Paddle OCR Dockerization

Kebli Younes

Dockerfile

The Dockerfile is used to build a Docker image for the application. It starts from a base image `tiangolo/uwsgi-nginx-flask:python3.8`, which is a pre-configured Flask and uWSGI server with Nginx.

1. Updates the package lists for upgrades and new package installations.
2. Installs necessary libraries for the application.
3. Uninstalls the current versions of `protobuf`, `paddlepaddle`, and `openai` packages.
4. Installs specific versions of `protobuf`, `paddlepaddle`, and `openai` packages.
5. Installs other necessary Python packages for the application.
6. Sets an environment variable `PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION` to `python`.
7. Sets the working directory in the container to `/app`.
8. Copies the application code to the `/app` directory in the container.
9. Exposes port 5000 for the application.
10. Sets the command to run the application.

docker-compose.yml

The `docker-compose.yml` file is used to define and run the multi-container Docker application. It specifies the services, networks, and volumes for the application.

The `web` service is built using the Dockerfile. It maps the host's port 5000 to the container's port 5000. It also mounts the current directory on the host to the `/app` directory in the container. The Flask application is set to run on `0.0.0.0`.

app.py

The `app.py` file is the main entry point for the Flask application. It imports necessary modules and sets up a Flask application. It defines a route `/structureocr` that accepts GET requests. The route handler uses the Paddle OCR library to perform OCR on an image and returns the result.

llm_differentiation.py

The `llm_differentiation.py` file is a Python script that uses the OpenAI API to determine the type of a document. It accepts a document as input and returns whether the document type is Imaging, Laboratory, or Vaccination.

1. Sets the OpenAI API key.
2. Converts the input document to a JSON string and truncates it to the first 4000 characters.
3. Defines a prompt for the OpenAI model. The prompt includes a brief explanation of each document type and instructions for the model to return only the document type.
4. Calls the OpenAI API with the prompt and specifies the model, temperature, and maximum number of tokens for the response.
5. Extracts the generated text from the API response.
6. Prints the generated text and returns it prefixed with "Document type: ".

This script is used by the `app.py` file to determine the category of a document.

Usage

To build and run the application, follow these steps:

1. Build the Docker image:

```
    docker-compose build
```

2. Run the Docker container:

```
    docker-compose up
```

The application will be accessible at `http://localhost:5000/structureocr`.

License

This project is licensed under the terms of the MIT license.