hugomcp@di.ubi.pt, 2024/25

Interaction with Large Scale Models

Practical Sheet 2

Using LLAVA on **Hugging Face** to experiment with different temperature values.

Step 1: Create a Free Hugging Face Account

1. Go to Hugging Face
2. Click **Sign Up** (if you don't have an account).
3. Follow the steps to create an account.

Step 2: Get a Free API Token

1. After logging in, go to your **Hugging Face Settings**: https://huggingface.co/settings/tokens
2. Click **New token** → Name it (e.g., `LLAVA_API`).
3. Set permissions to **"read"** and copy the token.

Step 3: Select a LLAVA Model

LLAVA (Large Language and Vision Assistant) has several versions:

- LLaVA-1.5-13B *(larger, better quality)*
- LLaVA-7B *(smaller, faster)*

Open one of the links above and test it in the **Hugging Face Spaces UI** (without API).

Step 4: Use LLAVA API with Python

You can interact with LLAVA using the Hugging Face Inference API.

Install `requests` (if not already installed)

```
pip install requests
```

Python Script to Send a Query with Temperature Control

```
import requests

# Set your Hugging Face API token
API_TOKEN = "your_huggingface_token_here"

# Select the LLAVA model
model_id = "liuhaotian/llava-1.5-13b"

# Define the API URL
API_URL = f"https://api-
inference.huggingface.co/models/{model_id}"
```

```
# Define your prompt
query = "Describe how convolution works in image processing."

# Experiment with different temperature values
temperature = 0.7  # Try 0.1 (more deterministic) or 1.2 (more
creative)

# Prepare the request payload
headers = {"Authorization": f"Bearer {API_TOKEN}"}
payload = {
    "inputs": query,
    "parameters": {
        "temperature": temperature,
        "max_length": 500  # Limit response length
    }
}

# Send the request
response = requests.post(API_URL, headers=headers, json=payload)

# Print the output
print(response.json())
```

Step 5: Modify `temperature` and Experiment

- **Set `temperature=0.1`** → More precise, less creative.
- **Set `temperature=1.2`** → More diverse, creative responses.
- **Try different queries** to see how LLAVA responds with different settings.

Exercices

1. Context Truncation and Its Effects

The objective here is to understand how a model's context length affects response quality.

Instructions:

- Use an AI model (such as LLAVA) with a text-generation playground.
- Provide the following prompt and observe the response:

"*Here is a story beginning: Alice walked into the dark forest, searching for the lost amulet. The trees whispered secrets as she moved forward. Continue the story.*"

- Now, extend the story manually by adding a few more sentences. Then, feed the extended story into the model, asking it to continue again.
- Repeat the process until the text reaches approximately 4,000 tokens (or the model's known context limit).
- Finally, ask the model a question about the first part of the story:

"*What color was Alice's cloak at the beginning?*"

Discussion Questions:

- Does the model correctly remember details from the start?
- At what point does it start forgetting early details?
- How does truncation affect coherence in storytelling?

2. Optimizing Context for Better Responses

Now, the goal is to learn how to optimize prompts when working within a model's context limit.

Instructions:

- Given a long text (e.g., a Wikipedia article or a long customer service chat transcript), try asking the model a question about details from the beginning of the text.
- If the model forgets earlier details, apply the following techniques:
    - Summarization: Reduce the text into key points before feeding it to the model.
    - Chunking + Retrieval: Split the text into sections and query only the most relevant part.
    - Explicit Memory Prompts: Periodically reintroduce key information in the conversation.

  Compare how these techniques improve the AI's accuracy.

Example Scenarios:

- Customer Support: The AI helps a user troubleshoot an issue across multiple chat messages.
- Research Assistant: The AI answers questions about a long scientific paper.

3. Exploring the Effect of Different k Values

In this exercise, we should understand how different top-k values influence the diversity of responses.

- Use a text generation model (e.g., GPT-4 playground or OpenAI API).
- Input the following prompt:

  "Describe a futuristic city in the year 2150."

- Generate responses using **different top-k values**:
    - **k = 5** (low randomness)
    - **k = 20** (medium randomness)

- **k = 50** (high randomness)

- Compare the responses.

- How do the details change as **k increases**?
- Are responses with **low k** more structured but predictable?
- Do responses with **high k** introduce more unusual ideas?

4. **Controlling Creativity in Storytelling**

Now, we should learn how to use **top-k** to balance coherence and creativity.

- Use the following prompt:

  "Write the beginning of a fantasy story where a young explorer finds a hidden portal in the forest."

- Generate responses at different **top-k** values:

  - **k = 3** (very deterministic)
  - **k = 10** (balanced)
  - **k = 50** (highly creative)

- Analyze the differences.

  - Which version is **most structured**?
  - Which version introduces **unexpected elements**?
  - In what cases would a **higher top-k** be preferred?