Hugging Face Transformers Qwen 2.5 Math Model

In this notebook, I will illustrate the capabilities of the `Qwen/Qwen2.5-Math-1.5B-Instruct` model.

The Qwen 2.5 Math model is a state-of-the-art large language model (LLM) specifically designed to solve mathematical problems efficiently and accurately.

Set up Hugging Face Access Token:

```
import os
from google.colab import userdata

# Retrieve the Hugging Face token from the user data
hf_token = userdata.get('HF_Key')  # Make sure 'HF_Key' matches the name you set ir

# Check if the token was retrieved successfully
if hf_token is None:
    raise ValueError("Hugging Face token not found. Please ensure it is set in the

# Set the token as an environment variable (optional)
os.environ['HF_KEY'] = hf_token

# You can now use os.environ['HF_KEY'] wherever you need the token
print("Hugging Face token retrieved and set as an environment variable.")
```

```
Hugging Face token retrieved and set as an environment variable.
```

```
# Install required packages
!pip install huggingface_hub[hf_xet] transformers
```

```
#transformers>=4.37.0 for Qwen2.5-Math models
pip install --upgrade transformers>=4.37.0
```

```
!pip install torch
```

```
import transformers
print(transformers.__version__)
```

```
4.52.2
```

```
#Import necessary libraries
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer
```

Running the model as per the Hugging Face quick start guide to demonstrate its capabilities.

```
model_name = "Qwen/Qwen2.5-Math-1.5B-Instruct"
device = "cuda" # the device to load the model onto

model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype="auto",
    device_map="auto"
)
tokenizer = AutoTokenizer.from_pretrained(model_name)

prompt = "Find the value of $x$ that satisfies the equation $4x+5 = 6x+7$."

# CoT
messages = [
    {"role": "system", "content": "Please reason step by step, and put your final
    {"role": "user", "content": prompt}
]

# TIR
messages = [
    {"role": "system", "content": "Please integrate natural language reasoning wi
    {"role": "user", "content": prompt}
]

text = tokenizer.apply_chat_template(
    messages,
    tokenize=False,
    add_generation_prompt=True
)
model_inputs = tokenizer([text], return_tensors="pt").to(device)
```

```
generated_ids = model.generate(
    **model_inputs,
    max_new_tokens=512
)
generated_ids = [
    output_ids[len(input_ids):] for input_ids, output_ids in zip(model_inputs.inp
]

response = tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: Use
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tak
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access pu
  warnings.warn(
model.safetensors: 100%                    3.09G/3.09G [01:08<00:00, 46.2MB/s]
generation_config.json: 100%               160/160 [00:00<00:00, 16.8kB/s]
tokenizer_config.json: 100%                7.32k/7.32k [00:00<00:00, 549kB/s]
vocab.json: 100%                           2.78M/2.78M [00:00<00:00, 11.8MB/s]
merges.txt: 100%                           1.67M/1.67M [00:00<00:00, 2.44MB/s]
tokenizer.json: 100%                       7.03M/7.03M [00:00<00:00, 16.3MB/s]
```

Experimenting with Prompts: Solving Mathematical Equations

```
# Define the prompt
prompt = "Find the value of $x$ that satisfies the equation $4x+5 = 6x+7$."

# Step a) Define messages for CoT
messages = [
    {"role": "system", "content": "Please reason step by step, and put your final
    {"role": "user", "content": prompt}
]

# Step b) Prepare the input for the model
text = tokenizer.apply_chat_template(
    messages,
    tokenize=False,
    add_generation_prompt=True
)
```

```
# Step c) Print the formatted input text for debugging
print("Formatted Input Text:")
print(text)

model_inputs = tokenizer([text], return_tensors="pt").to(device)

# Step d) Generate the response
generated_ids = model.generate(
    **model_inputs,
    max_new_tokens=512
)

# Print the generated IDs to check if any output was produced
print("Generated IDs:")
print(generated_ids)

# Decode the generated response
generated_ids = [
    output_ids[len(input_ids):] for input_ids, output_ids in zip(model_inputs.inp
]

# Step e)Check if there are any generated IDs before decoding
if generated_ids:
    response = tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]
    print("Model Response:")
    print(response)
else:
    print("No response generated.")
```

```
Formatted Input Text:
<|im_start|>system
Please reason step by step, and put your final answer within \boxed{}.<|im_end
<|im_start|>user
Find the value of $x$ that satisfies the equation $4x+5 = 6x+7$.<|im_end|>
<|im_start|>assistant

Generated IDs:
tensor([[151644,   8948,    198,   5501,   2874,   3019,    553,   3019,      1
            323,   2182,    697,   1590,   4226,   2878,   1124,  79075,  4639
         151645,    198, 151644,    872,    198,   9885,    279,    897,     31
            400,     87,      3,    429,  67901,    279,  23606,    400,      1
             87,     10,     20,    284,    220,     21,     87,     10,      2
          12947, 151645,    198, 151644,  77091,    198,   1249,  11625,     27
          23606,  17767,     19,     87,    488,    220,     20,    284,     22
             21,     87,    488,    220,     22,     59,    701,    582,     68
           1795,    264,   3019,  14319,  29208,   5486,    311,  42123,     27
           3890,  17767,     87,     59,   3593,     16,     13,   3070,    313
           2144,  17767,     19,     87,  57758,    504,   2176,  11067,     31
```

```
                 279,   23606,      25,    1019,     256,    1124,    9640,     256,     22
                  19,      87,     488,     220,      20,     481,     220,      19,      8
                 284,     220,      21,      87,     488,     220,      22,     481,     22
                  19,      87,     198,     256,    1124,     921,     256,   61242,    776
                2176,   11067,      11,     582,     633,     510,     256,    1124,    964
                 256,     220,      20,     284,     220,      17,      87,     488,     22
                  22,     198,     256,    1124,    2533,      17,      13,    3070,    313
                2144,     220,      22,     504,    2176,   11067,     315,     279,   2360
                  25,    1019,     256,    1124,    9640,     256,     220,      20,     48
                 220,      22,     284,     220,      17,      87,     488,     220,      2
                 481,     220,      22,     198,     256,    1124,     921,     256,   6124
                7766,    2176,   11067,      11,     582,     633,     510,     256,    112
                9640,     256,     481,      17,     284,     220,      17,      87,     19
                 256,    1124,    2533,      18,      13,    3070,   12509,     577,    217
               11067,     553,     220,      17,      25,    1019,     256,    1124,    964
                 256,    1124,   37018,   19999,      17,   15170,      17,      92,     28
                1124,   37018,      90,      17,      87,   15170,      17,     532,     25
                1124,     921,     256,   61242,    7766,    2176,   11067,      11,     58
                 633,     510,     256,    1124,    9640,     256,     481,      16,     28
                 856,     198,     256,    1124,    2533,   44500,      11,     279,     89
                 315,   17767,      87,   57758,     429,   67901,     279,   23606,     37
                1124,   11520,   79075,   19999,      16,   11035,     568,  151645]],
           device='cuda:0')
    Model Response:
    To solve the equation \(4x + 5 = 6x + 7\), we will follow a step-by-step appro

    1. **Subtract \(4x\) from both sides of the equation:**
       \[
       4x + 5 - 4x = 6x + 7 - 4x
       \]
       Simplifying both sides, we get:
       \[
       5 = 2x + 7
       \]

    2. **Subtract 7 from both sides of the equation:**
       \[
       5 - 7 = 2x + 7 - 7
       \]
       Simplifying both sides, we get:
```

Another prompt:

```
# Define the new prompt
prompt = "What is the value of $ y $ in the equation $ 3y - 4 = 11 $?"

# SDefine messages for CoT
messages = [
    {"role": "system", "content": "Please reason step by step, and put your final
    {"role": "user", "content": prompt}
```

```
]

# Prepare the input for the model
text = tokenizer.apply_chat_template(
    messages,
    tokenize=False,
    add_generation_prompt=True
)

# Print the formatted input text for debugging
print("Formatted Input Text:")
print(text)

model_inputs = tokenizer([text], return_tensors="pt").to(device)

# Generate the response
generated_ids = model.generate(
    **model_inputs,
    max_new_tokens=512
)

# Print the generated IDs to check if any output was produced
print("Generated IDs:")
print(generated_ids)

# Decode the generated response
generated_ids = [
    output_ids[len(input_ids):] for input_ids, output_ids in zip(model_inputs.inp
]

# Check if there are any generated IDs before decoding
if generated_ids:
    response = tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]
    print("Model Response:")
    print(response)
else:
    print("No response generated.")
```

```
Formatted Input Text:
<|im_start|>system
Please reason step by step, and put your final answer within \boxed{}.<|im_end
<|im_start|>user
What is the value of $ y $ in the equation $ 3y − 4 = 11 $?<|im_end|>
<|im_start|>assistant

Generated IDs:
tensor([[151644,   8948,    198,   5501,   2874,   3019,    553,   3019,     1
            323,   2182,    697,   1590,   4226,   2878,   1124,  79075,  4639
```

```
         151645,    198, 151644,    872,    198,   3838,    374,    279,    89
            315,    400,    379,    400,    304,    279,  23606,    400,    22
             18,     88,    481,    220,     19,    284,    220,     16,     1
          50061, 151645,    198, 151644,  77091,    198,   1249,   1477,    27
            897,    315,  17767,    379,   1124,      8,    304,    279,  2360
          17767,    220,     18,     88,    481,    220,     19,    284,    22
             16,     16,   1124,    701,    582,    686,   1795,    264,   301
          14319,  29208,   5486,   1447,     16,     13,   3070,   3872,  3306
            279,   4647,    448,    279,   3890,  17767,    379,   1124,  3229
            510,    256,   1205,   1191,    553,   7842,    220,     19,    31
           2176,  11067,    315,    279,  23606,    311,  21725,    279,   678
           4647,    389,    279,   2115,   3108,    624,    256,   1124,   964
            256,    220,     18,     88,    481,    220,     19,    488,    22
             19,    284,    220,     16,     16,    488,    220,     19,    19
            256,   1124,    921,    256,  61242,   7766,   2176,  11067,     1
            582,    633,    510,    256,   1124,   9640,    256,    220,     1
             88,    284,    220,     16,     20,    198,    256,   1124,   253
             17,     13,   3070,     50,   3948,    369,  17767,    379,   112
          32295,    510,    256,   9295,     11,    582,  21749,   2176,  1106
            315,    279,  23606,    553,    220,     18,    311,  42123,  1776
            379,   1124,   4292,    256,   1124,   9640,    256,   1124,  3701
             90,     18,     88,  15170,     18,     92,    284,   1124,  3701
             90,     16,     20,  15170,     18,    532,    256,   1124,    92
            256,  61242,   7766,   2176,  11067,     11,    582,    633,    51
            256,   1124,   9640,    256,    379,    284,    220,     20,    19
            256,   1124,   2533,  54815,     11,    279,    897,    315,  1776
            379,   1124,      8,    374,   1124,  11520,  79075,     90,     2
          11035,    568, 151645]], device='cuda:0')
Model Response:
To find the value of \( y \) in the equation \( 3y - 4 = 11 \), we will follow

1. **Isolate the term with the variable \( y \)**:
   We start by adding 4 to both sides of the equation to eliminate the constan
   \[
   3y - 4 + 4 = 11 + 4
   \]
   Simplifying both sides, we get:
   \[
   3y = 15
   \]

2. **Solve for \( y \)**:
   Next, we divide both sides of the equation by 3 to isolate \( y \).
   \[
   \frac{3y}{3} = \frac{15}{3}
   \]
   Simplifying both sides, we get:
   \[
   y = 5
```

Explanation of the Prompt Code

The prompt code involves defining the interaction messages for the model, preparing the input format, generating a response based on the input, and finally decoding the generated output into a readable format. This process allows you to effectively utilize the model to solve mathematical equations or other tasks.

Detailing how to prepare the input for the model, generate a response, and decode the output:

Step a) Define Messages for Chain of Thought (CoT)

```
messages = [
    {"role": "system", "content": "Please reason step by step, and put your fina
    {"role": "user", "content": prompt}
]
```

Explanation:

- In this step, we define a list of messages that will be used to interact with the model.
- The first message is from the "system," instructing the model to reason through the problem step by step and to format the final answer within a LaTeX box (using `\boxed{}`), which is a common way to present mathematical answers.
- The second message is from the "user," containing the actual prompt (the mathematical equation) that you want the model to solve. This structure helps the model understand the context and the task it needs to perform.

Step b) Prepare the Input for the Model

```
text = tokenizer.apply_chat_template(
    messages,
    tokenize=False,
    add_generation_prompt=True
)
```

Explanation:

- Here, you prepare the input for the model using a tokenizer. The `apply_chat_template` method formats the messages into a structure that the model can understand.
- The `tokenize=False` argument indicates that you do not want to tokenize the messages at this stage, while `add_generation_prompt=True` ensures that any necessary prompts for generation are included.
- The resulting `text` variable will contain the formatted input that combines the system and

user messages, making it ready for the model to process.

Step c) Generate the Response

```
generated_ids = model.generate(
    **model_inputs,
    max_new_tokens=512
)
```

Explanation:

- In this step, you call the `generate` method of the model to produce a response based on the prepared input.
- The `model_inputs` variable (which should be defined earlier in your code) contains the tokenized input data.
- The `max_new_tokens=512` argument specifies the maximum number of new tokens that the model can generate in its response, helping to control the length of the output.
- The result, `generated_ids`, will contain the IDs of the tokens generated by the model, which represent the model's response to the prompt.

Step d) Decode the Generated Response

```
generated_ids = [
    output_ids[len(input_ids):] for input_ids, output_ids in zip(model_inputs.in
]
```

Explanation:

- This step involves decoding the generated token IDs back into human-readable text.
- The list comprehension iterates over pairs of `input_ids` (the original input tokens) and `output_ids` (the generated tokens). It slices the `output_ids` to exclude the tokens that correspond to the input, effectively isolating only the newly generated tokens.
- The resulting `generated_ids` list will contain only the tokens that represent the model's response.

Step e) Final Decoding and Output

```
if generated_ids:
    response = tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0
    print("Model Response:")
    print(response)
```

```
else:
    print("No response generated.")
```

Explanation:

- This step checks if there are any generated IDs. If there are, it decodes them into a string using the `batch_decode` method of the tokenizer, which converts the token IDs back into text while skipping any special tokens.
- The decoded response is then printed out. If no response was generated, a message indicating that is printed instead.

Summary

In this notebook, I've explored how to utilize the Hugging Face Qwen 2.5 Math model to solve mathematical equations using a structured prompt code. First, I demonstrated how to define interaction messages that guide the model in reasoning step by step. Next, I prepared the input format to ensure compatibility with the model. Finally, I generated a response based on the provided prompt and decoded the output into a human-readable format.