LLM630 Compute Kit - StackFlow API LLM Demo

StackFlow

StackFlow is an AI framework developed for the Module LLM and LLM630C Kit. This framework integrates AI algorithms such as speech, vision, and large language models. The LLM630 Compute Kit has StackFlow pre-installed by default. Users can use the LLM630 Compute Kit as an LLM compute server and access it via APIs for LLM inference.

1.Preparation

- Refer to the LLM630 Compute Kit - Config tutorial to learn how to configure network and file transfers on the LLM630 Compute Kit, and obtain the device’ s IP address.

2.Client Program

Download the client test script, ensuring that the PC and the LLM630 Compute Kit are on the same network segment. Run the script, passing in the device IP address as a parameter.

- llm-qwen2.5-1B.py

```
python .\llm-qwen2.5-1B.py --host 192.168.20.24
```

```
Setup LLM...
Setup LLM finished.
```

```python
import socket
import json
import argparse

def create_tcp_connection(host, port):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((host, port))
    return sock


def send_json(sock, data):
    json_data = json.dumps(data, ensure_ascii=False) + '\n'
    sock.sendall(json_data.encode('utf-8'))


def receive_response(sock):
    response = ''
    while True:
        part = sock.recv(4096).decode('utf-8')
        response += part
        if '\n' in response:
            break
    return response.strip()


def close_connection(sock):
    if sock:
        sock.close()


def create_init_data():
    return {
        "request_id": "llm_001",
        "work_id": "llm",
        "action": "setup",
        "object": "llm.setup",
        "data": {
            "model": "qwen2.5-0.5B-prefill-20e",
            "response_format": "llm.utf-8.stream",
            "input": "llm.utf-8.stream",
            "enoutput": True,
            "max_token_len": 1023,
            "prompt": "You are a knowledgeable assistant capable of answering various questions and pro
        }
    }


def parse_setup_response(response_data, sent_request_id):
    error = response_data.get('error')
    request_id = response_data.get('request_id')

    if request_id != sent_request_id:
```

```
        print(f"Request ID mismatch: sent {sent_request_id}, received {request_id}")
        return None

    if error and error.get('code') != 0:
        print(f"Error Code: {error['code']}, Message: {error['message']}")
        return None

    return response_data.get('work_id')


def setup(sock, init_data):
    sent_request_id = init_data['request_id']
    send_json(sock, init_data)
    response = receive_response(sock)
    response_data = json.loads(response)
    return parse_setup_response(response_data, sent_request_id)


def exit_session(sock, deinit_data):
    send_json(sock, deinit_data)
    response = receive_response(sock)
    response_data = json.loads(response)
    print("Exit Response:", response_data)


def parse_inference_response(response_data):
    error = response_data.get('error')
    if error and error.get('code') != 0:
        print(f"Error Code: {error['code']}, Message: {error['message']}")
        return None

    return response_data.get('data')


def main(host, port):
    sock = create_tcp_connection(host, port)

    try:
        print("Setup LLM...")
        init_data = create_init_data()
        llm_work_id = setup(sock, init_data)
        print("Setup LLM finished.")

        while True:
            user_input = input("Enter your message (or 'exit' to quit): ")
            if user_input.lower() == 'exit':
                break

            send_json(sock, {
                "request_id": "llm_001",
                "work_id": llm_work_id,
                "action": "inference",
                "object": "llm.utf-8.stream",
                "data": {
```

```
                    "delta": user_input,
                    "index": 0,
                    "finish": True
                }
            })

            while True:
                response = receive_response(sock)
                response_data = json.loads(response)

                data = parse_inference_response(response_data)
                if data is None:
                    break

                delta = data.get('delta')
                finish = data.get('finish')
                print(delta, end='', flush=True)

                if finish:
                    print()
                    break

        exit_session(sock, {
            "request_id": "llm_exit",
            "work_id": llm_work_id,
            "action": "exit"
        })
    finally:
        close_connection(sock)


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='TCP Client to send JSON data.')
    parser.add_argument('--host', type=str, default='localhost', help='Server hostname (default: localh
    parser.add_argument('--port', type=int, default=10001, help='Server port (default: 10001)')

    args = parser.parse_args()
    main(args.host, args.port)
```

3.Start Interaction

```
Enter your message (or 'exit' to quit): who are you?
I am a large language model created by Alibaba Cloud. I am called Qwen. I am designed to assist with a
```