NAVER 부스트캠프 AI Tech 3기
Level 3 최종 프로젝트

Wrap-UP Report

CV-16 Medic 조

권순호_T3012
서다빈_T3248
서예현_T3105
이상윤_T3148
전경민_T3187

프로젝트 개요

프로젝트 소개

- 사용자의 알약 이미지로부터 알약을 식별하는 인공지능 서비스이다.

프로젝트 목적

- 사용자의 알약 이미지로부터 성상, 제형, 색상을 식별 후 조건에 맞는 알약을 검색하여 알약의 종류를 식별한다.

타겟층

- 지리적, 물리적 한계로 약국이나 병원을 방문하기 어려운 사람
- 알약은 있지만 알약을 구분할 수 없는 사람

프로젝트 선정 배경

- 종종 일어나는 처방 실수, 및 착각으로 인한 약물사고를 예방하고자 하였다.
- 실제 보건 계열 종사자에 따르면 노년 층의 경우 어떤 알약인지 병원에 방문하여 알약을 찾는 경우가 존재한다고 하며, 한국의 통계를 보았을 때도 약물 오복용에 의한 사고는 줄지 않고 계속 유지되고 있는 추세이다.

프로젝트 팀 구성 및 역할

권순호_T3012	FastAPI, streamlit, GCP, OCR, Text Recognition
서다빈_T3248	FastAPI, streamlit, OCR, Text Recognition
서예현_T3105	Data EDA, Data Pre-processing, Image Classification, Custom Dataset Production
이상윤_T3148	Metric learning, Instance Segmentation, Docker
전경민_T3187	Data EDA, Data Pre-processing, Data Annotation, OCR, Text Recognition

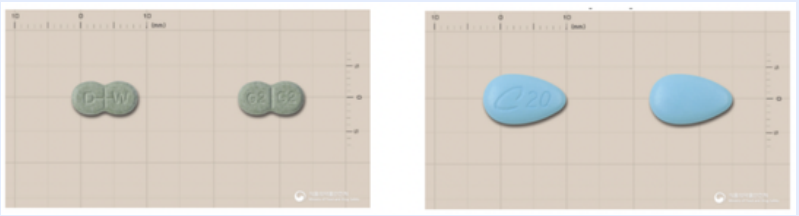
프로젝트 수행 결과

프로젝트 수행 계획 및 방법

- Model 팀과 BackEnd 팀으로 나누어 진행하는 것으로 계획하여 진행하였다.
- Model 팀에서는 Data EDA부터 Pre-processing과 같은 전처리, Model 실험 등을 진행하고, BackEnd 팀에서는 FastAPI를 이용해 BackEnd 구축부터 Streamlit을 이용한 Front-Back, BentoML과 MySQL, GCP, docker를 이용하였다.
- 이후, BackEnd 팀의 업무가 조기 종료됨에 따라 Model로 넘어와서 Segmentation, OCR 등의 업무를 진행하였다.
- Notion을 활용하여 팀원들 간의 실험 결과를 공유하였다. (Link)
- Github issue를 활용하여 진행 상황을 공유하였다. (Link)

사용 데이터

1. 의약품 안전나라 데이터(Link)
2. epillid_benchmark (Link)
3. 기타 이미지 데이터(Link)



자료 1. 의약품 안전나라 데이터

실험 및 개선

- 데이터 EDA 및 전수 조사

데이터 내 색상이 "갈색-연한"인 알약들

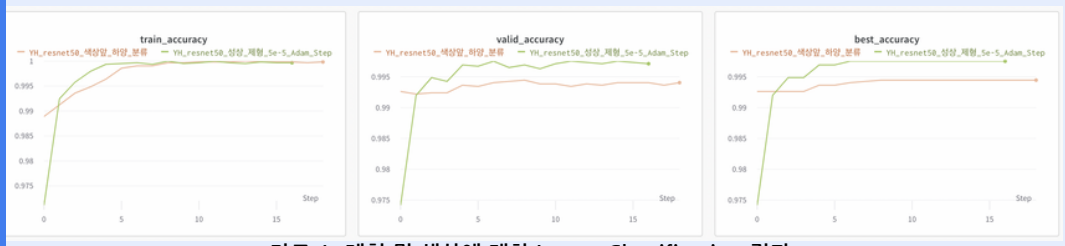
데이터 내 이미지(좌측), 실제 알약 이미지(우측)

자료 2. 데이터 EDA 중 발견한 Noise data

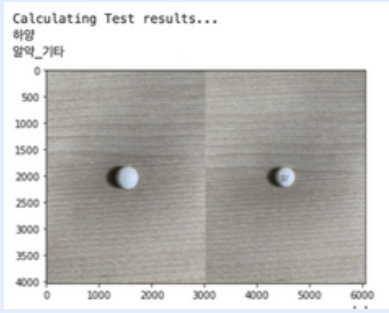
  - Noise data를 찾아내고, 학습에 사용하고자 하는 column들을 찾아내었다.
  - 추가적으로 최종적으로 진행할 test를 위한 test data production을 진행하였다.

- Image Classification
  - 진행 목적
    - image classification을 통해 최종 similarity를 계산하는 단계에서 비교를 해야 하는 후보군의 수를 줄인다.

- Color Classification + Type & Shape Classification
  - 주어진 알약에 대해 색깔(White인지 아닌지)을 분류하는 모델과, 알약의 종류(Tablet인지 Capsule인지)와 알약의 모양을 분류하는 모델을 개발하였다. 정확도는 99.5% 이상을 기록하였다.

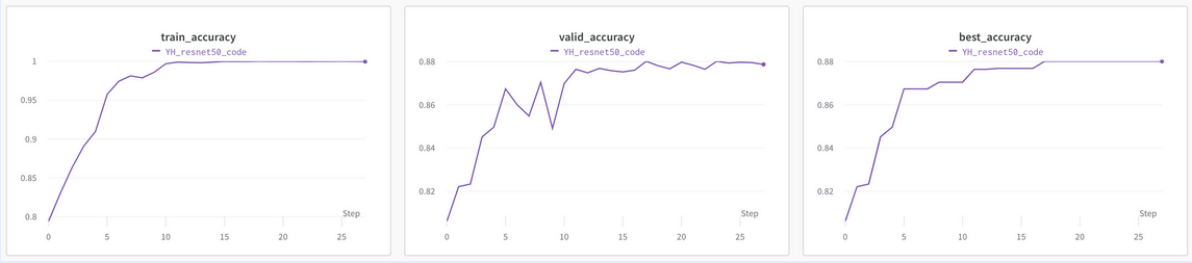


자료 1. 제형 및 색상에 대한 Image Classification 결과



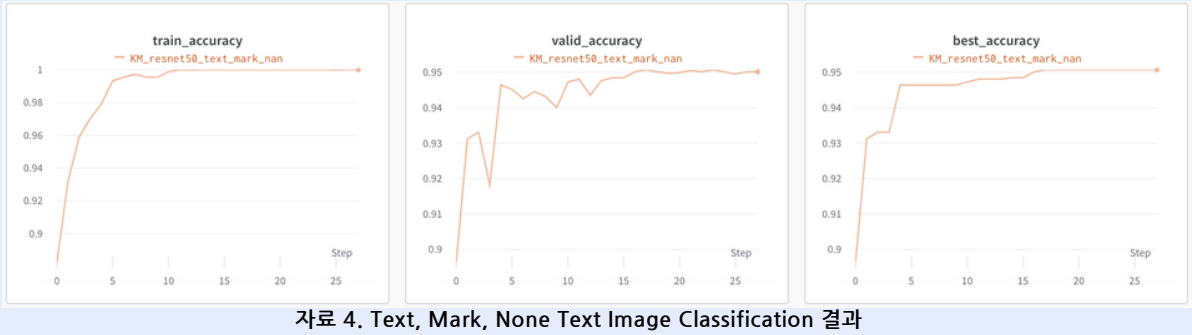
자료 2. 알약 제형과 색상 분류 test 결과

- Code Classification
  - Code(제품코드형) Classification의 경우 accuracy가 88%를 기록하면서, 오류가 나와서는 안 되는 의료 task에 맞지 않아 폐기하였다.



자료 3. Code에 대한 Image Classification 결과

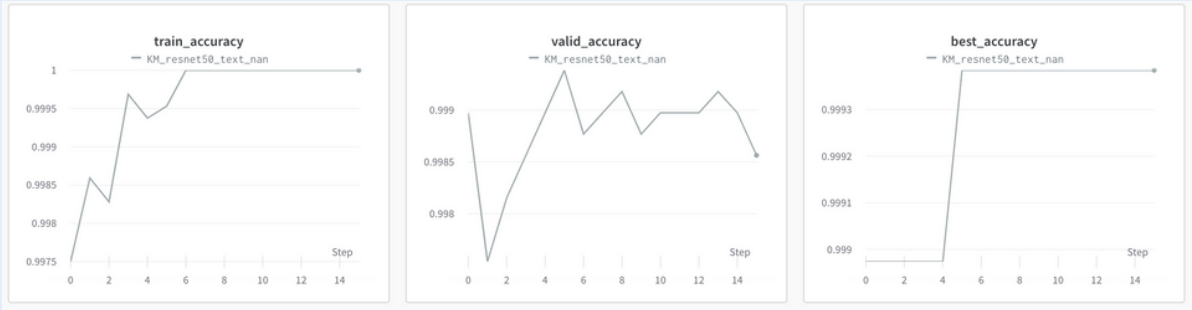
- Text type Classification
  - Text vs Mark vs None Text



자료 4. Text, Mark, None Text Image Classification 결과

- 전반적으로 accuracy가 양호하게 나타난 것으로 보였지만 mark에 대한 인식률이 현저하게 낮았고 실제 test용 데이터로 분류를 해본 결과 제대로 인식하지 못하여 classification 범위를 text와 none text로 제한하기로 결정하였다.

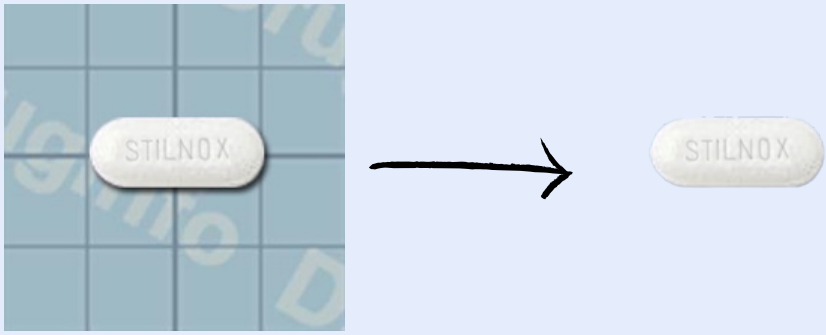
  - Text vs None Text



자료 5. Text, None Text Image Classification 결과

        - 알약에 Text가 존재하는지와 존재하지 않는지를 Image Classification을 통해서 비교하였고 정확도는 높게 측정되었으나 실제 text를 인식하는 것이 아니라면 크게 도움이 되지 않을 것 같아 폐기하였다.
    - 진행 결과
        - 프로젝트 전반의 inference time을 15s → 7s로 획기적으로 줄이는데 성공하였다.

- Instance Segmentation
    - 식약처 데이터처럼 정제된 사진이 아닌 일반 사용자의 wild한 데이터에서도 약을 잘 찾을 수 있도록 하기 위해 배경으로 인한 노이즈를 제거하기 위해 시도하였다.
    - 본래 instance segmentation을 수행하려 하였으나 결과가 만족스럽지 못해 폐기하고 YOLOv5로 object detection을 수행한 후 remove.bg 라이브러리로 주위의 배경을 제거하여 segmentation을 수행하였다. yolov5s가 0.994의 mAP를 기록하여 그대로 사용하였다.



자료 1. Segmentation이 진행되는 모습을 간단하게 나타낸 그림

- Metric Learning
    - 본래 ImageNet pretrained 된 모델은 알약을 구분할 수 있는 feature extraction에 적합하지 않아서 같은 알약은 cosine distance가 가깝게 다른 알약은 멀게 하도록 학습시켰다.
    - Triplet Loss, Contrastive Loss, Cross Entropy Loss를 사용하였다. 함께 사용하여 multi-head metric learning을 수행
    - 미국 NIH의 알약 데이터와 마이크로소프트가 사용자들로부터 수집한 데이터인 ePillID dataset을 활용하여 학습하였다.

- OCR
    - 진행 목적
        - 데이터 셋에 근거하여 색상 및 제형만으로는 알약이 특정되기 어려울 것이라고 판단하였다.
        - 실제로 '색상 앞'이 하양, '의약품 제형'이 원형인 데이터는 24,000여 개 중 5,186개였다.
        - 앞, 뒷면의 text를 모두 이용할 경우 겹치는 데이터가 거의 존재하지 않았다.

- 진행 과정
  - i. OCR Model의 후보군을 탐색하고 범위를 좁혔다. (EasyOCR, KorOCR, deep-text-recognition-benchmark, CRAFT-pytorch)
  - ii. 각 성능 비교 및 탐지 가능 언어 영역을 고려하여 모델을 선정하였다. (CRAFT-pytorch로 text 영역 추출 후, deep-text-recognition-benchmark로 해당 영역의 text 예측)
  - iii. deep-text-recognition-benchmark의 경우 성능이 후보군 중 매우 뛰어나지만 한글을 검출할 수 없어서 한글과 영어를 모두 검출할 수 있게 새롭게 label한 데이터로 재학습하였다.

```
[300000/300000] Train loss: 0.00000, Valid loss: 7.05664, Elapsed_time: 40335.21605
Current_accuracy : 11.816, Current_norm_ED  : 0.36
Best_accuracy    : 12.254, Best_norm_ED     : 0.36
--------------------------------------------------------------------------------
Ground Truth                | Prediction                  | Confidence Score & T/F
--------------------------------------------------------------------------------
kpp                         | kap                         | 0.4969  False
dp300                       | e0                          | 0.7098  False
엔프렌드                    | 센논                        | 0.2374  False
loz                         | lce                         | 0.2646  False
kln                         | kdn                         | 0.8878  False
--------------------------------------------------------------------------------
```

자료 1. 새롭게 label한 데이터로 학습하는 모습

  - iv. 새롭게 label한 데이터로 테스트한 deep-text-recognition-benchmark 모델의 결과를 confidence score 별 threshold를 기준으로 정확도를 비교하였다. (total_cnt: 전체 데이터 개수, hangul_cnt: 전체 데이터 중 한글 label 개수, correct_cnt: ground truth label == predicted 일치하는 개수, wrong_cnt: gt와 다르게 예측한 개수)

confidence score threshold	total_cnt	hangul_cnt	correct_cnt	wrong_cnt	correct / total	correct / (total - hangul)
0	2282	354	1127	801	0.4939	0.5845
0.5	1186	10	958	218	0.8078	0.8146
0.6	1057	7	900	150	0.8515	0.8571
0.7	925	5	816	104	0.8822	0.8870
0.75	865	4	781	80	0.9029	0.9072
0.8	805	2	740	63	0.9193	0.9215

자료 2. confidence score 별 threshold를 기준으로 정확도를 비교한 자료

  - v. 모델의 정확도를 고려하여 Jaccard Similarity와 Precision Metric 간 비교하여 이를 기반으로 예측 알약 후보군을 전달하였다. (다만, confidence score threshold가 높아질수록 사용 가능한 text의 개수가 줄어들게 되어 해당 부분에서 중지하고 최종 적용하지 않았다.)

correct = jaccord_similarity ≥ 0.5

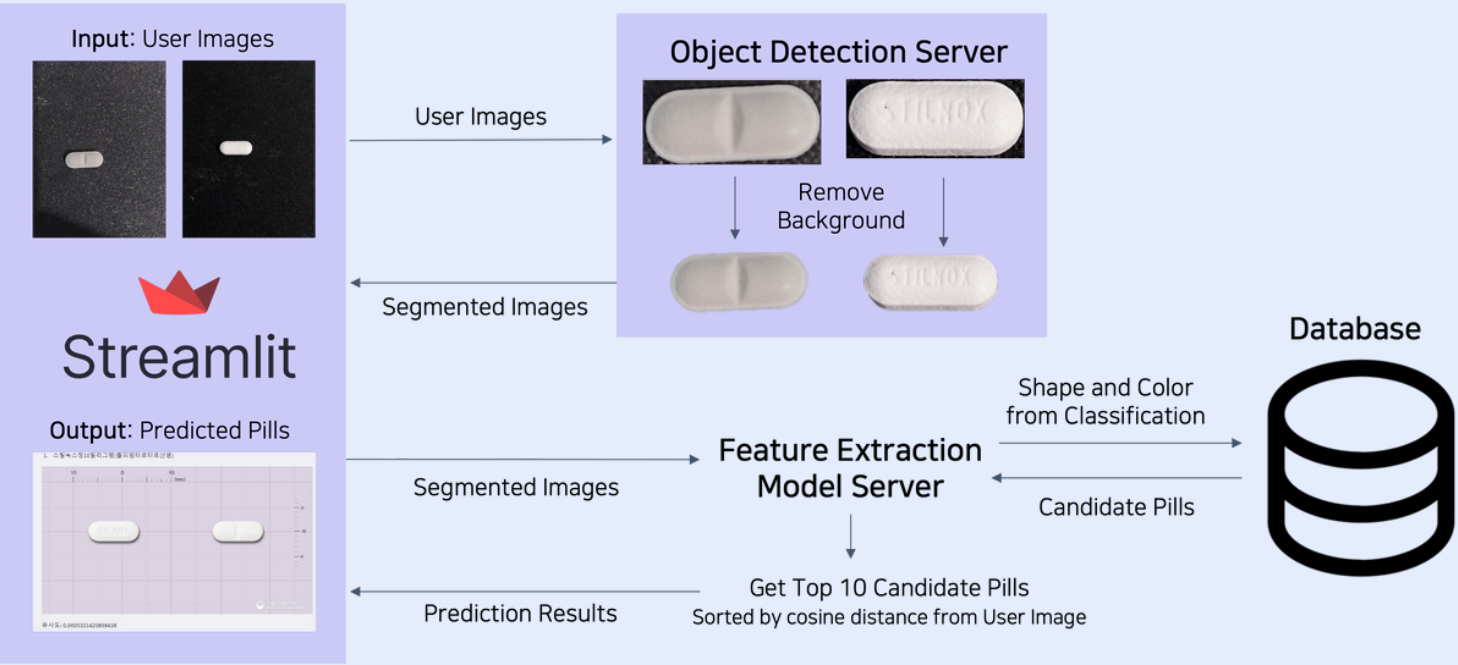
confidence score threshold	total_cnt	hangul_cnt	correct_cnt	wrong_cnt	correct / total	correct / (total - hangul)
0	2282	354	1452	476	0.6363	0.7531
0.5	1186	10	1086	90	0.9157	0.9235
0.6	1057	7	994	56	0.9404	0.9467
0.7	925	5	888	32	0.96	0.9652
0.75	865	4	834	27	0.9642	0.9686
0.8	805	2	787	16	0.9776	0.9801

correct = precision ≥ 0.5

confidence score threshold	total_cnt	hangul_cnt	correct_cnt	wrong_cnt	correct / total	correct / (total - hangul)
0	2282	354	1596	332	0.6994	0.8278
0.5	1186	10	1114	62	0.9393	0.9473
0.6	1057	7	1010	40	0.9555	0.9619
0.7	925	5	898	22	0.9708	0.9761
0.75	865	4	843	18	0.9746	0.9791
0.8	805	2	792	11	0.9839	0.9863

자료 3. Jaccard Similarity와 Precision Metric 각각에 대한 정확도를 비교한 자료

프로젝트 구조



- Docker를 이용해 streamlit을 이용한 front-end와 FastAPI를 이용한 back-end 서버를 구성해서 배포하였다.
- 이후 서버에서 docker-compose 커맨드를 사용하여 서비스하였다.
- GCP 클라우드 서버의 성능이 aistages 서버의 성능보다 좋아서 현재 inference time은 약 7초 → 5초 정도로 줄었다.

결과

- 총 30개의 직접 수집한 약 데이터로 테스트
- top-1 Accuracy 43%
- top-5 Accuracy 80%

자체 평가 의견

잘한 점들

- End-to-end로 모든 task를 진행해 보면서 지금까지는 주어진 task를 받아서 하는 것에서 벗어나 직접 문제를 정의하고 이를 어떻게 해결할지 모든 팀원들이 같이 토론해 보았다.
- 모두가 일정에 맞추어 자신에게 주어진 일을 완료하였고, 모든 것을 다 같이 공유하고자 하는 마인드를 통해 다 함께 프로젝트를 진행한다는 느낌을 강하게 받았다.
- 데이터베이스 전체를 탐색함으로 인해 15초나 걸렸던 inference time을 image classification과 GCP 사용으로 5초까지 줄이는데 성공하였다.

시도했으나 잘 되지 않았던 것들

- 알약의 검색 범위를 좁히기 위해 OCR 및 pill code classification을 시도하였지만 성능이 잘 나오지 않아 최종 적용하지 못했다.
- NNI를 통한 hyper-parameter tuning을 진행하였으나, 잘 작동하지 않아 shell script로 대체하였다.

아쉬웠던 점들

- 전체적인 설계를 진행하고 실험을 한 것이 아닌 설계, 실험, 설계, 실험의 과정을 반복하였다.
- 배경에 따라 색이 바뀌는 캡슐들에 대한 accuracy가 낮게 나왔다.
- False prediction이 없어야 하는 의료 분야에서 100%의 accuracy를 기록하지 못하였다.
- XGBoost와 같은 기능을 사용해 보지 못하였다.
- torch.hub 모델 이슈 때문에 BentoML을 사용하지 못하였다. ONNX나 TorchScript 공부를 했더라면 사용할 수 있었을 텐데 아쉽다.
- GitHub Action을 통한 자동 배포를 실습해 보지 못한 것이 아쉽다. Docker compose를 사용하기로 하여서 복잡함이 더해졌던 것 같다.

프로젝트를 통해 배운 점 또는 시사점

- A부터 Z까지 직접 문제를 파악 및 정의하고 데이터 수집 및 EDA + 모델 선정 및 성능 향상을 시도하는 이 모든 일련의 과정을 직접 경험해 보았다는 것이 의미 있었다.
- Github, Notion 등을 활용한 협업을 통해 프로젝트 내에서 서로 다른 분야를 맡고 있더라도 진행 상황을 바로바로 확인할 수 있었다.
- Docker image를 만들어보니, 모델 서빙의 중요성과 경량화의 중요성을 느낄 수 있었다. 도커 이미지가 6GB 가까이 되는걸 보고 왜 그렇게 많은 개발자들이 모델 서빙에 대해 고민하는지 알 수 있었다.

개인 회고 - 권순호_T3012

목표

- streamlit, FastAPI 사용한 Front-End, Back-End 구성한다.
- GCP를 적용하여 DB내에 알약 이미지 및 data 저장한다.
- MySQL을 적용하여 DB내의 알약 검색한다.
- text recognition + OCR을 적용한다.
- 기존의 OCR을 custom 하여 우리만의 데이터셋으로 학습을 시도한다.
- OCR 모델로 영어 뿐만이 아닌 한글도 식별할 수 있도록 변형한다.

내가 한 것

- 강의 내용을 토대로 Streamlit과 FastAPI를 적용하여 기본적인 구성을 완성하였다.
- 이후 약간의 Front-End를 구성하고 User-friendly하게 UI를 구성하였다.
- Google Cloud Platform 내의 버킷을 반들고 GCP 내의 알약 데이터들을 저장
- MySQL을 사용하여 DB내의 알약을 검색하였다.
- 이후 모델 쪽에 합류하여, 알약의 검색 범위를 조금 더 좁히기 위해 다양한 실험을 시도하였다.
- 그 중에서도 text recognition 모델을 적용한 전처리를 통해, OCR 모델의 성능 향상을 시도하였다.
- 기존의 영어만 구분할 수 있었던 OCR에 한글까지 식별할 수 있도록 추가적으로 custom을 진행하였다.
- 가지고 있는 알약의 text image로 학습을 시도하였다.

한계점

- OCR을 적용하려고 기존의 모델을 custom 하여 시도하였지만, 모델의 성능이 좋지 않아 결국 적용하지 못하였다.
- 시간상, 추가적으로 bentoML 등 product serving 관점에서 다양한 것들을 시도하지 못했다.

다음에 시도해볼 것

- minio, bentoML 적용
- Kubernetes 배포
- 성능이 좋지 않았던 OCR 모델의 성능을 개선하고 다시 프로젝트에 적용
- GCP가 아닌 AWS S3로 시도

개인 회고 - 서다빈_T3248

목표

- 아이디어 구성 단계부터 최종 서비스 서빙 단계까지 전반적인 workflow 이해하기
- 부스트캠프 강의를 통해 새롭게 배우게 된 product serving 지식을 바탕으로 서비스 만들기 (e.g. Streamlit, FastAPI)
- 알약 위에 작성되어 있는 텍스트 인식하는 기능 구현

내가 한 것

- Streamlit으로 서비스의 기본적인 front-end 기능 구현 후 FastAPI 적용하였다.
- 알약 이미지 데이터의 텍스트 정보를 활용하기 위해 CRAFT와 text recognition benchmark를 사용하였다.
- CRAFT 모델은 결과로 input 이미지 위에 텍스트 영역을 박스로 표현하고 해당 박스의 좌표를 txt 파일에 저장한다. 이미지에 있는 텍스트를 읽는 text recognition 모델을 사용하기 위해 CRAFT 코드를 수정해서 텍스트에 해당되는 영역들만 crop 한 이미지들을 저장하도록 하였다.
- 팀 내에서 직접 label 한 알약 이미지의 텍스트 데이터를 test 하여 알약 데이터에 대한 모델 평가를 진행하였다. 예측된 텍스트와 ground truth를 비교하여 모델의 결과가 사용 가능한지 평가하였다.

한계점

- 과감하게 신선한 아이디어를 내고 그것들을 바탕으로 다양한 실험을 해봤으면 좋았을 것 같다.
- 처음 시작할 때 최종 프로젝트의 백앤드 서빙을 담당했기에 product serving 강의를 이해하는 것을 우선순위로 두었었는데, 프로젝트를 우선시했으면 기여를 더 많이 했을 수 있었을 것이다.

다음에 시도해볼 것

- 한글 데이터를 더 확보해서 text recognition 모델 재학습 후 서비스에 적용시키고 싶다.
- 최종 product를 정해진 기간에 맞춰 완성할 수 있도록 계획하면서 프로젝트 관리를 제대로 하고 싶다.
- 투자한 시간에 비해 결과물이 많이 부족하다고 느끼게 되면서 스트레스를 받았다. 다음에는 멘탈을 챙기면서 힘들어도 꾸준히 노력할 것이다.

개인 회고 - 서예현_T3105

목표

- Data collection 및 EDA를 통해 수집한 데이터를 분석하고, 데이터에서 classification task에 사용할 수 있는 column들을 특정한다.
- Raw data를 학습에 사용할 수 있도록 preprocess 한다.
- EDA 분석을 통해 학습하고자 했던 column에 대해 preprocess된 데이터를 학습시킨다.
- 학습 결과를 분석하고 더 나은 모델을 만들기 위해 label 단순화, hyperparameter tuning, column 재분석 등을 진행한다.

내가 한 것

- 1주차

Data collection 및 EDA를 통해 수집한 데이터를 분석하고, 특히 추후 학습에 방해가 될 수 있는 noise data에 대한 분석을 위주로 진행하였다.

- 2주차

1주차에서 진행한 EDA를 바탕으로 noise data 제거 및 data preprocessing 작업을 진행하였고, EDA를 통해 학습에 사용하고자 하는 column들에 대한 기초적인 탐색이 시작되었다. 추가적으로 최종적으로 진행할 test를 위한 test data production을 진행하였다.

- 3주차

timm을 사용하여 pretrained된 모델을 불러왔으며, product serving의 특성상 모델의 inference time이 굉장히 빨라야하기 때문에, 기존에 SOTA라고 불리는 모델보다는 굉장히 간단하게 구성된 resnet18, resnet50과 같은 모델로 학습을 진행하였다.
classification task 별로 가장 성능이 좋은 모델을 찾은 후에는, hyperparameter tuning을 통해 해당 모델의 성능을 가장 많이 끌어올릴 수 있는 hyperparameter를 찾았다.
최종적으로 프로젝트의 inference time을 15s → 7s로 50% 이상 감소시켰다.

- 4주차

seed 고정 및 readme 작성을 통해 완벽히 reproduction이 가능하도록 만들었고, 성능이 잘 나오지 않았던 task들에 대해서 새로운 방식으로 접근해보면서 성능을 최대 7.5%까지 향상시켰다.

한계점

- Hyper Parameter Tuning 자동화 실패
- Noise data 에 대한 적절한 대응 실패로 인해 제품코드형 classification model 폐기

다음에 시도해볼 것

- 하나의 모델에서 가장 좋은 성능을 찾아주는 XGBoost 사용
- 랩업 리포트가 아닌 일종의 'tech report' 작성

For more information:
https://yehyunsuh.notion.site/Day105-2022-06-10-c76df6543d7047cca181cc8750555644

개인 회고 - 이상윤_T3148

목표

- 사용자가 보낸 약을 segmentation해서 통해 배경으로 인한 노이즈 제거
- Metric learning을 통한 효과적인 feature 추출 학습
- Docker를 통한 환경 격리 및 이미지로 클라우드에 배포

내가 한 것

- 구글 클라우드 서버 구축, 데이터베이스 구축, 도커 이미지를 통한 클라우드 배포 docker compose를 이용해서 streamlit frontednd와 fastapi 백엔드가 서로 공유하는 port를 통해 통신할 수 있도록 하였다.
- Object detection을 통한 알약 영역 분리, 배경 제거를 통한 segmentation을 시도하였다, 초기에는 Instance segmentation을 시도할 생각이었지만. Object detection을 먼저 하고 배경제거를 하는 것이 성능이 더 좋고 더 빨랐기에 후자의 방법을 선택하였다.
- Metric learning을 통한 알약 feature 추출 학습. triplet loss, contrastive loss 등을 사용하였다.

한계점

- Docker 이미지 경량화를 시도하지 못했다. 백엔드 이미지가 6gb가 넘는데 줄이지 못했다.
- Bentoml 적용을 하지 못했다. bentoml이 공식적으로 torch.hub 모델을 지원하지 않아 적용에 어려움이 있어서 적용하지 못하였다.
- UI 개선을 하지 못했다. 프론트엔드 지식이 부족하여 깔끔한 UI 구축을 하지 못하였다.

다음에 시도해볼 것

- torch.hub 모델을 ONNX로 변환하여 BentoML 적용
- React를 이용한 편리한 사용자 UI 구축
- Redis를 통한 빠르게 알약 feature를 저장하고 불러오는 기술

개인 회고 - 전경민_T3187

목표

- End-to-End 프로젝트를 처음 해보기 때문에 전반적인 Flow를 이해하고 기존에 진행했던 대회형 프로젝트와 다른 점을 파악하여 적응한다.
- 자신있는 부분이 아니더라도 도전하여 배운다.
- AI Engineer간 협력이 아닌 AI와 BackEnd간 협력에 대해 이해한다.
- 기본적인 API적용과 모델 Serving을 시도한다.

내가 한 것

- Data EDA를 통해 Data Noise를 파악하고 학습 계획을 설정 및 공유하였다.
- 학습을 위한 Data pre-processing을 진행하고 image data를 시각적으로 분석하였다.
- Text 유무등을 파악하는 Image Classification 실험을 진행하였다.
- Segmentation을 위한 Data Annotation과 OCR에서 사용할 Custom data labeling을 진행하였다.
- 다양한 OCR Engine을 찾아보고 이를 적용하고자 실험하였다.
- OCR Engine의 정확도를 고려하여 Metric을 조사하고 가장 정확하게 전달할 수 있는 Text Similarity 방법을 선정하여 전달하였다.

한계점

- Data와 Model만 보았기 때문에 목표였던 API 적용을 시도해보지 못하였다.
- OCR 실험을 긴 시간동안 진행하였고 프로젝트 마감 전까지 시도했기 때문에 모델 Serving을 해보지 못하였다.
- OCR에 긴 시간을 투자하였음에도 불구하고 정확성 및 시간적 요소 등을 고려하여 첫 서비스 배포에 적용되지 못하였다.
- 개인적으로 긴장을 너무 과하게 하여 의사소통 측면에 조금 영향을 주지 않았나 생각한다.

다음에 시도해볼 것

- Data 및 Model뿐만 아니라 API 적용과 BackEnd 측면까지 시도해보면 좋을 것 같다.
- 성공하지 못한 실험을 조금 더 분석해보고 적용이 가능한 지에 대한 상세 분석을 통해 현재 실험을 마무리한다.
- 모델 Serving이 필요한 이유를 고민해보고 직접 느낀다음 적용해 본다.
- 자신감을 조금 높이고 긴장도를 낮추어 본다.