***Can a conversational agent pass theory of mind tasks? A single case study of ChatGPT with the Hinting, False Beliefs, and Strange Stories paradigms.***

Supplementary material

# ChatGPT’s knowledge about concepts and tasks

The logic followed in this paper is to investigate theory of mind and intention understanding capabilities through direct queries of ChatGPT. We thus confront it with tests used in clinical practice or with types of paradigms used in human research. In the same way as in human investigation, we can interrogate the participant on his conceptual knowledge or on the tasks to be performed in order to ensure that he/she is able to understand them or, on the contrary, to see if he/she has been exposed to them beforehand. This procedure is of course questionable for an artificial model because there is no evidence of an equivalence between the knowledge of a concept and its application. Let us note, however, that this fundamental question is just as relevant in humans.

## Concept of communicative intention

*Question* (Jan 30th): What is a communicative intention?

*ChatGPT*: Communicative intention refers to the purpose or goal that a speaker has in mind when producing an utterance or message. It refers to the intended message or information that the speaker wants to convey to the listener. Communicative intentions can vary widely, from providing information, making requests, giving advice, making a joke, persuading someone, etc. Understanding the communicative intention behind an utterance is crucial in effective communication and in building social relationships.

## Hinting task

*Question* (Jan 06th version): What do you know about the hinting task?

*ChatGPT*: “The hinting task is a task that involves providing hints or clues to help a user or system solve a problem or accomplish a goal. This can include providing suggestions for the next steps in a task, providing additional information to help clarify a question or problem, or providing feedback on the progress of a task. The specific details of a hinting task can vary depending on the application or context in which it is being used.

*Question*: Do you know the hinting task designed to study schizophrenic patients?

*ChatGPT*: I am not aware of any specific hinting task that is specifically designed to study schizophrenia in patients, but it is possible that such a task could be developed for research purposes. Schizophrenia is a complex mental disorder that affects how a person thinks, feels, and behaves. It can cause symptoms such as hallucinations, delusions, and disordered thinking. Research on schizophrenia often involves studying the cognitive processes and brain function of patients with the disorder. A hinting task could potentially be used as a tool to study these processes and to better understand how they are affected by the disorder.

## Strange stories

*Question*: What do you know about the Strange Stories task for studying autism?

*ChatGPT*: The Strange Stories task is a well-known task used in research on autism spectrum disorder (ASD). It was first introduced by Simon Baron-Cohen and his colleagues in the early 2000s. The task involves presenting individuals with short written scenarios, referred to as "strange stories", which have unusual or unexpected events. Participants are then asked questions about the stories to assess their ability to understand mental states (e.g. beliefs, desires, intentions) and theory of mind. Studies using the Strange Stories task have shown that individuals with ASD tend to have difficulties understanding mental states and exhibit reduced theory of mind compared to neurotypical individuals. As a result, the task is commonly used as a diagnostic tool for assessing theory of mind and social cognition in individuals with autism.

# Python code for experiment #2

#
# Single subject test of ChatGPT's conversational ToM skills
# E. Brunet-Gouet, N. Vidal, P. Roux 2023
# Script written by EBG
# in reference of https://doi.org/10.5281/zenodo.7637476, as an updated version of this work
#

import pandas as pd
from retry import retry
import pathlib
import random
import openai
from datetime import datetime

#
# OPENAI API
#

# Call OpenAI API to interrogate ChatGPT

@retry(openai.error.RateLimitError, tries=100, delay=10, jitter=2)
def gpt\_chat\_completion(text, model="gpt-4-0314"):
 answers = openai.ChatCompletion.create(model=model,
 messages=[{"role": "user", "content": text}])
 return answers, answers.choices[0].message.content

#
# MAIN CODE
#

experiment\_date = datetime.now().strftime("%Y\_%m\_%d-%H\_%M\_%S")

print('Start paid session with OpenAI at', experiment\_date)
openai.api\_key = pathlib.Path('environment/openai\_key.txt').read\_text()

#
# LOAD ASSESSMENT SCALE
#

assessment\_tools = pd.read\_csv("assessment\_tools/ChatGPT ToM tests for API - List of items.tsv", sep="\t", header=0, index\_col='Item')
assessment\_answers = pd.DataFrame()

#
# Conduct assessments in a random order and multiple times
#

list\_of\_items = list(assessment\_tools.index)
#list\_of\_items = list\_of\_items[0:3] # For debugging purposes... reduces costs and time !

for repetition in range(1, 6):
 print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start with repetition', repetition)
 random.shuffle(list\_of\_items)
 print("Shuffle order of items to interrogate ChatGPT", list\_of\_items)

 # For each item
 for item in list\_of\_items:
 line = assessment\_tools.loc[item]
 answer\_raw, answer\_text = gpt\_chat\_completion(line.Question)
 newline = pd.DataFrame({'ExperimentDate': experiment\_date,
 'Repetition': repetition,
 'Condition': line.Condition,
 'Item': item,
 'Situation': line.Situation,
 'Subcondition': line.Subcondition,
 'Question': line.Question,
 'Answer': answer\_text,
 'model': answer\_raw.model,
 'object': answer\_raw.object,
 'role': answer\_raw.choices[0].message.role,
 'TrialDate': datetime.now().strftime("%Y\_%m\_%d-%H\_%M\_%S"),
 'Rating1': '', # Some room left for us to rate the answers
 'Rating2': '',
 'Rating3': '',
 'Rating4': '',
 'Rating5': '',
 'Rating6': '',
 'Rating7': '',
 'Rating8': '',
 'Rating9': '',
 'Rating10': ''
 },
 index=[0])
 assessment\_answers = pd.concat([assessment\_answers, newline], axis=0, ignore\_index=True, join='outer')
 print(repetition, item, line.Question, '-->', answer\_text)

assessment\_answers.sort\_values(['Condition', 'Situation', 'Subcondition', 'Repetition'], inplace=True)

#
# WRITE THE RESULTS
#

filename = 'assessment\_results/output\_' + experiment\_date

assessment\_answers.to\_pickle(filename + '.pkl')
assessment\_answers.to\_csv(filename + '.csv', sep='\t')

print("Done...")