sql-commandgenerator-oss

November 10, 2024

[1]:
```
!pip install torch transformers bitsandbytes accelerate sqlparse
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages
(2.5.0+cu121)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-
packages (4.44.2)
Collecting bitsandbytes
  Downloading bitsandbytes-0.44.1-py3-none-manylinux_2_24_x86_64.whl.metadata
(3.5 kB)
Requirement already satisfied: accelerate in /usr/local/lib/python3.10/dist-
packages (0.34.2)
Requirement already satisfied: sqlparse in /usr/local/lib/python3.10/dist-
packages (0.5.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from torch) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in
/usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch) (2024.10.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-
packages (from torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy==1.13.1->torch) (1.3.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.24.7)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2024.9.11)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
```

```
packages (from transformers) (2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.4.5)
Requirement already satisfied: tokenizers<0.20,>=0.19 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.19.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-
packages (from transformers) (4.66.6)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages
(from accelerate) (5.9.5)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers)
(2024.8.30)
Downloading bitsandbytes-0.44.1-py3-none-manylinux_2_24_x86_64.whl (122.4 MB)
                         122.4/122.4 MB
7.1 MB/s eta 0:00:00
Installing collected packages: bitsandbytes
Successfully installed bitsandbytes-0.44.1
```

[2]:
```python
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
```

[3]:
```python
torch.cuda.is_available()
```

[3]:
```
True
```

[4]:
```python
available_memory = torch.cuda.get_device_properties(0).total_memory
```

[5]:
```python
print(available_memory)
```

```
15835660288
```

[6]:
```python
# Import required libraries
from transformers import AutoTokenizer, AutoModelForCausalLM
import torch

# Set the model name
model_name = "defog/sqlcoder-7b-2"

# Load the tokenizer
```

```
tokenizer = AutoTokenizer.from_pretrained(model_name)

# Check available memory (assuming available_memory is defined elsewhere in␣
 ↪your code)
if available_memory > 16e9:
    # If you have at least 15GB of GPU memory, load the model in float16
    model = AutoModelForCausalLM.from_pretrained(
        model_name,
        trust_remote_code=True,
        torch_dtype=torch.float16,
        device_map="auto",
        use_cache=True,
    )
else:
    # Otherwise, load in 8 bits (slower but uses less memory)
    model = AutoModelForCausalLM.from_pretrained(
        model_name,
        trust_remote_code=True,
        load_in_8bit=True,
        device_map="auto",
        use_cache=True,
    )
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89:
UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab
and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
  warnings.warn(
```

```
tokenizer_config.json:   0%|          | 0.00/1.84k [00:00<?, ?B/s]
```

```
tokenizer.model:   0%|          | 0.00/500k [00:00<?, ?B/s]
```

```
tokenizer.json:   0%|          | 0.00/1.84M [00:00<?, ?B/s]
```

```
special_tokens_map.json:   0%|          | 0.00/515 [00:00<?, ?B/s]
```

```
config.json:   0%|          | 0.00/691 [00:00<?, ?B/s]
```

```
The `load_in_4bit` and `load_in_8bit` arguments are deprecated and will be
removed in the future versions. Please, pass a `BitsAndBytesConfig` object in
`quantization_config` argument instead.
```

```
model.safetensors.index.json:   0%|          | 0.00/23.9k [00:00<?, ?B/s]
```

```
Downloading shards:   0%|          | 0/3 [00:00<?, ?it/s]
```

```
model-00001-of-00003.safetensors:   0%|          | 0.00/4.94G [00:00<?, ?B/s]

model-00002-of-00003.safetensors:   0%|          | 0.00/4.95G [00:00<?, ?B/s]

model-00003-of-00003.safetensors:   0%|          | 0.00/3.59G [00:00<?, ?B/s]

Loading checkpoint shards:   0%|          | 0/3 [00:00<?, ?it/s]

generation_config.json:   0%|          | 0.00/111 [00:00<?, ?B/s]
```

[7]:
```
# Set the Question & Prompt and Tokenize
prompt = """### Task
Generate a SQL query to answer [QUESTION] (question) [/QUESTION]

### Instructions

- If you cannot answer the question with the available database schema, return␣
  ↪"I do not know."
- Remember that revenue is price multiplied by quantity.
- Remember that cost is supply_price multiplied by quantity.

### Database Schema

This query will run on a database whose schema is represented in this string:

CREATE TABLE products (
    product_id INTEGER PRIMARY KEY,  -- Unique ID for each product
    name VARCHAR(58),                -- Name of the product
    price DECIMAL(10,2),             -- Price of each unit of the product
    quantity INTEGER                 -- Current quantity in stock
);

CREATE TABLE customers (
    customer_id INTEGER PRIMARY KEY, -- Unique ID for each customer
    name VARCHAR(50),                -- Name of the customer
    address VARCHAR(100)             -- Mailing address of the customer
);

CREATE TABLE salespeople (
    salesperson_id INTEGER PRIMARY KEY, -- Unique ID for each salesperson
    name VARCHAR(58),                   -- Name of the salesperson
    region VARCHAR(50)                  -- Geographic sales region
);

CREATE TABLE sales (
    sale_id INTEGER PRIMARY KEY,         -- Unique ID for each sale
    product_id INTEGER,                  -- ID of product sold
    customer_id INTEGER,                 -- ID of customer who made purchase
    salesperson_id INTEGER,              -- ID of salesperson who made the sale
    sale_date DATE,                     -- Date the sale occurred
    quantity INTEGER                    -- Quantity of product sold
);

CREATE TABLE product_suppliers (
    supplier_id INTEGER PRIMARY KEY,    -- Unique ID for each supplier
    product_id INTEGER                  -- ID of product supplied
);

-- Relationships:
-- sales.product_id can be joined with products.product_id
-- sales.customer_id can be joined with customers.customer_id
-- sales.salesperson_id can be joined with salespeople.salesperson_id
-- product_suppliers.product_id can be joined with products.product_id

### Answer

Given the database schema, here is the SQL query that answers [QUESTION]␣
 ↪{question} [/QUESTION]:

[SQL]
"""
```

[8]:
```python
# Generate SQL
import sqlparse


def generate_query(question):
    # Update the prompt with the question
    updated_prompt = prompt.format(question=question)

    # Tokenize the updated prompt
    inputs = tokenizer(updated_prompt, return_tensors="pt").to("cuda")

    # Generate the SQL query
    generated_ids = model.generate(
        **inputs,
        num_return_sequences=1,
        eos_token_id=tokenizer.eos_token_id,
        pad_token_id=tokenizer.eos_token_id,
        max_new_tokens=400,
        do_sample=False,
        num_beams=1,
    )

    # Decode the generated output
    outputs = tokenizer.batch_decode(generated_ids, skip_special_tokens=True)
    # Clear cache to manage memory (useful for Colab)
    torch.cuda.empty_cache()
    torch.cuda.synchronize()

    # Format the SQL output
    return sqlparse.format(outputs[0].split("[SQL]")[-1], reindent=True)
```

[10]:
```
#Question
question = "What was the highest quantity sold last month?"

generated_sql = generate_query(question)

print(generated_sql)

question = "Which salesperson sold large amount of products last month?"

generated_sql = generate_query(question)

print(generated_sql)
```

```
SELECT "name",
       SUM("price" * "quantity") AS total_revenue,
       SUM("supply_price" * "quantity") AS total_cost
FROM products
GROUP BY "name";

SELECT "name",
       SUM("price" * "quantity") AS total_revenue,
       SUM("supply_price" * "quantity") AS total_cost
FROM products
GROUP BY "name";
```

[ ]:
```

```