ML API 서버 만들기 Demo

Flask, Django

Ask Company 이진석

me@askcompany.kr

Agenda

1. 통합개발환경 / 소스코드 편집기 추천
2. Live Coding Demo : ML 모델 Web API (Feat. Flask)
3. 웹 프레임워크 소개 및 장고
4. Live Coding Demo : ML 모델 Web API (Feat. Django)
5. 배포 소개
6. 컨테이너를 추천합니다.

통합개발환경 / 소스코드 편집기

추천 IDE / 소스코드 편집기

좋은 장비, 좋은 개발환경은 우리의 시간과 정신을 아껴줍니다. 인건비가 제일 비싸요.

- PyCharm Professional
  - 유료
  - 학생 및 교직원을 위한 교육용 라이센스가 1년 단위로 갱신
  - JetBrains 사의 다른 IDE도 추천 : Intelli J 등
- Visual Studio Code
  - 세계 최강 소스코드 편집기
  - 하지만 IDE는 아니예요.

Live Coding Demo

ML 모델 Web API (Feat. Flask)

ML 모델을 구동하는 Web API 서버를 구현해봅시다.

1. 필요한 라이브러리 체크
2. ML 모델 함수 구현
3. ML 모델 테스트
4. 웹서비스 인터페이스 붙이기
5. 클라이언트에서의 호출
    1. Httpie, cUrl, Postman 등
    2. 웹 페이지 (바닐라js, jQuery 등)
    3. 웹 SPA (Single Page Application) : Reactjs, Vuejs, Angular
    4. 모바일 애플리케이션 : iOS/Android 등

STEP #1: 필요한 라이브러리 체크

- requirements.txt 파일

```
flask
pillow
tensorflow
keras
pytest
```

- 설치 명령

```
pip install -r requirements.txt
```

STEP #2: ML 모델 함수 구현 (ml.py)

```python
from tensorflow.keras.models import load_model


# 애플리케이션이 구동될 때, 모델을 전역공간에 로딩
model = load_model("Predict_Model.h5")


def predict(image_path: str) -> int:
    """
    이미지 파일을 읽어 예측한 숫자를 반환
    """

    resized_data = resize_image(image_path)
    res = model.predict(resized_data)
    return int(res.argmax())  # JSON 직렬화를 위해 int64 to int
```

```python
import numpy as np
from PIL import Image


def resize_image(image_path: str) -> np.ndarray:
    """
    이미지 파일을 읽어 grayscale 처리하고 28x28 리사이징한 ndarray 데이터를 반환
    """

    with Image.open(image_path) as im:
        w, h = im.size
        min_size = min(w, h)
        if w > h:
            left, top = (w - min_size) // 2, 0
        else:
            left, top = 0, (h - min_size) // 2
        right = left + min_size
        bottom = top + min_size

        rect = (left, top, right, bottom)
        cropped_im = im.crop(rect).convert("L")  # Grayscale

        resized_im = cropped_im.resize((28, 28))
        resized_pixels = np.resize(resized_im, (1, 784))
        resized_data = ((np.array(resized_pixels) / 255) - 1) * -1

        return resized_data
```

STEP #3: ML 모델 테스트 (test_ml.py)

```python
import pytest
from ml import predict


@pytest.mark.parametrize("image_path, expected", [
    ("./image-data/3.jpg", 3),
    ("./image-data/4.jpg", 4),
])
def test_predict(image_path, expected):
    assert predict(image_path) == expected
```

수행

```
❯ pytest -v
collected 2 items
test_ml.py::test_predict[./image-data/3.jpg-3] PASSED [ 50%]
test_ml.py::test_predict[./image-data/4.jpg-4] PASSED [100%]
```

STEP #4: 웹서비스 인터페이스 붙이기 (app.py)

```python
from typing import Dict
from flask import Flask, jsonify, request
import ml

app = Flask(__name__)

@app.route("/predict", methods=["POST"])
def predict():
    expected: Dict[str, int] = {}

    # 업로드된 모든 파일에 대해서 예측
    for name, f in request.files.items():
        expected[name] = ml.predict(f)

    return jsonify({"expected": expected})
```

STEP #5: 클라이언트에서의 호출

httpie를 활용한 요청 확인

```
❯ http -f POST http://localhost:5000/predict \
    image1@./notebooks/data/4.png \
    image2@./notebooks/data/8.png

{
    "expected": {
        "image1": 4,
        "image2": 8
    }
}
```

웹 프레임워크

다양한 웹 프레임워크

- Python : Django, Flask, Fastapi 등
- Nodejs : Express, Next.js 등
- Ruby : Ruby on rails, Sinatra 등
- PHP : Laravel, Lumen 등
- Java : Spring 등

어떤 언어/프레임워크를 선택해야 하나?

1. 가고자 하는 회사/조직에서 사용하는 언어/프레임워크
    - Java는 ... 할많하않.
2. 초심자는 흥미를 느끼는 언어/프레임워크
    - 주력이 필요합니다. 주력은 바뀔 수 있습니다.
    - 하나를 깊게 파세요. 언어/프레임워크가 아니라 웹을 이해하세요.
3. 배울 자료가 많은 언어/프레임워크
4. 목적에 맞는 언어/프레임워크
    - 뭐든 만들어봐야 실력이 늡니다.
    - 만들고자 하는 목적에 맞는 언어/프레임워크를 선택하세요.
    - 생산성은 개인차가 있지만, 언어/프레임워크에 따라 평균적인 생산성은 몇 배이상 납니다.

Python

- iOS/Android 네이티브 앱 개발을 제외한 거의 모든 영역의 개발이 가능
  - 주력 : 머신러닝, 데이터분석, 웹개발, 자동화 등
- 최고의 표현력을 가진 언어
- 강력한 개발 커뮤니티를 가진 언어
- 느리다?
  - CPU 연산이 C에 비해서 느리지만, 대개의 서비스에서 병목은 CPU 연산이 아니라 Disk/Network I/O, 그리고 우리의 개발력.
  - 파이썬에서 CPU 연산이 많이 필요한 부분은 C/C++로 구현되어있어요. 우리는 CPython을 사용하고 있습니다.
  - C로 짠 모든 코드가 파이썬보다 빠르다고 할 수 있을까? 언어가 아니라 우리의 문제.
  - 우리가 발로 짜는 코드가 문제일 뿐, 우리가 만든 서비스에서 언어/프레임워크 성능이 발목잡는 그런 일은 우리에게 생기지 않습니다.
  - 느려서 서비스에 못 쓰겠다고 하는 사람은 대개 파이썬을 제대로 써보지 않고, 풍문으로만 들은 사람.
  - 선 무당이 사람 잡습니다. 서비스와 언어/프레임워크를 잘 이해하면 문제가 없습니다.

추천 시스템을 위한 어플리케이션 서버 개발 후기 @ kakao - 김광섭

Python 웹프레임워크

- Flask, Fastapi
  - Micro Web Library -> 배우기 쉽다? 빠르다? No. 제공하는 기능이 적다. 여기에도 이런 저런 라이브러리를 붙이다보면 @_@;;;
  - 정말 간단한 API 1~2개만 구현할 경우, 유용
  - 바닥부터 한땀한땀 올리고자 할 경우
- Django
  - Fullstack Web Framework -> 배우기 어렵다? 느리다? No. 제공하는 기능이 많다.
  - 지원 기능: 인증, GIS, 패턴화된 요청 처리, 데이터베이스 연동, 템플릿 엔진, 메모리 캐싱, REST API 등등등

Live Coding Demo

ML 모델 Web API (Feat. django)

순서

1. 필요한 라이브러리 체크
2. 장고 프로젝트 생성
3. 장고 프로젝트 내 장고 앱 app 생성 및 등록
4. ml.py 포팅
5. app/views.py 및 app/urls.py 구현
6. 테스트

STEP #1: 필요한 라이브러리 체크

- requirements.txt 파일

```
django~=3.2.0
pillow
tensorflow
keras
pytest
```

- 설치 명령

```
pip install -r requirements.txt
```

app/ml.py

```
import numpy as np
from pathlib import Path
from PIL import Image
from tensorflow.keras.models import load_model


# app/assets/Predict_Model.h5 경로에 모델 파일 추가하고, 경로 계산 로직 추가
model_path = Path(__file__).parent / "assets" / "Predict_Model.h5"
model = load_model(str(model_path))


def predict(image_path: str) -> int:
    다른 코드들은 동일
```

app/views.py

```
from typing import Dict
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from . import ml


@csrf_exempt
def predict(request):
    expected: Dict[str, int] = {}
    for name, f in request.FILES.items():
        expected[name] = ml.predict(f)
    return JsonResponse(expected)
```

테스트

```
❯ http -f POST http://localhost:8000/predict/ \
    image1@./notebooks/data/4.png \
    image2@./notebooks/data/8.png

{
    "expected": {
        "image1": 4,
        "image2": 8
    }
}
```

배포

배포할려면?

프론트엔드 개발과 백엔드 개발은 다른 세상이고, 클라우드 인프라도 완전히 새로운 세상. @_@;;;

- PaaS 서비스를 추천
  - Heroku : Free Plan도 있어요. 신용카드만 등록하면 750시간/월 제공.
  - Azure Web App for Containers, Google Cloud Run 등등등
- AWS/Google/Azure VM에 배포 : 리눅스에 익숙하지 않다면 삽질이 많습니다.
  - 이는 클라우드를 쓰는 것이 아니라 단순히 VM을 활용하는 레벨. 로컬에 VirtualBox 쓰는 것과 같아요.
  - 클라우드를 쓴다는 것은 클라우드 네트워크와 "관리형 서비스"를 사용하는 것. => 인프라 레벨
- Tip : Djangp/Flask 배포에서 알 수 없는 오류가 발생한다면 Sentry 서비스와 연동해서 에러를 로깅하세요.

어떤 언어/프레임워크/클라우드를 쓰더라도 컨테이너 기술은 사랑입니다.

- 처음부터 컨테이너를 익힐 필요는 없어요. 먼저 개발에 익숙해지는 것이 우선.
- 컨테이너에 익숙해지면 OS에 상관없이 개발환경 구축이 심플해지고, 배포도 단순해집니다.

실습한 플라스크 프로젝트의 경우

```
# Dockerfile
FROM tensorflow/tensorflow

WORKDIR /app
RUN pip3 install flask pillow pytest gunicorn
COPY . .

CMD gunicorn app:app -b :8080

# 쉘에서 빌드 및 실행
docker build -t sample-flask .
docker run --publish 8080:8080 -it sample-flask
```

실습한 장고 프로젝트의 경우

```
# Dockerfile
FROM tensorflow/tensorflow

WORKDIR /app
RUN pip3 install django pillow pytest gunicorn
COPY . .

CMD gunicorn myproj.wsgi:application -b :8080

# 쉘에서 빌드 및 실행
docker build -t sample-django .
docker run --publish 8080:8080 -it sample-django
```

감사합니다.

인생은 짧아요. 여러분의 시간을 아끼세요.

You need Python and Django.