

Capstone Design Individual Report

컴퓨터학부 심화컴퓨터 전공 2020110206

오영선(1팀)

저의 주 역할은 모델 개발, 회의록 기록으로 종합설계 프로젝트 1을 하며 개발한 과정과 성과, 그 외 맡았던 역할에 대해 작성하였습니다.

목차 :

1. 사전 조사, 수행계획서 작성 및 일정표 작성
2. 개발환경 구축, 포트포워딩 설정, 데이터 전처리
3. OCR 학습모델 선정 및 선행 연구 조사
4. 학습 데이터 전처리
5. 모델 학습, 수행테스트 및 트러블슈팅
6. 모델 학습 성과
7. 모델 실행 파일 작성
8. 회의록 기록(역할)

1. 사전 조사, 수행계획서 작성 및 일정표 작성

첫 미팅 후 앞으로의 수행계획서 작성 및 모델 학습을 위해 사전조사를 진행하였습니다.

조사한 내용은 크게 1. 손글씨의 중요성 2. OCR을 학습에 사용한 사례, 3. 맞춤법API로 나누어 조사하였으며, 소재목으로 분류하여 알아보기 쉽게 정리한 후 팀원들과 함께 공유하여 조사한 내용을 토대로 다같이 수행계획서를 작성하였습니다.

<https://www.notion.so/8cbe1852ce0047e299f92dc38ee6088e?pvs=4>

오영선 자료조사

koreascience.kr
<https://koreascience.kr/article/CFKO201930060791848.pdf>

한글 OCR 성능을 높이기 위해서 문자인식 부분에 초점을 맞추어 연구 방향을 설정하였다.

<http://koreascience.or.kr/article/CFKO201821464987120.pdf>

https://github.com/parksunwoo/ocr_kor

딥러닝을 활용한 한글문서 OCR 연구

https://github.com/parksunwoo/ocr_kor/blob/master/document/HCLT2019_deeplearningOCR.pdf

한글 OCR 구현코드

OCR API

카카오 OCR Api

<https://mskim8717.tistory.com/49>

네이버 OCR Api

<https://www.ncloud.com/product/aiService/ocr>

교육 방향성

한글 맞춤법 사용 실태 및 맞춤법 교육의 문제 : 고등학생을 중심으로

<https://scienceon.kisti.re.kr/srch/selectPORSrchArticle.do?cn=DIKO0009984665>

띄어쓰기의 중요성

<https://koreascience.kr/article/CFKO200908355727731.page>

손글씨의 중요성

[<https://www.sciencetimes.co.kr/news/성적-올리려면-손으로-필기하라/>]

맞춤법 검사API

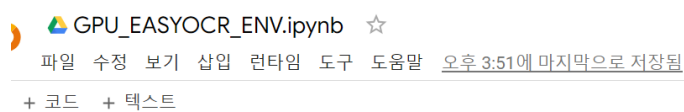
https://console.kakaioi.io/docs/posts/aiservice-nlp/kgc/2022-05-26-aiservice-kgc_api/aiservice-kgc_api#맞춤법-검사-api-reference

<https://acdongpgm.tistory.com/242>

이후 각자의 역할분배가 완료 된 후, 일정표를 작성하여 팀원과 대학원생에게 해당 내용을 공유하였습니다.



또, 쿠다 드라이브 설치가 필요해 쿠다의 버전을 확인 후, 맞는 드라이버를 찾아 부팅해주었습니다. 드라이브 설치 후에도 여러 번 드라이브가 맞지 않다는 예러가 떠, safe mode를 해제 한 후 다시 버전에 맞춰 설치해주어야 했습니다.



GPU 설정하기

런타임 -> 런타임 유형 설정 -> GPU 선택

▶ 파이썬 버전 3.9 설정하기

<https://velog.io/@rubying/LAB-Google-Colab-Python-%EB%B2%84%EC%A0%84-%EC%97%85%EA%B7%B8%EB%A0%88%EC%9D%B4%EB%93%9C>

```
[ ] 1 !python -V

Python 3.10.11

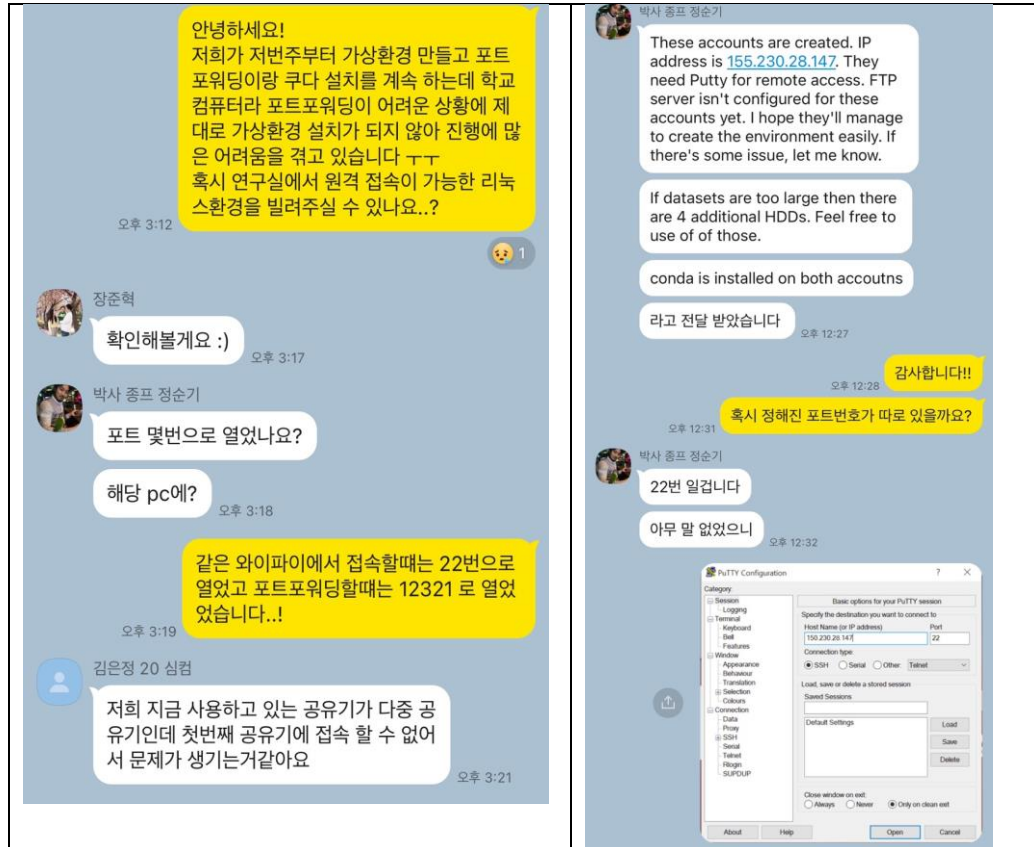
[ ] 1 !wget https://www.python.org/ftp/python/3.3.6/Python-3.3.6.tgz
2 !tar Python-3.3.6.tgz
3 !Python-3.3.6/configure
4 !make
5 !sudo make install

--2023-05-07 11:24:01-- https://www.python.org/ftp/python/3.3.6/Python-3.3.6.tgz
Resolving www.python.org (www.python.org)... 100.222.222.222
Connecting to www.python.org (www.python.org):80... connected.
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 1048576
Content-Disposition: attachment; filename=Python-3.3.6.tgz
100% |#####| 1048576 bytes received in 10s (104.86 KB/s)
Saving Python-3.3.6.tgz to ./
Python-3.3.6.tgz saved. [100%]
```

https://colab.research.google.com/drive/1nk7eATLSPvg3ZIsL0brXaCSfqrWmdE_2

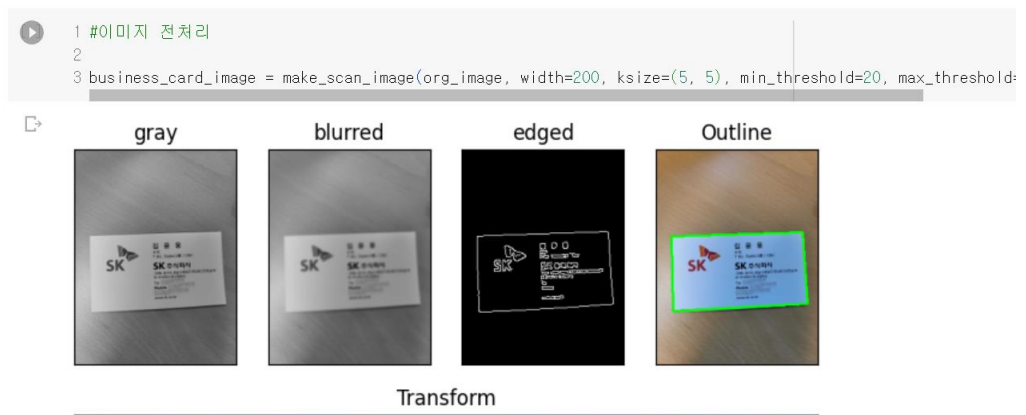
B. 포트 포워딩

다음으로 모델을 개발하기 전, 서버 배포를 고려해 GPU가 있는 컴퓨터 환경에 외부 접속이 가능하도록 포트포워딩 설정이 필요했습니다. 저희팀에서 1차 개발에 사용한 컴퓨터는 학교에서 사용하고 있는 컴퓨터였는데, 해당 컴퓨터의 네트워크가 학교 네트워크로 연결되어 있어 여러 번 포트포워딩 시도를 해주었음에도 보안으로 인해 포트포워딩 접속이 불가능했습니다. 또 알아본 결과, 컴퓨터가 여러 라우터의 서브 라우터로 설정되어있어 해당 컴퓨터에서는 포트포워딩이 어려웠고, 대학원분들께 도움을 구해 해당 연구실의 컴퓨터의 권한을 받았습니다. 이후 연구실의 컴퓨터에 포트포워딩을 설정하여 서버 배포, 모델학습이 가능하도록 하였습니다.



C. 데이터 전처리

easyOCR모듈을 돌리기 전 사진으로 찍은 이미지를 OCR하기 쉽도록 전처리하는 과정이 필요했습니다. 따라서 grayscale, GaussianBlur, Canny, outline 과정을 거쳐 총 4단계에 거쳐 인식시킬 이미지를 만드는 코드를 작성했습니다.

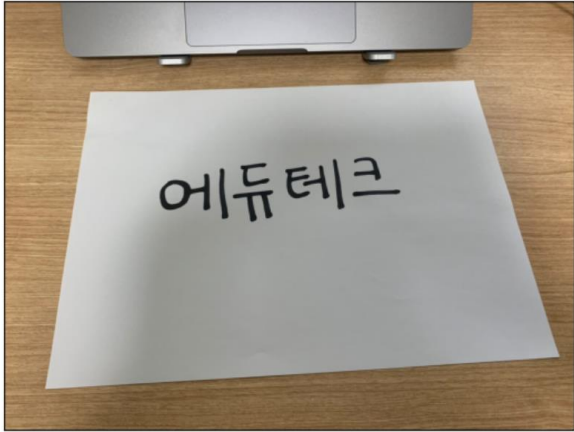


```

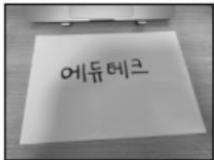
1 url = '/content/test.jpg'
2
3 org_image = cv2.imread(url, cv2.IMREAD_COLOR)
4 plt.imshow("original image", org_image)

```

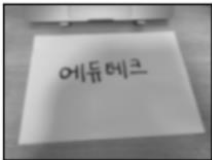
original image




gray



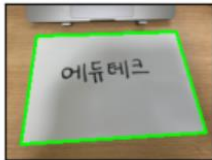
blurred



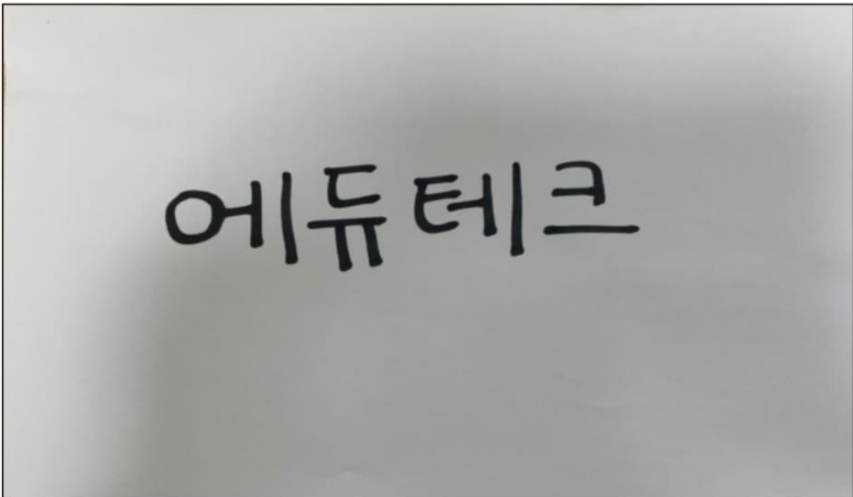
edged



Outline



Transform



직접 사진을 찍은 뒤 업로드하여 전처리를 해보았습니다.

3. OCR 학습모델 선정 및 선행 연구 조사

OCR모델을 사용하기 전 오픈소스 OCR API를 조사한 결과 Tesseract와 EasyOCR 두가지를 대표적으로 사용해 추가적으로 데이터를 학습시킬 수 있음을 확인하고, 해당 API들에

대해 학습시키는 방법을 조사하고 기록하였습니다.

OCR 모델 선행 연구 조사

테서렉트를 통한 오프라인 문자 인식, 문자 학습

<https://secmem.tistory.com/489>

- 오프라인 문자인식 : OCR
- 이진화, 윤곽선 추출 후 형태분석
- 오프라인 대표 오픈소스 : 테서렉트

테서렉트 향상법

1. 추가적인 보정작업
2. 자체 데이터 학습 기능

Tesseract OCR 4.0 학습 4.0 학습

<https://m.blog.naver.com/jerry1455/221412511622>

- a. [lang],[fontname].exp[num].tif 으로 이미지 파일 이름 처리
 - a. lang : OCR 사용시 불러올 문자열?????(mykor)
 - b. fontname : 사용자가 만든 폰트이름 - handwriting
 - c. num : file index
- b. 학습을 위한 이미지로부터 box 생성
 - a. tesseract.exe mykor.hand.exp0.tif mykor.hand.exp0 batch.nochop makebox
 - b. mykor.hand.exp0.box 파일이 생성된다.
- c. tesseract.exe mykor.hand.exp0.tif mykor.hand.exp0 nobatch box.train 으로 학습시킴
 - a. mykor.hand.exp0.tr 파일, mykor.hand.exp0.txt 파일 생성되고
 - b. unicharset 파일 생성하기(테서렉트가 출력가능한 모든 문자들의 집합파일)

EasyOCR 사용자 모델 학습하기

<https://davelogs.tistory.com/76>

1. EasyOCR에서 사용하고 있는 신경망 모델은 학습의 각 단계별로 다음과 같은 또 다른 오픈소스 기반 프로젝트를 이용한다.
- 학습데이터 생성: [TextRecognitionDataGenerator](#)
- 학습데이터 변환: [TRDG2DTRB](#)
- 모델 학습 및 배포: [Deep-Text-Recognition-Benchmark](#)
- 사용자 학습 모델 사용: [EasyOCR](#)

TesseractOCR, EasyOCR 실행과 학습

<https://velog.io/@mminjg/TesseractOCR-EasyOCR-실행과-학습>

1. 티켓 정보를 바탕으로 하는 공연 후기 서비스 OCR
2. Docker 사용
3. Tesseract, OpenCV, EasyOCR
4. <https://github.com/TCAT-capstone/ocr-preprocessor> 전처리 코

kocrnn: CRNN 기반의 한글 텍스트 인식 모델 학습

<https://github.com/apphia39/pghj-kocrnn/tree/main>

<https://www.notion.so/OCR-7511632133bb4430b591cd22b2de8e4d?pvs=4>

다음으로는 Tesseract와 easyOCR의 성능을 비교 후 더 나은 성능을 보이는 EasyOCR을 택하였습니다. 코랩에서 각 모듈을 설치 후 여러장의 이미지로 demo를 돌려본 결과, 전반적으로 EasyOCR이 더 좋은 성능을 보이는 것을 확인하였습니다.

Tessract demo

<https://colab.research.google.com/drive/1EMB8wkNfBlOjE1mHoGwrISEzLwELqGgl>

EasyOCR demo

https://colab.research.google.com/drive/16Shq7JRmYOc_-WUnqqnGVn3pvx8qTkPG

비교 코드

https://colab.research.google.com/drive/1_NnnkPRWHCEOrjdjxH7XQy77-Au4KX-B

compare Tesseract vs Easyocr.ipynb

파일

수정

보기

삽입

런타임

도구

도움말

6월 16일에 마지막으로 수정됨

+ 코드

+ 텍스트

연결

1

##easy OCR

2

from imutils.perspective import four_point_transform

3

from imutils.contours import sort_contours

4

import imutils

5

from easyocr import Reader

6

import cv2

7

import numpy as np

8

from PIL import ImageFont, ImageDraw, Image

9

10

from google.colab.patches import cv2_imshow

11

12

path = '/content/drive/MyDrive/Colab Notebooks/New_sample/원천데이터/01_handwriting_sentence_images/1_sentence/00000019.png'

13

14

image = cv2.imread(path, cv2.IMREAD_COLOR)

15

path = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

16

cv2_imshow(image) # 이미지를 보여주는 코드로 생략가능

17

18

langs = ['ko', 'en']

19

20

print("[INFO] OCR'ing input image...")

21

reader = Reader(lang_list=langs, gpu=True)

22

results = reader.readtext(image)

23

24

simple_results = reader.readtext(image, detail = 0)

25

simple_results #인식된 글자만 추출

부산저 축은행그룹이 금융감독기관의 수장인 금감원장에게 로비를 벌

부산저 축은행그룹이 금융#특기관외 42 중정원장에게 교비를 벌

WARNING:easyocr.easyocr:Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU.

WARNING:easyocr.easyocr:Downloading detection model, please wait. This may take several minutes depending upon your network connection.

[INFO] OCR'ing input image...

Progress: [100.0% Complete]WARNING:easyocr.easyocr:Downloading recognition model, please wait. This may take several minutes depending upon your network connection.

Tesseract	<div> <div>Tesseract_OCR_Test.ipynb</div> <div> <div>파일</div> <div>수정</div> <div>보기</div> <div>삽입</div> <div>런타임</div> <div>도구</div> <div>도움말</div> <div>6월 16일에 마지막으로 수정됨</div> </div> <div> <div>+ 코드</div> <div>+ 텍스트</div> </div> <div> <div> <div>1</div> <div>!sudo apt install tesseract-ocr</div> </div> <div> <div>2</div> <div>!sudo apt install libtesseract-dev</div> </div> </div> <div> <div>Reading package lists... Done</div> <div>Building dependency tree</div> <div>Reading state information... Done</div> <div>The following additional packages will be installed:</div> <div>tesseract-ocr-eng tesseract-ocr-osd</div> <div>The following NEW packages will be installed:</div> <div>tesseract-ocr tesseract-ocr-eng tesseract-ocr-osd</div> <div>0 upgraded, 3 newly installed, 0 to remove and 23 not upgraded.</div> <div>Need to get 4,850 kB of archives.</div> <div>After this operation, 16.3 MB of additional disk space will be used.</div> <div>Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 tesseract-ocr-eng all 1:4.00-git30-7274cfa-1 [1,598 kB]</div> <div>Get:2 http://archive.ubuntu.com/ubuntu focal/universe amd64 tesseract-ocr-osd all 1:4.00-git30-7274cfa-1 [2,990 kB]</div> <div>Get:3 http://archive.ubuntu.com/ubuntu focal/universe amd64 tesseract-ocr amd64 4.1.1-2build2 [262 kB]</div> <div>Fetched 4,850 kB in 1s (4,768 kB/s)</div> <div>debconf: unable to initialize frontend: Dialog</div> <div>debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 76, <?>)</div> <div>debconf: falling back to frontend: Readline</div> </div> </div>
EasyOCR	<div> <div>easyOcr_demo.ipynb</div> <div> <div>파일</div> <div>수정</div> <div>보기</div> <div>삽입</div> <div>런타임</div> <div>도구</div> <div>도움말</div> <div>6월 16일에 마지막으로 수정됨</div> </div> <div> <div>+ 코드</div> <div>+ 텍스트</div> </div> <div> <div>필요 패키지 설치</div> </div> <div> <div> <div>1</div> <div>!pip install easyocr</div> </div> <div> <div>2</div> <div>!pip install opencv-python-headless==4.1.2.30</div> </div> </div> <div> <div>Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/</div> <div>Collecting easyocr</div> <div>Downloading easyocr-1.6.2-py3-none-any.whl (2.9 MB)</div> <div>Requirement already satisfied: torch in /usr/local/lib/python3.9/dist-packages (from easyocr) (2.0.0+cu118)</div> <div>Requirement already satisfied: Shapely in /usr/local/lib/python3.9/dist-packages (from easyocr) (2.0.1)</div> </div> </div>

손글씨data	Tesseract OCR, EasyOCR 성능 비교
인식률 :	<p>부산거 축은행그룹이 금융감독기관의 수장인 금감원장에게 로비를 벌</p> <p>Tesseract</p> <p>부산거 축은행그룹이 금융#감독기관의 42 증강원장에게 교비를 벌</p> <p>EasyOCR</p> <p> 100.1% Complete['부 산거 축은 행그 품이', '금융 7 독기관'의 수장인 금 감 원 장에게 로비를 벌']</p>

4. 학습 데이터 전처리

모델 학습을 할 때, 크게 두가지 기준을 두고 각각 데이터 학습을 실행하였습니다.

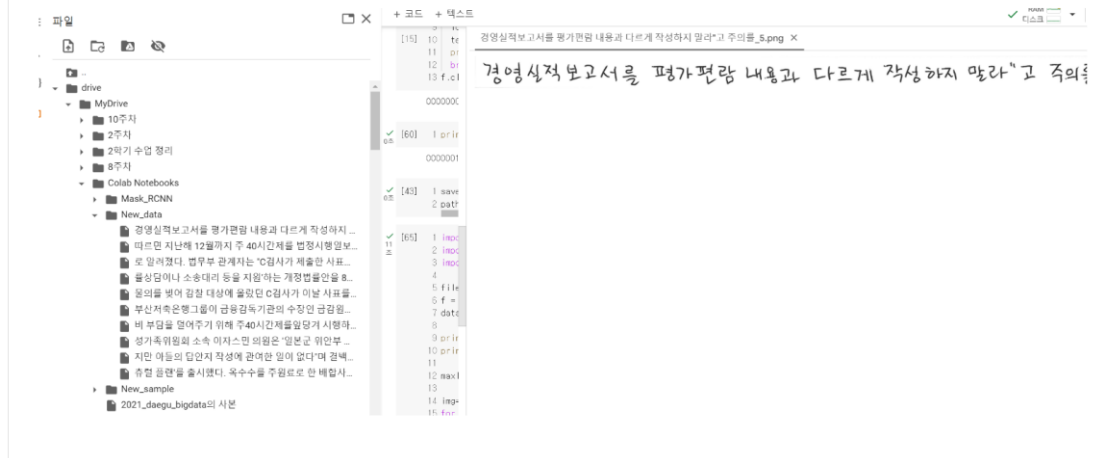
1. EasyOCR – Korean_g2(선행모델)에 AI_HUB 손글씨 데이터를 추가 학습한 케이스
2. 비교를 위해 기존 모델 없이 처음 학습을 하는 모델 : 신경망 모델 학습 단계로 CLOVA AI에서 제공하는 deep-text-recognition-benchmark라는 오픈소스 프로젝트를 이용하였습니다.

두가지 모델 모두 deep-text-recognition-benchmark의 변형 형태의 모델이기 때문에 앞서 데이터의 전처리가 필요했습니다. 먼저 이미지에는 번호(0000001.png)로 인덱싱이 되어있고, 해당 내용, type(손글씨/인쇄체) 등의 부수적인 정보는 json파일로 따로 저장되어 있는 상황이었습니다.

- a. Json – Image 인덱스 매칭 후 [문장]_[idx].jpg 로 rename

https://colab.research.google.com/drive/1_hn96G8LjKYvBWQ1oNr5PQJJKhtY5LFi#scrollTo=XIjJ6a3haCm4

▼ 코드완성본 + 결과물



- Train-validation-test 로 파일 0.7, 0.15, 0.15비율에 맞춰 분리
- TextRecognitionDataGenerator (<https://github.com/DaveLogs/TRDG2DTRB.git>)를 활용
해 학습데이터 변환
- deep-text-recognition-benchmark/create_lmdb_dataset.py 로 IMDB파일 작성해주기

5. 모델 학습, 수행테스트 및 트러블슈팅

앞서 말한 두번째 학습 방법에서는 kocrnn: CRNN 기반의 한글 텍스트 인식 모델 학습 (<https://github.com/apphia39/pghj-kocrnn/tree/main>)를 참고하되, 몇가지 트러블 슈팅 작업이 필요했습니다.

- 해당 소스파일의 dataset 을 작업하는 과정(json-image 매칭) 이 40 만개 정도의 데이터에서 최소 8 시간이 걸렸기 때문에 이미지를 binarySort 로 정렬하도록 코드를 변경해 시간을 단축시켰습니다.
- 해당 파일로 dataset 을 만들 때 `num total samples of /: 1 x 1.0 (total_data_usage_ratio) = 1` 이라는 문제가 발생하였다. 해당 문제 때문에 모델이 학습시 하나의 데이터만 학습해 학습률이 0%대로, 모델의 구조를 변경하거나 추가로 데이터를 더하더라도 학습이 전혀 이루어지지 않았습니다.

해당 소스코드를 확인해보니 이미지이름 + 문장 내용 \n 으로 구성되어야 하는 코드가 \t 로 인해 전부 하나의 문장으로 작성되었고, split 으로 맨 앞의 문장만 사용해 학습률이 0 이었음을 발견할 수 있어 해당 부분의 코드를 정상적으로 작동할 수 있도록 수정하였습니다.

수정 후 정상적으로 작성된 `gt_test(train, validation).txt` 파일 :

EasyOCR 추가학습 모델

```
[298000/300000] Train loss: 0.83615, Valid loss: 1.32023, Elapsed_time: 43540.98362
Current_accuracy : 65.017, Current_norm_ED : 0.53
Best_accuracy : 65.017, Best_norm_ED : 0.53
-----
Ground Truth      | Prediction      | Confidence Score & T/F
-----
우 려             | 우 려             | 0.7746 True
환 경             | 환 경             | 0.9584 True
상 께 하 다       | 상 하 다         | 0.4605 False
대 학 생         | 대 학 생         | 0.9875 True
-----
[300000/300000] Train loss: 0.87090, Valid loss: 1.36092, Elapsed_time: 43835.32289
Current_accuracy : 63.370, Current_norm_ED : 0.52
Best_accuracy : 65.017, Best_norm_ED : 0.53
-----
Ground Truth      | Prediction      | Confidence Score & T/F
-----
|                 |                 | 0.6799 True
|                 |                 | 0.9977 True
를 러 나 다       | 러 나 다         | 0.8655 False
별 도             | 별 도             | 0.6989 True
-----
end the training
(new_env) caps1@workstation:~/test/pghj-kocrnn$ ls
```

3) Deep-text-recognition-benchmark의 소스코드 train.py, dataset.py(데이터-> 데이터 batch set으로 변경하는 코드) 에서 사용하는 파이썬 버전이 맞지 않아 코드가 정상적으로 실행 되지 않았고, 우리가 사용하는 파이썬3.9에 맞춰 다음과 같이 변경해주었습니다.

- Dataset- 87번 줄 : (image, text = next(data_loader_iter) : dataloader 내부에서 next를 호출해 해당 모듈에는 해당 함수가 없다는 에러가 발생했으나, 이를 외부에서 next를 호출하는 방식으로 변경함
- Train 256번 줄 : parser.add_argument('--data_filtering_off', default=True, help='for data_filtering_off mode') data tensor size를 character수에 맞춰 조정해 주었음에도 tensor size mismatch에러가 지속적으로 발생해, 해당 부분의 default를 조정한 후 사용해줘야 문제 없이 학습할 수 있었음.
- Create_lmdb.py 47번 줄 : #imagePath, label = datalist[i].strip('\n').split('\t')

```
imagePath = datalist[i].strip('\n').split('\t')[0]
```

```
label = datalist[i].strip('\n').split('\t')[1]
```

두 변수에 각각 값을 할당해주지 못해 해당 코드를 수정 후 정상적으로 작동하도록 함.

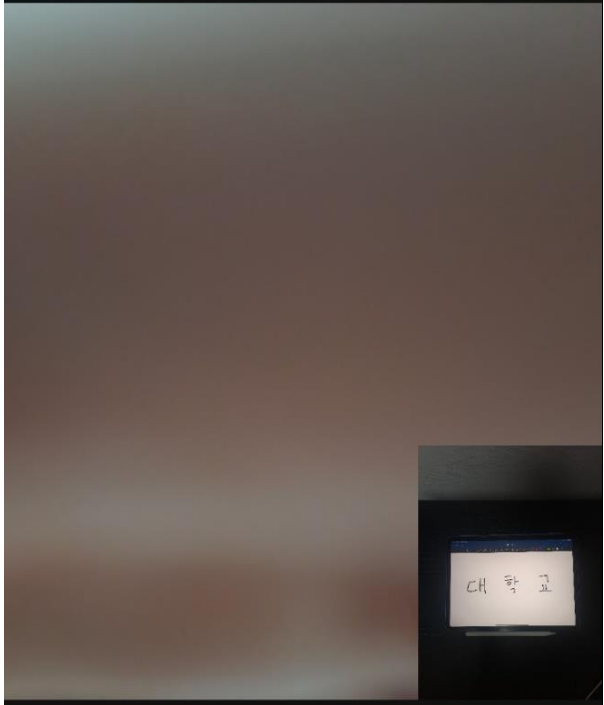
```
no changes added to commit (use "git add" and/or "git commit -a")
(base) caps1@workstation:~/test/easyocr/capstone_ocr$ git add .
(base) caps1@workstation:~/test/easyocr/capstone_ocr$ cd saved_models/
(base) caps1@workstation:~/test/easyocr/capstone_ocr/saved_models$ ls
학습완료된 모델이 저장되는 곳입니다 None-ResNet-None-CTC-Seed1111 TPS-ResNet-BiLSTM-CTC-Seed1111
None-ResNet-None-Attn-Seed1111 None-VGG-BiLSTM-CTC-Seed1111
(base) caps1@workstation:~/test/easyocr/capstone_ocr/saved_models$
```

이후 두가지 모델에 대해 각각 22.4%, 65.017% 의 성능을 확인하였으며, 최초학습 모델에 대해서는 Transformer, Prediction, Attention등의 모델 구조와 파라미터를 바꾸며 성능을 개선할 수 있는 방향으로 조정해보았습니다. EasyOCR은 None-VGG-BiLSTM-CTC모델

을 사용하고 있어 추가학습 모델은 그에 맞춰 변형해주고, 기본 모델에 맞춰 설정된 input size와 output사이즈를 EasyOCR의 config에 맞춰 변형했습니다. GPU서버의 환경에 맞춰 workers, batchsize 등을 조정해주고, learning rate는 여러 번 학습을 돌린 후 0.1일 때 성능이 좋아 0.1로 학습을 진행했습니다. charaters는 deep-text-recognition-benchmark의 설정과 달라 실제로 실행시 size mismatch for module.Prediction.weight: copying a param with shape torch.Size([1009, 256]) from checkpoint, the shape in current model is torch.Size([1008, 256]). 에러가 계속 발생하였고, 파라미터를 주었음에도 FT option때문에 정해진 tensor크기로 사용하게 되어 이 부분은 EasyOCR config에 설정된 글자 1013자에 맞춰 train.py의 한글 수를 더 늘려주는 방법을 택했습니다.

6. 성과

Model	Best accuracy
TPS-ResNet-BiLSTM-CTC	22.45
None-VGG-BiLSTM-CTC with 40만개 data	65.02
None-VGG-BiLSTM-CTC with 70만개 data	74.12

<div>ocr_application</div>  <div>텍스트 인식</div> <p>빠 학 고</p>	<div>← 맞춤법 검사 결과</div> <p>맞춤법 정답률은 100.00%입니다.</p> <div>정답 문장</div> <p>대학교</p> <div>인식된 문장</div> <p>대학교</p> <div>사진 확인</div>
초기 모델	EasyOCR추가 학습된 모델

7. 모델 실행 파일 작성

```

from easyocr.easyocr import *

# GPU 설정
os.environ['CUDA_VISIBLE_DEVICES'] = '0,1'

def get_files(path):
    file_list = []

    files = [f for f in os.listdir(path) if not f.startswith('.')] # skip hidden file
    files.sort()
    abspath = os.path.abspath(path)
    for file in files:
        file_path = os.path.join(abspath, file)
        file_list.append(file_path)

    return file_list, len(file_list)

if __name__ == '__main__':

    # Using default model
    # reader = Reader(['ko'], gpu=True)

    # Using custom model
    reader = Reader(['ko'], gpu=True,
                    model_storage_directory='./workspace/user_network_dir',
                    user_network_directory='./workspace/user_network_dir',
                    recog_network='custom')

    files, count = get_files('./workspace/demo_images')

    for idx, file in enumerate(files):
        filename = os.path.basename(file)

        result = reader.readtext(file)

        # ./easyocr/utils.py 733 lines
        # result[0]: bbox
        # result[1]: string
        # result[2]: confidence
        for (bbox, string, confidence) in result:
            print("filename: '%s', confidence: %.4f, string: '%s'" % (filename, confidence, string))
            # print('bbox: ', bbox)

```

모델학습서버에서 학습된 모델(정확도가 가장 높은 모델을 korcnn.pth로 rename 한 뒤, ML서버로 모델을 이동시켰음)을 바로 사용할 수 있도록 작성된 run.py이다. 파일 경로를 입력한 뒤,

(venv) \$ python3 run.py 실행을 통해 이미지를 인식한 결과를 출력하도록 작성하였습니다.

'나가다' 손글씨 파일을 run.py 파일로 모델을 돌려본 결과 '나기다' 라고 인식
<pre> (new_env) caps1@workstation:~/test/pghj-kocrnn/EasyOCR\$ python run.py filename: 'test.png', confidence: 0.0825, string: '나 기 다' (new_env) caps1@workstation:~/test/pghj-kocrnn/EasyOCR\$ [4] 0:bash* </pre>

8. 회의록 기록

종프 회의록				
<div> <div>새로 만들기</div> <div> <div>✂</div> <div>📄</div> <div>📁</div> <div>🗨</div> <div>📧</div> <div>🗑</div> </div> <div> <div>↕ 정렬</div> <div>☰ 보기</div> <div>⋮</div> </div> </div>				
→ > 종프 회의록				
<div> <div>🏠 홈</div> <div> <div>📁 OneDrive - Personal</div> <div>📄 문서</div> <div>🖼 바탕 화면</div> <div>📷 사진</div> </div> <div> <div>🖼 바탕 화면</div> <div>⬇ 다운로드</div> <div>📄 문서</div> <div>📷 사진</div> </div> </div>	<div> <div>☐ 이름</div> <div>⤴</div> </div> <div> <div>📄 0405 종프 회의 3주차</div> <div>📄 230426_종프회의록</div> <div>📄 230503 종프 회의 4주차</div> <div>☐ 📄 230607회의</div> <div>📄 230510 회의록</div> </div>	<div>수정된 날짜</div> <div>2023-06-21 오후 1:17</div> <div>2023-06-21 오후 1:17</div> <div>2023-06-21 오후 1:17</div> <div>2023-06-07 오후 6:39</div> <div>2023-06-20 오후 3:15</div>	<div>유형</div> <div>Microsoft Word 문서</div> <div>Microsoft Word 문서</div> <div>Microsoft Word 문서</div> <div>Microsoft Word 문서</div> <div>Microsoft Word 문서</div>	<div>크기</div> <div>18KB</div> <div>18KB</div> <div>15KB</div> <div>16KB</div> <div>18KB</div>

매 주 수요일마다 대표 대학원생분들과 진행하는 미팅 회의록을 기록 후 공유하였습니다. 이 후 따로 일정이 정해진 작업은 개인 캘린더에 기록 후 단독방에 리마인드 하였습니다.

참고자료 :

<https://github.com/oyoungsun/capstone1/tree/main>

capstone1

종합설계프로젝트1 - EasyOCR 로 한국어 손글씨 인식 향상시키기

Colab 폴더 :

GPU_EASYOCR_ENV

cuda, 파이썬 환경 설정

Tesseract_OCR_SampleData.ipynb

테서렉트로 한국어 학습 후 한국어데이터(AI Hub 이용) 인식을 확인해봄.

compare_Tesseract_vs_EasyOCR

테서렉트와 easyocr의 성능 비교

EASY_OCR_train.ipynb

easyOcr 한국어모델 koreaG2모델을 이용해 한국어 단어(위키디피아 모듈을 통해 생성)학습시킨 후, 성능 확인

Aihub_data_prepare.ipynb

모델에 학습시키기 위해 데이터를 학습용 데이터에 맞게 파일명 변경하는 코드

(Latest)Model_training.ipynb