

WildBytes – puntata 6. Stanislavski!

Ciao Youtubers e bentrovati su WildBytes! in questo nuovo sfavillante episodio, adesso che abbiamo gli strumenti per ingegnerizzare il nostro prompt, useremo una sorta di metodo "stanislavskij" per istruire chat gpt ed avere la risposta che stiamo aspettando.

Abbiamo implementato un sistema per creare un semplice prompt, adesso implementeremo il prompt come suggerito da openAi.

```
def createOpenAiPrompt(self, question):
    messages = [
        {
            "role": "system",
            "content": "Rispondi alla domanda tenendo conto delle informazioni e delle direttive fornite. Se sono "
                "presenti informazioni in MEMORY, basa la tua risposta su di esse. Se sono state fornite "
                "direttive specifiche, rispetta quelle direttive. Se è stato fornito un contesto, "
                "assicurati che la tua risposta sia pertinente a quel contesto. E, infine, se sono stati "
                "forniti esempi, usa quegli esempi come guida per formulare la tua risposta.",
        }
    ]
    if self.isMemoryMode:
        messages.append({"role": "system", "content": f"MEMORY: {' '.join(self.memories)}"})
    if self.isDirectiveMode:
        messages.append({"role": "system", "content": f"DIRECTIVE: {' '.join(self.directives)}"})
    if self.isContextMode:
        messages.append({"role": "system", "content": f"CONTEXT: {' '.join(self.contexts)}"})
    if self.isExampleMode:
        messages.append({"role": "system", "content": f"EXAMPLE: {' '.join(self.examples)}"})
    messages.append({"role": "user", "content": f"{question}"})
    return messages
```

Per usare questo metodo dobbiamo creare un meccanismo di switch e lo possiamo realizzare usando una variabile di controllo isStandardMode che attiveremo semplicemente scrivendo #mode nel prompt al posto della domanda.

```
elif "reset" in command:
    self.reset()
elif "mode" in command:
    self.changeMode()
elif "prompt" in command:
    self.printPrompt()
```

```
def changeMode(self):
    answer = input("Vuoi attivare la modalità standard - 0 o la
    modalità openAI - 1? [0/1] ")
    if answer == "0":
        self.isStandardMode = True
    elif answer == "1":
        self.isStandardMode = False

def printPrompt(self):
    if self.isStandardMode:
        print(self.createSimplePrompt(""))
    else:
        print(self.createOpenAiPrompt(""))
```

In questo modo possiamo passare da una modalità all'altra e verificare il prompt.

Nello stesso modo in cui un regista istruisce l'attore utilizzando una certa sulla verità emotiva che porta all'identificazione profonda con il proprio personaggio. Quello che dobbiamo fare per istruire chat gpt è quello di fornirgli tutti i dettagli necessari per creare una risposta in base al suo personaggio.

Nell'esempio del sergente Hartman, se dico a chatGpt: Sono il sergente Maggiore Hartman istruttore di corpi speciali dei Marines. Tutte le volte che ti faccio una domanda dovrai rispondere: Signorsì certo signore. Hai capito bene luridissimo verme? Lui senza direttive, contesto, memoria e esempi, potrebbe non cogliere la citazione e magari rispondere: SignorSì luridissimo verme che potrebbe non essere esattamente quello che sto cercando.

In questo caso il mio contesto potrebbe essere un qualcosa del tipo: Sei una recluta che ambisce a entrare nel corpo dei marines e stai facendo l'addestramento. E' molto importante rispondere correttamente agli ufficiali, che potrebbero offenderti per capire la tua volontà e dedizione ad entrare nel corpo.

Guardando il film, ci possiamo rendere conto che tutte le reclute devono rispondere il più brevemente possibile usando la formula: Signore, {risposta}. signore e che se la domanda in realtà non sta chiedendo qualcosa ma semplicemente imponendo un obbligo, il miglior modo di rispondere è: Signorsì , certo signore.

Possiamo quindi aggiungere alle directive una frase del tipo:

Rispondi usando il minor numero di parole e ricordati che la migliore risposta ad una domanda che non chiede niente è sempre signorsì, certo

signore. Se invece sai cosa rispondere la formula corretta è sempre: Signore {risposta}. Signore.

Inserendo questi dati è più facile che facendogli la famosa domanda:
Sono il sergente Maggiore Hartman istruttore di corpi speciali dei Marines. Tutte le volte che ti faccio una domanda dovrai rispondere: Signorsì certo signore. Hai capito bene luridissimo verme?

lui risponda: Signorsì, certo signore.

Il senso di tutto questo è che posso ovviamente caricare l'intera sceneggiatura e chiedere a chatGpt di rispondere esclusivamente in base al testo. La grandezza di questo sistema però è che in qualche modo chat Gpt sa improvvisare e quindi potrebbe darvi una risposta incredibile come fece Rutger Hauer in Blade Runner quando improvvisò la celebre frase: "Io ne ho viste cose che voi umani non potreste immaginarvi..."

Possiamo andare avanti fornendogli una memoria: il tuo nome è Leonard. Il tuo cognome è Lennon Lawrence. Quindi dargli degli esempi: Se un ufficiale ti chiede come ti chiami devi rispondere Signore, Soldato {cognome}. Signore. Ovviamente possiamo andare avanti fornendogli tutte le indicazioni sulle sue origini, circa il fatto che non ha fratelli, dove è nato, dove abita in modo che ci risponda come fa il soldato nel film.

Una delle cose che però possiamo sviluppare è questa. Nel film il sergente Hartman affibbia dei nomignoli a tutti. Lo fa perchè nel suo contesto cerca di sottomettere tutti quei soggetti che reputa più difficili. Però è possibile creare un qualcosa in modo che se noi, in quanto sergenti Hartman affibbiamo un nomignolo a chat gpt lui lo metta automaticamente in memoria? La risposta è sì e per farlo possiamo usare una forma speciale introdotta da openAi nella nuova release delle api. In pratica possiamo dirgli che se all'interno della domanda trova un qualcosa che deve ricordare la sua risposta sarà "function_call": funzione in modo da richiamare la funzione per memorizzare automaticamente qualcosa.

La prima volta che ho visto questa proprietà di chatGpt ho cominciato a farmi un sacco di domande che ho scoperto avere un risposta una risposta molto più semplice di quello che pensavo:

k:
clamoroso ora funziona. In pratica chat gpt decide da solo se l'argomento deve essere memorizzato solamente perchè la funzione si chiama addToMemory...
chatGpt:

Esatto! La capacità di OpenAI GPT-3 di comprendere e rispondere alle istruzioni è davvero impressionante. Quando gli fornisci una funzione come `addToMemory`, e poi gli chiedi qualcosa che sembra dover essere memorizzato (come "memorizza che mi chiamo k"), riconosce il contesto e decide di chiamare quella funzione. Questo comportamento è basato su come GPT-3 è stato addestrato e sulla vasta quantità di dati a cui ha avuto accesso. È una combinazione di comprensione del linguaggio e di intuizione basata sui dati.

Tuttavia, è sempre una buona idea testare e verificare che tutto funzioni come previsto, poiché GPT-3 può non sempre interpretare le cose esattamente come ci si aspetta.

```
def addToMemory(self, content):
    """Aggiunge informazioni alla memoria"""
    self.parent.addMemoriesFromGpt(content)
    print(f"sto aggiungendo alla memoria: {content}")
    return content

def getAnswer(self, question):
    """
    Ottiene una risposta da openAI. Ogni risposta costa 0.002 dollari.
    :param question:
    :return:
    """
    if type(question) is str:
        messages = [
            {"role": "user", "content": f"{question}"},
        ]
    elif type(question) is list:
        messages = question
    else:
        return "Errore: la domanda deve essere una stringa o una lista di messaggi."
    self.chatHistory.append({"role": "user", "content": f"{question}"})
    functions = [
        {
            "name": "addToMemory",
            "description": "Aggiunge informazioni alla memoria",
            "parameters": {
                "type": "object",
                "properties": {
                    "content": {
                        "type": "string",
                        "description": "Informazioni da memorizzare",
                    }
                },
                "required": ["content"],
            },
        },
    ]
    response = openai.ChatCompletion.create(
        model=self.model,
        messages=messages,
        max_tokens=50, # limita la lunghezza della risposta
        temperature=0, # 0 = risposta più probabile, 1 = risposta più creativa
    )
```

```

        functions=functions,
        function_call="auto",
    )
    if isinstance(response, str):
        return f"Errore: {response}"
    answer = response['choices'][0]['message']['content']
    self.promptTokens += int(response['usage']['prompt_tokens'])
    self.completionTokens += int(response['usage']['completion_tokens'])
    self.totalTokens += int(response['usage']['total_tokens'])

    if 'function_call' in response['choices'][0]['message'] and
response['choices'][0]['message']['function_call']['
    'name'] == 'addToMemory':
        try:
            arguments =
json.loads(response['choices'][0]['message']['function_call']['arguments'])
            content_to_add = arguments['content']

            self.addToMemory(content_to_add)
            # E aggiungi la risposta al tuo chatHistory
            self.chatHistory.append({"role": "assistant", "content": f"Added
to memory: {content_to_add}"})
        except Exception as e:
            print(response)
            return f"Errore: {e}"
    else:
        # Altrimenti, semplicemente aggiungi la risposta normale al tuo
chatHistory
        self.chatHistory.append({"role": "assistant", "content": f"{answer}"})

    return f"Risposta: {answer}\n{self.remainingTokens()}"

```

Ovviamente devo creare una relazione fra le librerie in modo che se chatGpt chiama la funzione va ad aggiungere i parametri nella variabile memories della classe chat.

```

def __init__(self, parent):
    openai.api_key = secretKeys.openAi
    self.initVariables()
    self.initHistory()
    self.parent = parent

```

Nella classe chat quindi devo aggiungere una funziona che mi permette di aggiungere qualcosa alla memoria quando chatGpt stabilisce che è un qualcosa che deve/può memorizzare.

```

def addMemoriesFromGpt(self, content):
    self.memories.append(content)
    self.isMemoryMode = True

```

Per testare la chiamata di funzione, possiamo far girare il programma e semplicemente dire alla chat: il mio nome è Bond, James Bond.

Domanda: Il mio nome è bond, james Bond
sto aggiungendo alla memoria: Il nome dell'utente è Bond, James Bond

Risposta: None

Ti rimangono \$0.0077. Puoi ancora utilizzare 3859 tokens.

Domanda: come mi chiamo?

Risposta: Il tuo nome è Bond, James Bond.

Ti rimangono \$0.0035. Puoi ancora utilizzare 1736 tokens.

Domanda: #memorize print

Commands: ['#memorize', 'print']

o: Il nome dell'utente è Bond, James Bond

Domanda:

come potete vedere chat gpt ha inserito il nome nella memoria.

Bene, siamo giunti al termine di questa puntata, e oltre a ricordarvi gli esempi sono Open Source con licenza MIT vi ricordo che nella descrizione di ogni puntata, potete trovare il link alla mia pagina gitHub dove trovare il codice che abbiamo utilizzato durante il video, e il pdf con il riassunto dei punti chiave oltre ai link per gli eventuali approfondimenti. Ma prima di salutarvi vi ricordo di mettere un like e iscrivermi al canale se non lo avete fatto e come sempre fate domande, tante domande... ma soprattutto sperimentate gente, sperimentate...

GoodByte Alla Prossima!