

实验复现报告

付翔

2024 年 12 月 8 日

目录

1	图像增强	2
2	文本到语音转换	2
3	文本到图片转换	3
4	光学字符识别	4
5	内容审核	4
6	Python 代码	5

1 图像增强



(a)

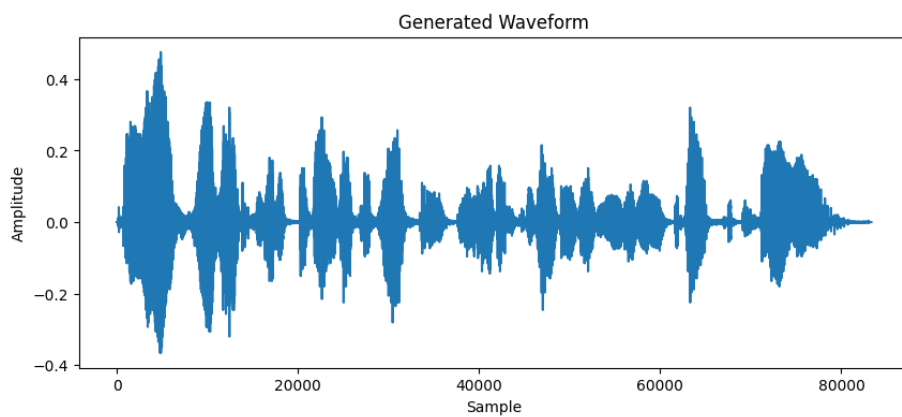


(b)

2 文本到语音转换

输入文本:

text = "Hello, this is a test of speech synthesis using Tacotron2."



3 文本到图片转换

输入文本:

Astronaut riding a horse.



4 光学字符识别



(a)



(b)

图片 2a描述: a man and woman standing on a cliff overlooking a lake

图片 2b描述: a wolf is walking through a hole in the snow

5 内容审核

输入文本:

"Those people are all garbage and don't deserve to live, You idiot, you can't do anything well."

类别	toxic	severe_toxic	obscene	threat	insult	identity_hate
是否存在	True	False	False	False	True	False

表 1: 内容判定情况

6 Python 代码

#####推荐使用虚拟环境，安装所需的包，避免出错。

#图像增强 (cmd)——

```
git clone https://github.com/xinntao/Real-ESRGAN.git
cd Real-ESRGAN
```

```
# 安装 basicsr - https://github.com/xinntao/BasicSR
```

```
# 我们使用 BasicSR 来训练以及推断
```

```
pip install basicsr
```

```
# facexlib 和 gfpgan 是用来增强人脸的
```

```
pip install facexlib
```

```
pip install gfpgan
```

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

```
python setup.py develop
```

```
wget https://github.com/xinntao/Real-ESRGAN/releases/download/v0.1.0/RealESRGAN_x4plus.pth
```

```
#python inference_realesrgan.py -n RealESRGAN_x4plus -i inputs --face_enhance
```

```
python inference_realesrgan.py -n RealESRGAN_x4plus -i photo1.jpg --face_enhance
```

#文本到语音转换——

```
import torch
```

```
import torchaudio
```

```
import matplotlib.pyplot as plt
```

```
from torchaudio.models import Tacotron2
```

```
from torchaudio.pipelines import TACOTRON2_WAVERNN_PHONE_LJSPEECH
```

```
# 加载 Tacotron2 模型和 WaveRNN 声码器
```

```
bundle = TACOTRON2_WAVERNN_PHONE_LJSPEECH
```

```

processor = bundle.get_text_processor()
tacotron2 = bundle.get_tacotron2()
vocoder = bundle.get_vocoder()

# 将模型设置为评估模式
tacotron2.eval()
vocoder.eval()

# 输入文本
text = "Hello , this is a test of speech synthesis using Tacotron2."

# 处理文本
with torch.inference_mode():
    processed, lengths = processor(text)

    # 生成梅尔频谱
    mel_spec, mel_lengths, _ = tacotron2.infer(processed, lengths)

    # 使用声码器生成波形
    waveforms, lengths = vocoder(mel_spec, mel_lengths)

# 可视化波形
plt.figure(figsize=(10, 4))
plt.plot(waveforms[0].cpu().numpy())
plt.title("Generated Waveform")
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.show()

# 保存音频文件
output_path = "output.wav"
torchaudio.save(output_path, waveforms.cpu(), sample_rate=22050)
print(f"Audio has been saved to {output_path}")

```

#文本到图片转换

```
import requests
import io
from PIL import Image

#API_URL = "" # 你的API地址
#headers = {} # 你的请求头
def query(payload):
    response = requests.post(API_URL, headers=headers, json=payload)
    return response.content

image_bytes = query({
    "inputs": "Astronaut_riding_a_horse", # 输入英文提示词
})

image = Image.open(io.BytesIO(image_bytes))
```

#光学字符识别

```
from transformers import BlipProcessor, BlipForConditionalGeneration
from PIL import Image
import requests

# 加载BLIP模型和处理器
processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-
```

```

# 读取图片
image_path = 'cc2.jpg' # 替换为你的图片路径
image = Image.open(image_path)

# 预处理图片
inputs = processor(images=image, return_tensors="pt")

# 生成描述
out = model.generate(**inputs)

# 解码并输出描述
description = processor.decode(out[0], skip_special_tokens=True)
print(" 图片描述:", description)

#内容审核
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.multioutput import MultiOutputClassifier
from sklearn.metrics import classification_report

# 加载数据集
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')
test_labels_df = pd.read_csv('test_labels.csv')
sample_submission_df = pd.read_csv('sample_submission.csv')

# 随机提取 100000 个样本
train_sample = train_df.sample(n=100000, random_state=12)
test_sample = test_df.sample(n=100000, random_state=12)

# 注意, test_labels 中的 ID 需要与 test_sample 中的 ID 匹配

```



```

test_labels_sample = test_labels_df[test_labels_df['id'].isin(test_sample['id'])]

# sample_submission中的ID也需要与test_sample中的ID匹配
sample_submission_sample = sample_submission_df[sample_submission_df['id'].isin(test_sample['id'])]

# 保存样本数据
train_sample.to_csv('train_sample.csv', index=False)
test_sample.to_csv('test_sample.csv', index=False)
test_labels_sample.to_csv('test_labels_sample.csv', index=False)
sample_submission_sample.to_csv('sample_submission_sample.csv', index=False)

# 加载样本数据集
train_sample = pd.read_csv('train_sample.csv')

# 提取特征和标签
X = train_sample['comment_text']
y = train_sample[['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_harassment']]

# 分割数据集
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# 特征提取
vectorizer = TfidfVectorizer(max_features=10000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_val_tfidf = vectorizer.transform(X_val)

# 训练多标签分类模型
model = MultiOutputClassifier(LogisticRegression(max_iter=1000))
model.fit(X_train_tfidf, y_train)

```

```

# 预测和评估
y_pred = model.predict(X_val_tfidf)
print(classification_report(y_val, y_pred, target_names=y.columns))

def predict_labels(comment):
    comment_tfidf = vectorizer.transform([comment])
    prediction = model.predict(comment_tfidf)
    labels = y.columns
    return {label: bool(pred) for label, pred in zip(labels, prediction[0])}

# 示例句子
comment = "Those people are all garbage and don't deserve to live"
prediction = predict_labels(comment)
print(prediction)

# 示例句子
comment = "You idiot, you can't do anything well."
prediction = predict_labels(comment)
print(prediction)

```