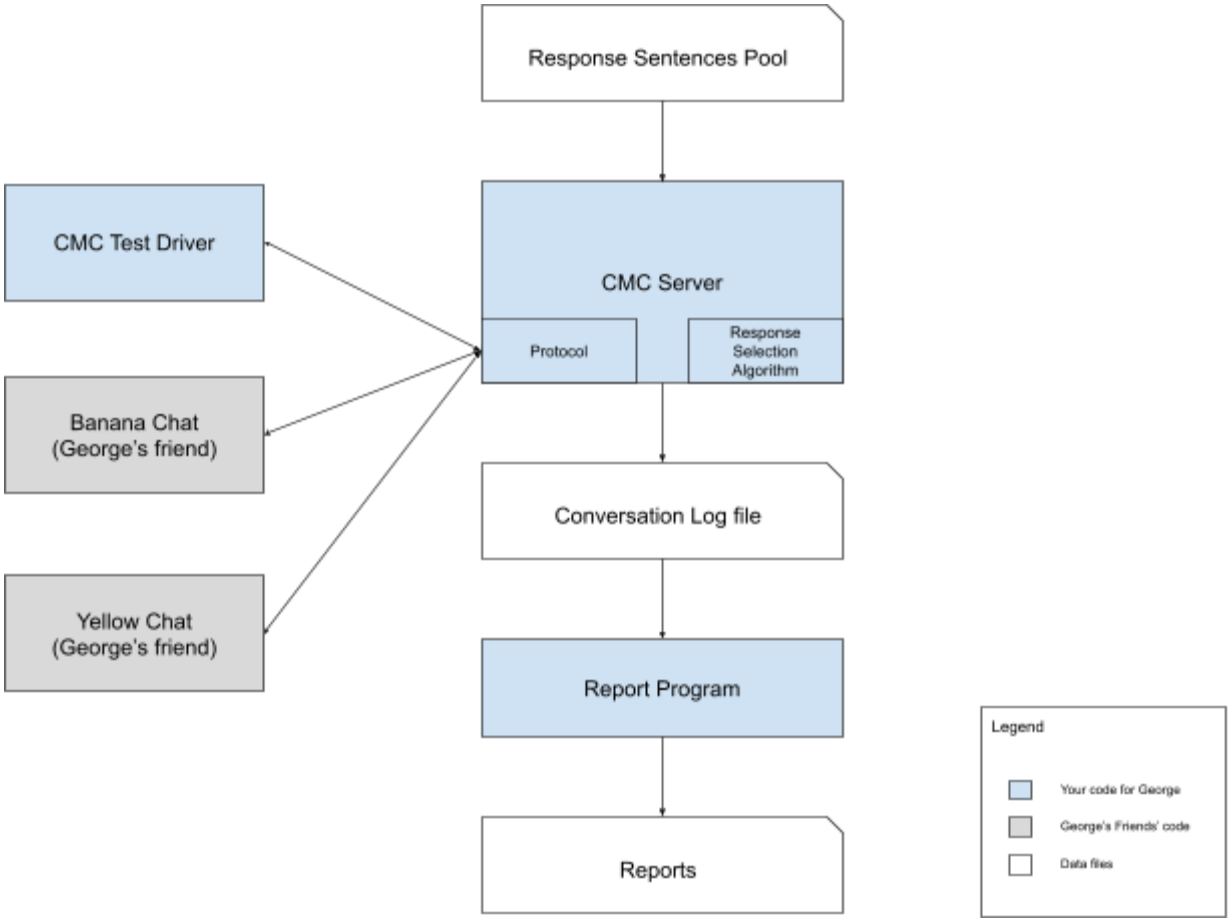
A basic system picks sentences purely randomly and the learning is isolated to one given friend, that is too slow. So, George cames up some more ideas:

- If a friend sends a message that other friends have already asked, other friends’ opinions can influence the response selection algorithm
- If a friend sends a sentence that nobody ever asked, then the response selection algorithm can consider all likes/unlikes regardless of the corresponding messages.

By observing the conversions, CMC will become smarter and so will George.

Since you are a brilliant engineer, George begs you to build this system for him.

Requirements of the CMC System



- The CMC Server is a server daemon that serves multiple chat clients via HTTP protocol. It responds to **at least** two types of messages.
    - `/message`: receives an input question and responds a sentence
    - `/rate`: informs CMC whether the friend likes the response or not
- The server must record the conversation history, at least including, “friend”, “message”, “response”, “like/unlike”
- While the server is running, it will become smarter as more conversations and likes/unlikes are collected.
- The data should be stored to files
- A report program can read those files and tell George
    - How many friends he has and who they are
    - He wants to share bananas with friends that helps the most, so he wants to know
        - Who sent the most messages
        - Who sent the most unique message
        - Who gave the most like/unlike
    - He wants to learn, so he wants to know
        - Successful (message, respons) pairs
        - Unsuccessful (message, response) pairs

- He relies on you to design an algorithm to judge whether how successful the (message, response) is

Task 1: Design doc

Please write a brief design document that covers at least the following content:

1. A system diagram.
2. A protocol design. Some of George's geek friends want to develop their own chat client!
3. The in memory data structures.
4. The schema of data to be stored in file to support the report program. Please define it in JSON or protobuf.
5. The algorithm for the "response select algorithm" in pseudo code.
6. The algorithm design for the report program in pseudo code.
7. In Chinese or English. In PDF format.

Task 2: CMC Server and a test driver

Please develop a server daemon that implements your above design.

1. A server app.
2. In Python, Golang, C++ or other programming languages.
3. Please try to avoid using heavy frameworks.
4. It should be able to compile and run on Linux or macOS .
5. Please fill the response candidates collection with some common chit chat utterances. Just for fun, you can also find some of George's quotes. For example, here.

Please develop a test driver that can interact with the CMC Server

1. It can be a simple script on terminal
2. It can communicate with the CMS Server according to the protocol design.
3. Multiple copies can be started simultaneously to simulate multiple friends.

For example

```
% talk-to-george --fried=Mark
CMC: Hello Mark!
Mark: Is George there?
CMC: He was a good little monkey and always very curious.
[CMC Survey]: 1 - like, 2 - unlike, 3 - continue
[Mark]: 3
Mark: Are you still coming over for the banana ice cream?
CMC: George promised to be good. But it is easy for little monkeys
to forget.
[CMC Survey]: 1 - like, 2 - unlike, 3 - continue
[Mark]: 1
```

```
Mark: Are you still coming over for the banana ice cream?
CMC: George promised to be good. But it is easy for little monkeys
to forget.
[CMC Survey]: 1 - like, 2 - unlike, 3 - continue
[Mark]: 3
Mark: Haha, you are learning :-)
...
```

[Bonus 1] feel free make the test driver more fancier
[Bonus 2] flesh the conversation data to file as the event happens. So George doesn't miss learning materia
[Bonus 3] ability to inspect the stats of the critical internal in memory data structures

Task 3: The Report Program

Please develop a report script that reads the output files of the CMC Server and prints out the report for George according to the requirement.

1. It is a command line script.
2. In Python or other popular programming languages.

[Bonus 4] feel free to add more reports that you want to recommend to George.

Output of the Assignment

1. Design document for Task1
2. Source code for Task 2 and Task3. Test cases are desired. Please include instructions on how to install and run your app, including:
    - required packages and frameworks
    - command lines for building and running all components.
3. Response sentence pool data file for Task 2.
4. Three test sessions to simulate three friends.
    - Take the screenshots of these three sessions
    - Capture the conversation log files from the CMC server
    - The output report of Task3
5. A project document that covers:
    - Please indicate whether you have finished each tasks
    - Please briefly explain the layout of your submission package
    - Please explain how to run your code so George can try it out
    - Please describe possible TODO items should you continue working on this project to the next level.

- Please describe how you spend time. For example, how long this assignment takes you in total, as well as how the time was distributed among doing research, coding, testing. George wants to share his banana and ice cream with you :-)

Submission & Evaluation

Thanks for considering Cicada Speech and spending your precious time working on this assignment! Please submit your works to the Google Drive folder before the deadline. Early submission is ok, it is one of the factors that we will consider. However, we always weigh quality and comprehensiveness.