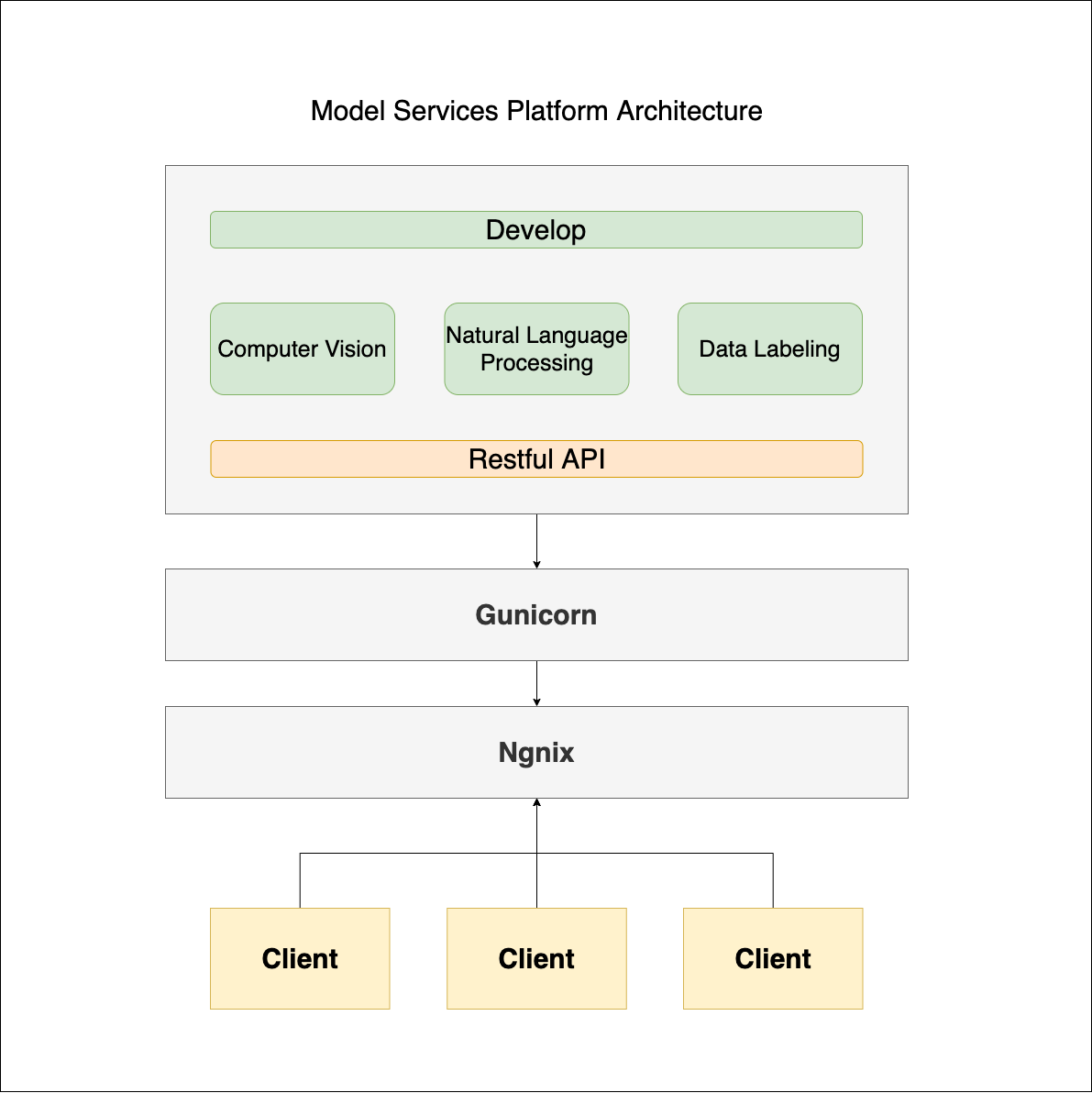
**Model Services**

**一、总架构**



**二、模块**

**1. CV(Computer Vison)模块**

1.1负责计算机视觉领域的模型服务

1.2 接口 172.16.20.21:5000

* 1. 已上线模型

**1.3.1中文文字预训练模型**

**1.3.1.1服务路径**

Flask框架，指定服务器IP和端口 'http://127.0.0.1:5000/'，指定访问路径pretrained\_predict

**1.3.1.2 POST API 设计**

POST接口调用PyTorch\_REST\_API\_URL + "pretrained\_predict"

**1.3.1.3返回数据示例**

{'prediction': [{'label': '一', 'location': [38, 25]}, {'label': '旷', 'location': [73, 51]}, {'label': '吉', 'location': [95, 123]}, {'label': '一', 'location': [181, 44]}, {'label': '削', 'location': [224, 65]}, {'label': '范', 'location': [234, 115]}, {'label': '式', 'location': [265, 41]}], 'success': True}

* + - 1. **Client 端示例**

import base64

import requests

PyTorch\_REST\_API\_URL = 'http://127.0.0.1:5000/'

def tyc\_predict(base64\_data):

r = requests.post(PyTorch\_REST\_API\_URL + "pretrained\_predict", data = {'image': base64\_data}).json()

return r

if \_\_name\_\_ == '\_\_main\_\_':

with open("/home/ruochi/Downloads/1.png","rb") as f:

base64\_data = base64.b64encode(f.read())

print(pretrained\_predict(base64\_data))

* + 1. **天眼查 OCR 识别模型**

**1.3.2.1服务路径**

Flask框架，指定服务器IP和端口 'http://127.0.0.1:5000/'，指定访问路径tyc\_predict

**1.3.2.2 POST API 设计**

POST接口调用PyTorch\_REST\_API\_URL + "tyc\_predict"

**1.3.2.3返回数据示例**

{'prediction': [{'label': '三', 'location': [38, 25]}, {'label': '解', 'location': [73, 51]}, {'label': '害', 'location': [95, 123]}, {'label': '乏', 'location': [181, 44]}, {'label': '勘', 'location': [224, 65]}, {'label': '潜', 'location': [234, 115]}, {'label': '谗', 'location': [265, 41]}], 'success': True}

**1.3.2.4Client 端示例**

import base64

import requests

PyTorch\_REST\_API\_URL = 'http://127.0.0.1:5000/'

def tyc\_predict(base64\_data):

r = requests.post(PyTorch\_REST\_API\_URL + "tyc\_predict", data = {'image': base64\_data}).json()

return r

if \_\_name\_\_ == '\_\_main\_\_':

with open("/home/ruochi/Downloads/1.png","rb") as f:

base64\_data = base64.b64encode(f.read())

* + 1. **国家企业信息信用中心 OCR 模型**

**1.3.3.1服务路径**

Flask框架，指定服务器IP和端口 'http://127.0.0.1:5000/'，指定访问路径gj\_predict

**1.3.3.2 POST API 设计**

POST接口调用PyTorch\_REST\_API\_URL + "gj\_predictt"

**1.3.3.3返回数据示例**

{'prediction': [{'target': '望远镜'}, {'index': 2, 'location': [116, 0, 227, 111]}, {'index': 7, 'location': [0, 232, 111,343]}, {'index': 8, 'location': [116, 232, 227, 343]}], 'success': True}

* + - 1. **Client 端示例**

import base64

import requests

PyTorch\_REST\_API\_URL = 'http://127.0.0.1:5000/'

def tyc\_predict(base64\_data):

r = requests.post(PyTorch\_REST\_API\_URL + "gj\_predict", data = {'image': base64\_data}).json()

return r

if \_\_name\_\_ == '\_\_main\_\_':

with open("/home/ruochi/Downloads/2.png","rb") as f:

base64\_data = base64.b64encode(f.read())

print(gj\_predict\_predict(base64\_data))

* + 1. **国家专利局 OCR 模型**

**1.3.4.1服务路径**

Flask框架，指定服务器IP和端口 'http://127.0.0.1:5000/'，指定访问路径patent\_predict

**1.3.4.2 POST API 设计**

POST接口调用PyTorch\_REST\_API\_URL + "patent\_predict"

**1.3.4.3返回数据示例**

{'prediction': [{'label': '一', 'location': [38, 25]}, {'label': '旷', 'location': [73, 51]}, {'label': '吉', 'location': [95, 123]}, {'label': '一', 'location': [181, 44]}, {'label': '削', 'location': [224, 65]}, {'label': '范', 'location': [234, 115]}, {'label': '式', 'location': [265, 41]}], 'success': True}

**1.3.4.4 Client 端示例**

import base64

import requests

PyTorch\_REST\_API\_URL = 'http://127.0.0.1:5000/'

def patent\_predict(base64\_data):

r = requests.post(PyTorch\_REST\_API\_URL + "patent\_predict", data = {'image': base64\_data}).json()

return r

with open("/home/ruochi/Documents/share/CV/OCR/patent/raw\_pic/17.png", "rb") as f:

base64\_data = base64.b64encode(f.read())

print(patent\_predict(base64\_data))

**2. NLP( Natural Language Processing) 模块**

1. 负责自然语言处理领域的模型服务

2. 接口 172.16.20.21:5001

1. 已上线模型

**2.3.1 Bert 语言模型**

**2.3.1.1服务路径**

Flask框架，指定服务器IP和端口 'http://127.0.0.1:5001/'，指定访问路径bert\_embedding

**2.3.4.2 POST API 设计**

POST接口调用PyTorch\_REST\_API\_URL + "bert\_embedding"

**2.3.4.3返回数据示例**

{“embedding”: list[list ....], “success”: True/False}

**2.3.4.4 Clinet 端示例**

import base64

import requests

from pathlib import Path

from PIL import Image

import matplotlib.pyplot as plt

import numpy as np

PyTorch\_REST\_API\_URL = 'http://172.16.20.21:5001/'

def pretrained\_predict(doc):

# Submit the request.

r = requests.post(PyTorch\_REST\_API\_URL + "bert\_embedding", data = {'doc': doc}).json()

return r

if \_\_name\_\_ == "\_\_main\_\_":

with open("test\_doc.txt","r") as f:

all\_docs = f.readlines()

sentences = all\_docs[0].split("；")

## predict single sentence

res = pretrained\_predict(sentences[0])

print(res["embedding"])

# ## predict doc (concat a list of sentences to a string)

# ## concatenating them with ||| (with whitespace before and after)

doc = " ||| ".join(sentences)

# print(doc)

res = pretrained\_predict(doc)

print(res["embedding"])

**3. DL(Data Labeling) 模块**

**3.1 负责数据标注领域的模型服务**

**3.2 接口 172.16.20.21:5002**

**3.3在标注验证码之前，会利用我们现在已有的模型进行切割和预标记，以减少人工标记成本。**

**3.3.1模块功能：**

（1）切割

（2）提供前端预标注

（3）打包分发（待完成）

（4）统计（待完成）

**3.3.2构建 YoloV4 的文字切割模块**

app\yolov4\_pytorch\yolo.py中提供了接口predict\_one(self, image)，和faster-rcnn类似，返回boxes。之后利用boxes进行切割。

**3.3.3构建文字预标注模型服务**

app\labeling\_app.py端口为5002，会访问接口tyc\_single\_predict()来进行单个字的预测

**3.3.4使用方法**

首先进行切割services\labelingPlatfrom\split\_save\_tyc.py，然后需要运行services\app\labeling\_app.py打开服务，端口为5002。然后运行labelingPlatfrom\label\_tyc\_data.py进行预标注

**4. DEV(Develop) 模块**

1. 供算法部门开发测试

2. 接口 172.16.20.21:5003

**三、服务器端技术**

**1. Gunicorn：**是一种Python WSGI UNIX HTTP Server。

功能：进程管理

协议转换

gunicorn -w 4 -b 0.0.0.0:5000 app:app

-w 并行工作线程数，-b 是设置监听IP和端口，app:变量名,python文件中可调用的wsgi接口名称

**2.Nginx**：Nginx 在启动后，会有一个 master 进程和多个相互独立的 worker 进程。

sudo service nginx restart

功能：资源缓存

负载均衡

反向代理

缓存请求

提高并发

1. **APIs**：
2. **load：加载模型**

**CV:**

model\_name：

- "tian\_yan\_cha"

- "pretrained\_model"

- "xin\_yong\_xin\_xi\_zhong\_xin"

- "patent"

PyTorch\_REST\_API\_URL = 'http://172.16.20.21:5000/'

r = requests.post(PyTorch\_REST\_API\_URL + "load\_model",

data={'model\_name': model\_name}).json()

**NLP:**

model\_name：

- "bert\_embedding"

./run\_bert.sh

1. **unload：卸载模型**

model\_name：

- "tian\_yan\_cha"

- "pretrained\_model"

- "xin\_yong\_xin\_xi\_zhong\_xin"

- "patent"

PyTorch\_REST\_API\_URL = 'http://172.16.20.21:5000/'

r = requests.post(PyTorch\_REST\_API\_URL + "unload\_model",

data={'model\_name': model\_name}).json()

1. **Predict**

**CV:**

PyTorch\_REST\_API\_URL = 'http://172.16.20.21:5002/'

r = requests.post(PyTorch\_REST\_API\_URL + "pretrained\_predict", data = {'image': base64\_data}).json()

r = requests.post(PyTorch\_REST\_API\_URL + "tyc\_predict", data = {'image': base64\_data}).json()

r = requests.post(PyTorch\_REST\_API\_URL + "gj\_predict", data={'image': base64\_data}).json()

r = requests.post(PyTorch\_REST\_API\_URL + "patent\_predict",

data={'image': base64\_data}).json()

**NLP:**

PyTorch\_REST\_API\_URL = 'http://172.16.20.21:5001/'

r = requests.post(PyTorch\_REST\_API\_URL + "bert\_embedding", data = {'doc': doc}).json()