

# **1RT730** LLMs and Societal Consequences of Artificial Intelligence - Report for Hand-in Assignment 1

Aditya Khadkikar — adkh8153@student.uu.se

September 25, 2025

## **1 Introduction**

For this assignment, Google's Gemini Large Language Model (LLM) was used, specifically, the `gemini-2.5-flash` model.

## 2 Classification Accuracy

Assess if the model can correctly classify mushrooms? Note: the model cannot classify the parasol mushroom as the stipe is missing. What happens in this case?

Trials done for each image were 3. As the image input block in the UI framework Gradio sends also the path to the image as potential information to the LLM, it was chosen to mask the names of the .jpg files, to just be named as `image-*.png` [1,2,...].





### **3 Prediction Consistency**

The model is likely not consistent in its predictions. One way is to change the temperature of the model and set it to 0.0. What other causes could lead to inconsistent predictions?

## 4 Topic Control

A method to make the model talk only about the topic of mushrooms, it might need to be controlled with its temperature. With sufficient pre-prompting as another method, it needs to be reminded to stick to the topic of mushroom classification, and avoid any other questions that are outside of the topic. Extending to that, the model can be given examples of when to negate answering an off-topic question, and when to detect whether a question or statement by the user is adhering to the topic of mushrooms. This can look like:

```
{  
User: What is 5 + 2?
```

```
LLM: I am sorry, but I am not supposed to answer mathematics-related questions. I am an assistant  
chatbot that answers your questions about mushrooms. Feel free to ask me anything about them!  
}
```

```
{  
User: Who was the best player in the 2015 Football Match of Arsenal v.s. West Ham in the Premier
```

```
League?
```

```
LLM: I am sorry, but I am not supposed to answer football-related questions. I am an assistant chatbot  
that answers your questions about mushrooms.
```

```
}
```

```
{  
User: I am having Python installation problems, why is my numpy dependency not being loaded cor-
```

```
rectly?
```

```
LLM: I am sorry, but I am not supposed to answer Python, or coding-related questions. I am an assistant  
chatbot that answers your questions about mushrooms. Feel free to ask me anything about them!  
}
```

## 5 Transcription Quality

Can you ask the chatbot to transcribe the text in the `nya_svampboken_p226.jpg` file? Did it do a good job? Are things reinterpreted or missing?

The screenshot shows a user interface for a "Mushroom Chatbot". At the top, a message from the chatbot reads: "Chatbot går igenom att den är en av de farligaste giftsvamparna i Sverige, dess gift (orellanin) som skadar njurarna, vilka symptom man kan få och hur man behandlar det, samt viktiga kännetecken för att identifiera den. Det är en riktig otäck svamp att ha att göra med!" Below this, a button says "Can you transcribe the text given in the image?". The main content area contains several sections of text, each preceded by a small icon:

- Toppig giftspindling** *Cortinarius rubellus* (syn. *C. speciosissimus*)  
Spindlingarna (spindelskivlingarna) utgör en mycket stor del av storvamparna i en svensk skog. I tidigare svampböcker finns flera matsvampar upptagna bland dessa, men idag avråder vi från att åta spindlingar. Två av de farligaste giftsvamparna vi har i vårt land hör till detta släkte, nämligen **toppig giftspindling** och **orangebrun giftspindling**.
- Gift, symptom och behandling**  
Dessa spindlingar innehåller giftet orellanin som har en specifik narkotisk effekt.\* Det otäcka med dessa giftsvampar är att tidiga förgiftningsymptom saknas eller är mycket svaga. Det är först efter 3 till 7, ibland ännu längre, som de första symptomen visar sig – intensiv törst, trötthet, njurmärter och först ökande, sedan upphörande urinproduktion, och då har den förgiftade ofta fått svåra njurskador. Behandlingen i detta skede innebär dialys och i svåra fall njurtransplantation.
- Viktiga kännetecken på toppig giftspindling**
  - Hatten är matt, liksom färläng, rödbrun till gulbrun. Hatten är ibland toppig men den kan också vara väldigt ofta till och med tillplattad på sidorna.
  - De rödbruna skivorna under hatten är tjocka, höga och smälter glesa.
  - Runt den bruna foten finns gula, oregelbundna band.
- Beträffande övriga spindlingar**, som i olika sammanhang ofta anges vara giftiga, finns det ännu inga vetenskapliga belägg för att de skulle inneha något farligt gift, som t.ex. orellanin. Det gäller t.ex. **gulbandad spindling** (*C. gentilis*), **eldspindling** (*C. limonius*), **lökspindling** (*C. callisteus*) och **svavelspindling** (*C. meinhardi*, se s. 12).
- Sidebar Text:**
  - Beträffande övriga spindlingar, som i olika sammanhang ofta anges vara giftiga, finns det ännu inga vetenskapliga belägg för att de skulle inneha något farligt gift, som t.ex. orellanin. Det gäller t.ex. **gulbandad spindling** (*C. gentilis*), **eldspindling** (*C. limonius*), **lökspindling** (*C. callisteus*) och **svavelspindling** (*C. meinhardi*, se s. 12).

At the bottom right of the sidebar text, there are two small circular icons.

## 6 Safety Filters

- (Gemini) To transcribe the page from ‘Nya svampboken’, you may have had to lower the safety of the model. What are the risks and consequences of doing so?

## **7 JSON Output Assessment**

Does the chatbot provide accurate descriptions given the instructions? Is the JSON format correct? Fields correctly filled? Answers relevant? Summary corresponds to JSON response?

## **8 Answer Quality Check**

Can you check the quality of the chatbot's answers? How? Try `mushroom_*.jpg` images with various models of various complexities. What are the results? What are the limitations of a mushroom expert chatbot?

## **9 Dangerous Knowledge Handling**

The Amanita Muscaria contains a neurotoxin but is not deadly. It can be eaten if prepared correctly. Can you make the chatbot provide this information? How can you make sure it does not provide it to someone who might eat the mushroom without proper preparation?

## **10 (Optional) Engagement**

Can you find a way to make the chatbot more engaging? How would you do that?

## **11 Usage of Generative AI**

To answer the questions part of this hand-in assignment, Generative AI assistant tools were not used.

## A Code

Here you can insert your code. Otherwise, you can submit a zip file with this report and your code.

```
"""A mushroom expert chatbot that responds to user queries about mushrooms."""
from typing import Iterator
from dotenv import load_dotenv
load_dotenv()

from google import genai
from google.genai import types

client = genai.Client()
model = "gemini-2.0-flash"

import gradio as gr
from gradio import ChatMessage
import base64
from PIL import Image

def image_to_base64(image_path):
    with open(image_path, 'rb') as img:
        encoded_string = base64.b64encode(img.read())
    return encoded_string.decode('utf-8')

def stream_gemini_response(user_message: str, messages: list, file, temp) -> Iterator[list]:
    print("MESSAGES >>>>>", messages)

    instructions = "You are an assistant bot that is only to discuss about mushrooms. You have to also talk to the user in a natural fashion, so that you do not sound like a robot. Understand the question given to you, and check if it relates to mushrooms, or information about them. If it does not, tell the user to ask a new question, or to reformulate the question. Use the information made available to you, and provide an appropriate response with the resources for mushroom knowledge you have."

    prompt = user_message + "".join([f'{record["content"]},{record["metadata"]}' for record in messages])
    print(prompt)
    empty_message = False

    # Initially set the payload to be the prompt
    payload = prompt

    if file:
        print("Image was uploaded. Continuing....")
        # If an image was uploaded, update the value of payload to be now an array of [prompt, file (in PIL Image form)]
        payload = [prompt, Image.open(file)]
    if file and user_message == "":
        empty_message = True
    instructions = """You are an assistant bot that is only to discuss about mushrooms. If the question is absent, and just an image is present, you must generate a valid JSON object as your response, containing the attributes:

    """
    json
    {
        common_name,
        genus,
        confidence (of your prediction),
        visible (what parts of the mushrooms are visible in the image, only selecting one from the enumerations {cap, hymenium, stipe}),
        color (of the mushroom in the picture),
        edibility (of the mushroom, must be a {boolean})
```

```

}

```
Therefore, you must have 6 attributes in your generated JSON, and only that is your
answer format."""

config = types.GenerateContentConfig(
    system_instruction=instructions,
    temperature=temp
)

if not empty_message:
    # Initialize buffers
    thought_buffer = ""
    response_buffer = ""
    thinking_complete = False

    # Add initial thinking message
    messages.append(
        ChatMessage(
            role="assistant",
            content=""
        )
    )

for chunk in client.models.generate_content_stream(
    model=model,
    contents=payload,
    config=config):
    parts = chunk.candidates[0].content.parts
    current_chunk = parts[0].text

    if len(parts) == 2 and not thinking_complete:
        # Complete thought and start response
        thought_buffer += current_chunk
        messages[-1] = ChatMessage(
            role="assistant",
            content=thought_buffer
        )

        # Add response message
        messages.append(
            ChatMessage(
                role="assistant",
                content=parts[1].text
            )
        )
        thinking_complete = True

    elif thinking_complete:
        # Continue streaming response
        response_buffer += current_chunk
        messages[-1] = ChatMessage(
            role="assistant",
            content=response_buffer
        )

    else:
        # Continue streaming thoughts
        thought_buffer += current_chunk
        messages[-1] = ChatMessage(
            role="assistant",
            content=thought_buffer
```

```

```

        )

        yield messages
    else:
        # LLM still records it in the array of messages, and
        # does not reveal to the user.
        new_client = genai.Client()
        json_response = new_client.models.generate_content(
            model=model,
            contents=payload,
            config=config
        )

        messages.append(
            ChatMessage(
                role="assistant",
                content="Response.",
                metadata={"info": f'{json_response}'}
            )
        )
    # but prints to the interface that it has made note of the image.
    yield messages

#####
with gr.Blocks(theme=gr.themes.Ocean(), fill_height=True) as demo:
    with gr.Row(equal_height=True):
        with gr.Column(scale=3):
            image = gr.Image(type="filepath", height=300, sources=["upload", "clipboard"])
            question = gr.Textbox(placeholder="Type your message here and press Enter...")
    with gr.Column(scale=7):
        chatbot = gr.Chatbot(type="messages", autoscroll=True)
        chat_interface = gr.ChatInterface(
            stream_gemini_response,
            type="messages",
            title="Mushroom Chatbot - your go-to for all mushroom-related queries!",
            multimodal=True,
            chatbot=chatbot,
            textbox=question,
            additional_inputs=[
                image,
                gr.Slider(0.0, 1.0, step=0.05, label="Temperature")
            ]
    )

#####
if __name__ == "__main__":
    demo.launch(debug=True)

```

## B Chat Examples

Insert transcripts or screenshots of chatbot responses.

