Adding Web Search To Your LLM

```
from dotenv import load_dotenv
_ = load_dotenv()
```

Asking Your LLM For The Latest Information

```
from utils import query_raven
question = "Hey, can you tell me more about this R1 thing that was announced by Rabbit? "

no_function_calling_prompt = \
f"""
<s> [INST] {question} [/INST]
"""
query_raven(no_function_calling_prompt)
```

```
'I\'m sorry, but I don\'t have any information about a specific "R1" thing that was announced by Rabbit. It\'s possible that you may have misspoken or misunderstood the information you received. If you could provide more context or clarify your question, I would be happy to try and help you.'
```

Providing Up To Date Information

```
import os

def do_web_search(full_user_prompt : str, num_results : int = 5):
    API_URL = f'{os.getenv("DLAI_TAVILY_BASE_URL", "https://api.tavily.com")}/search'
    payload = \
    {
      "api_key": os.environ["TAVILY_API_KEY"],
      "query": full_user_prompt,
      "search_depth": "basic",
      "include_answer": False,
      "include_images": False,
      "include_raw_content": False,
      "max_results": num_results,
      "include_domains": [],
      "exclude_domains": []
    }
    import requests
    response = requests.post(API_URL, json=payload)
    response = response.json()
    all_results = "\n\n".join(item["content"] for item in response["results"])
    return all_results
```

Calling Raven

```
function_calling_prompt = \
"""
Function:
def do_web_search(full_user_prompt : str, num_results : int = 5):
    '''
    Searches the web for the user question.
    '''

Example:
User Query: What is the oldest capital in the world?
Call: do_web_search(full_user_prompt="oldest capital")

User Query: {query}<human_end>
"""
fc_result = query_raven(function_calling_prompt.format(query=question))
print (fc_result)
```

```
do_web_search(full_user_prompt='R1 thing')
```

```
result = eval(fc_result)
```

```
full_prompt = \
f"""
<s> [INST]
{result}

Use the information above to answer the following question concisely.

Question:
{question} [/INST]
"""
```

```
grounded_response = query_raven(full_prompt.format(question = question))
```

```
print (grounded_response)
```

```
The R1 is an AI-powered gadget that can use your apps for you. It's a voice assistant that can answer questions, play music, and mc
```

Chatting With Your SQL Database

> Note below: The database values are randomly generated so your values may differ from those in the video.

```
from utils import create_random_database
create_random_database()
```

```
Toys in database:
(1, 'Mighty Doll', 6.25)
(2, 'Crazy Bear', 13.25)
(3, 'Magic Robot', 19.4)
(4, 'Wonder Car', 15.14)
(5, 'Super Bear', 19.01)
(6, 'Wonder Bear', 15.8)
(7, 'Super Doll', 8.78)
(8, 'Super Robot', 18.72)
(9, 'Wonder Car', 6.85)
(10, 'Happy Car', 11.09)
(11, 'Mighty Doll', 5.38)
(12, 'Magic Dragon', 16.52)
(13, 'Wonder Car', 7.81)
(14, 'Mighty Doll', 7.63)
(15, 'Happy Car', 14.53)
(16, 'Happy Car', 6.08)
(17, 'Crazy Doll', 7.29)
(18, 'Super Bear', 7.9)
(19, 'Happy Bear', 12.83)
(20, 'Crazy Car', 9.16)
(21, 'Mighty Robot', 7.72)
(22, 'Crazy Car', 13.77)
(23, 'Wonder Train', 11.39)
(24, 'Wonder Car', 19.0)
(25, 'Wonder Bear', 9.78)
(26, 'Mighty Doll', 13.8)
(27, 'Crazy Robot', 19.86)
(28, 'Super Train', 5.96)
(29, 'Mighty Bear', 7.9)
(30, 'Magic Bear', 9.6)
(31, 'Mighty Car', 5.8)
(32, 'Super Bear', 8.75)
(33, 'Super Robot', 15.95)
(34, 'Magic Car', 12.71)
(35, 'Crazy Robot', 19.4)
(36, 'Crazy Train', 19.25)
(37, 'Crazy Train', 8.37)
(38, 'Happy Car', 11.37)
(39, 'Crazy Robot', 18.27)
(40, 'Wonder Train', 7.81)
(41, 'Magic Bear', 10.19)
(42, 'Crazy Car', 7.74)
(43, 'Wonder Car', 18.46)
(44, 'Crazy Car', 7.5)
(45, 'Mighty Robot', 13.44)
(46, 'Wonder Bear', 16.91)
(47, 'Magic Dragon', 12.26)
(48, 'Wonder Train', 5.49)
(49, 'Mighty Robot', 5.53)
```

```
(50, 'Happy Robot', 19.91)
(51, 'Wonder Robot', 5.7)
(52, 'Magic Doll', 18.85)
(53, 'Super Car', 13.69)
(54, 'Happy Robot', 11.18)
(55, 'Wonder Bear', 8.7)
(56, 'Wonder Dragon', 13.87)
(57, 'Super Dragon', 7.9)
```

```
question = "What is the most expensive item we currently sell?"
```

```
from utils import execute_sql, query_raven

schema = \
"""
CREATE TABLE IF NOT EXISTS toys (
    id INTEGER PRIMARY KEY,
    name TEXT,
    price REAL
);
"""

raven_prompt = \
f'''
Function:
def execute_sql(sql_code : str):
  """
  Runs sql code for a company internal database
  """

Schema: {schema}
User Query: {question}
'''

output = query_raven(raven_prompt)
print (f"LLM's function call: {output}")
database_result = eval(output)
```

```
LLM's function call: execute_sql(sql_code='SELECT name, price FROM toys ORDER BY price DESC LIMIT 1')
```

```
full_prompt = \
f"""
<s> [INST]
{database_result}

Use the information above to answer the following question concisely.

Question:
{question} [/INST]
"""
grounded_response = query_raven(full_prompt)
print (grounded_response)
```

```
The most expensive item we currently sell is the Crazy Dragon, which costs 19.96.
```

Safer Interactions With Databases

```
import sqlite3
import random

# Internal database name setting
DB_NAME = 'toy_database.db'

# Connect to the database
def connect_db():
    return sqlite3.connect(DB_NAME)
```

```
# List all toys
def list_all_toys():
    with connect_db() as conn:
        cursor = conn.execute('SELECT * FROM toys')
        return cursor.fetchall()
```

```
# Find toy by name prefix
def find_toy_by_prefix(prefix):
    with connect_db() as conn:
        query = 'SELECT * FROM toys WHERE name LIKE ?'
        cursor = conn.execute(query, (prefix + '%',))
        return cursor.fetchall()
```

```
# Find toys in a price range
def find_toys_in_price_range(low_price, high_price):
    with connect_db() as conn:
        query = 'SELECT * FROM toys WHERE price BETWEEN ? AND ?'
        cursor = conn.execute(query, (low_price, high_price))
        return cursor.fetchall()
```

```
# Get a random selection of toys
def get_random_toys(count=5):
    with connect_db() as conn:
        cursor = conn.execute('SELECT * FROM toys')
        all_toys = cursor.fetchall()
        return random.sample(all_toys, min(count, len(all_toys)))
```

```
# Function to get the most expensive toy
def get_most_expensive_toy(count=1):
    with connect_db() as conn:
        cursor = conn.execute(f'SELECT * FROM toys ORDER BY price DESC LIMIT {count}')
        return cursor.fetchone()
```

```
# Function to get the cheapest toy
def get_cheapest_toy(count=1):
    with connect_db() as conn:
        cursor = conn.execute('SELECT * FROM toys ORDER BY price ASC LIMIT {count}')
        return cursor.fetchone()
```

```
raven_prompt = \
f'''
Function:
def list_all_toys():
    """
    Retrieves a list of all toys from the database. This function does not take any parameters.
    Returns: A list of tuples, where each tuple represents a toy with all its attributes (id, name, price).
    """

Function:
def find_toy_by_prefix(prefix):
    """
    Searches for and retrieves toys whose names start with a specified prefix.
    Parameters:
    - prefix (str): The prefix to search for in toy names.
    Returns: A list of tuples, where each tuple represents a toy that matches the prefix criteria.
    """

Function:
def find_toys_in_price_range(low_price, high_price):
    """
    Finds and returns toys within a specified price range.
    Parameters:
    - low_price (float): The lower bound of the price range.
    - high_price (float): The upper bound of the price range.
    Returns: A list of tuples, each representing a toy whose price falls within the specified range.
    """

Function:
def get_random_toys():
    """
    Selects and returns a random set of toys from the database, simulating a "featured toys" list.

    Returns: A list of tuples, each representing a randomly selected toy. The number of toys returned is up to the specified count
    """

Function:
def get_most_expensive_toy(count : int):
```

```
    """
    Retrieves the most expensive toy from the database.
    This function does not take any parameters.

    Returns: A tuple representing the most expensive toy, including its id, name, and price.
    """

Function:
def get_cheapest_toy(count : int):
    """
    Finds and retrieves the cheapest toy in the database.
    This function does not take any parameters.

    Returns: A tuple representing the cheapest toy, including its id, name, and price.
    """

User Query: {question}<human_end>

'''

output = query_raven(raven_prompt)
print (output)
results = eval(output)
```

```
get_most_expensive_toy(count=1)
```

```
full_prompt = \
f"""
<s> [INST]
{database_result}

Use the information above to answer the following question in a single sentence.

Question:
{question} [/INST]
"""
grounded_response = query_raven(full_prompt)
print (grounded_response)
```

```
The most expensive item we currently sell is the Crazy Dragon, which costs 19.96.
```

```
Start coding or generate with AI.
```