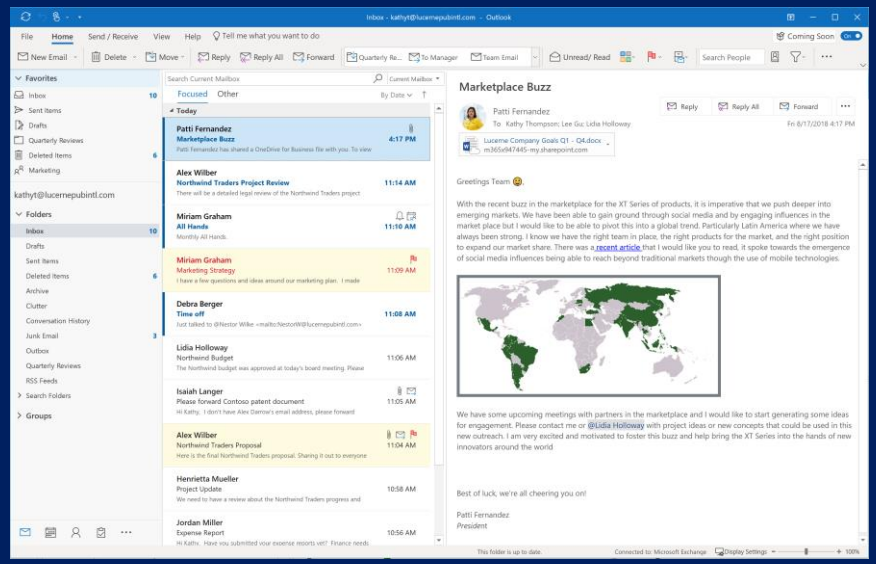
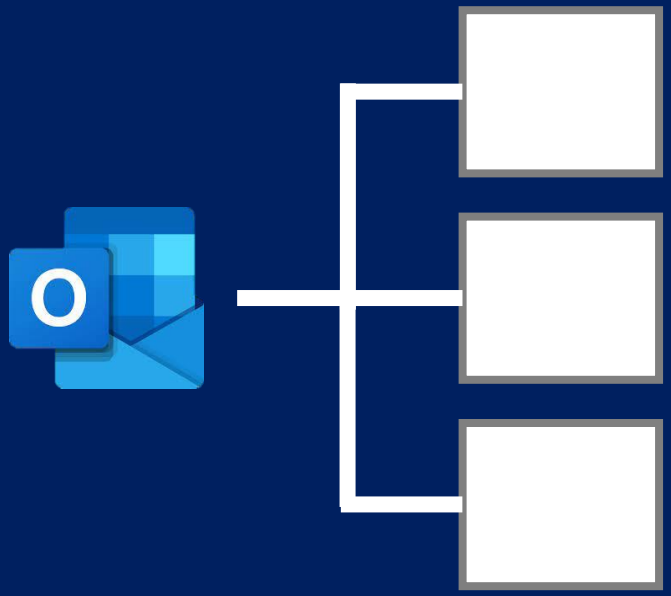
딥러닝 모델을 활용한
아웃룩 메일 분류기 개발 프로젝트

(Outlook E-Mail Classifier)





개발자 프로필

연락처

flash659@gmail.com

www.linkedin.com/in/youngjun-kim-052a25232 (LinkedIn)

대표 보유기술

ML / AI Capabilities - Structured Data, Computer Vision, Recommendation System

Data Analysis

Python

Certifications

SQL Developer

ADsP

YoungJun Kim

I am researching ML / AI. Recently, I am interested in recommendation system and researching it.
서울

간단프로필

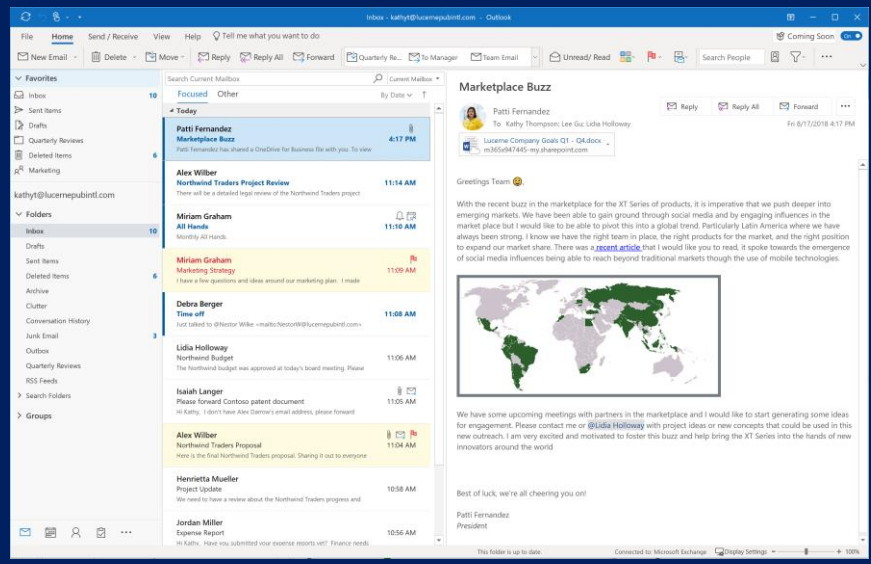
<Career>
2021.12 ~ Present - KMS Manager at 'Attorney At Law Yulchon' in Seoul, Republic of Korea

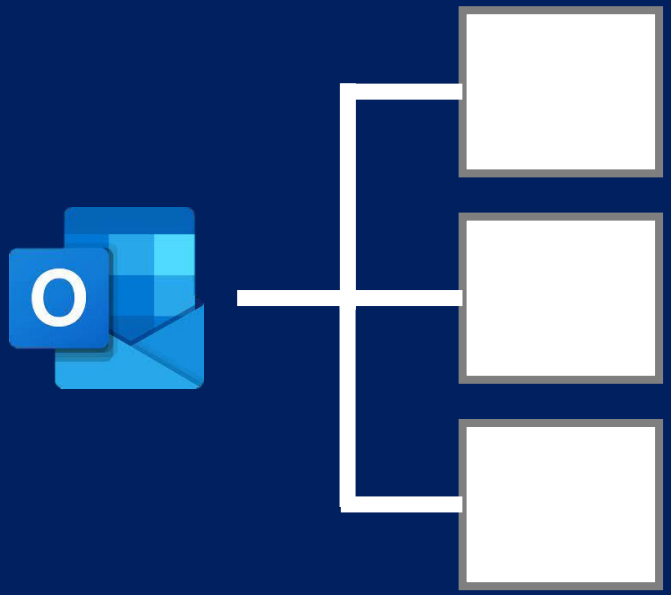
<Skills>
ML / AI Capabilities - Structured Data, Computer Vision, Recommendation System

<Competition>
Top 1% Stock Price Prediction (Structured Data - Regression) 2021.11.10 ~ 2021.11.29 in Dacon
Top 23% Plant Growth Period Prediction (Image - Regression) 2021.11.25 ~ 2021.12.17 in Dacon
Top 57% Pawpularity Contest (Multi Type - Regression) 2021.09.23 ~ 2022.01.14 in Kaggle

<Learning Now>
Tensorflow Recommendation System
Nvidia MERLIN

<Challenging Now>
H&M Personalized Fashion Recommendations (Multi Type - Recommendation) in Kaggle



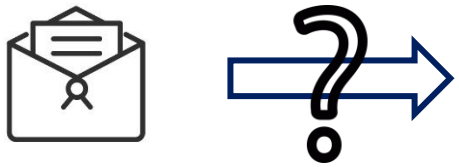


Contents

01 프로젝트 기획 배경

메일을 업무 카테고리에 맞게 분류해 놓고 싶은데,
지금 당장 처리해야할 업무는 계속 들어오고...

"이를 자동으로 분류하게 할 수는 없을까?"



- 사내 공지 메일
- A사 관련 프로젝트
- 타 부서 정기 메일

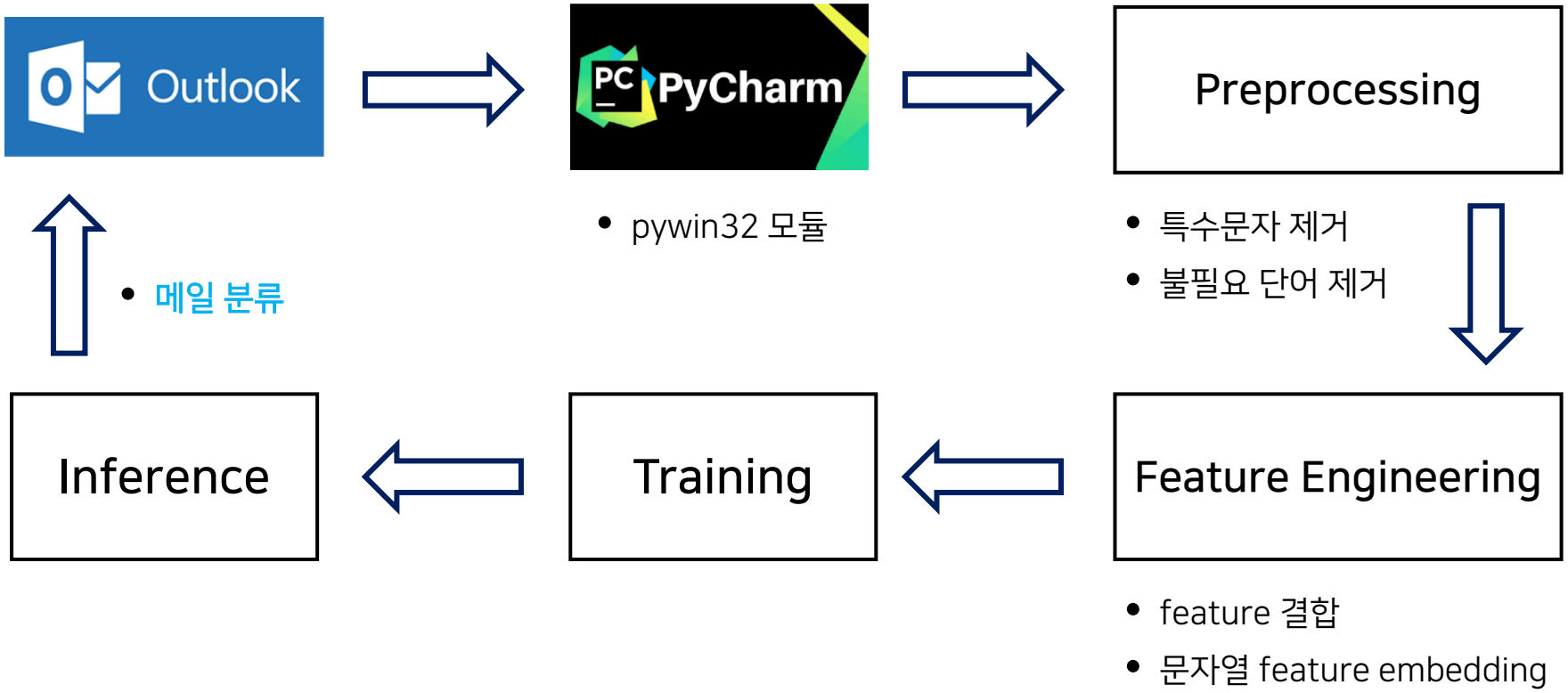
문제 정의

- ✓ 일일이 각 업무에 맞는 메일을 분류하기 번거롭다.
- ✓ 특히, 여러 프로젝트를 동시에 수행해야 하는 직무일수록 메일 분류에 더 많은 번거로움이 있다.
- ✓ overhead cost 중 하나로 볼 수 있다.

딥러닝 모델이 효과적인 이유

- ✓ 분류 작업 전 사전 정보 파악에 시간이 든다. (발신자 및 메일 내용을 모두 파악해야 한다.)
- ✓ 분류하는 프로세스 자체가 복잡하지는 않다. (메일의 정보 해석 ⇒ 분류될 메일함 선택)
- ✓ 메일이 잘못 분류되어도 심각한 위험을 초래하지는 않는다. (메일이 잘못 분류되었는데 그 메일이 중요할 확률은 적고, 오류에 대한 보완 장치로서 Outlook 자체 검색 기능이 있다.)

02 프로세스 개요


Outlook
PyCharm
Preprocessing
• pywin32 모듈
• 특수문자 제거
• 불필요 단어 제거
• 메일 분류
Inference
Training
Feature Engineering
• feature 결합
• 문자열 feature embedding

03 전처리

Feature set 및 Target 정의

Feature set

1. 발신자
2. 수신자 목록
3. 참조 목록
4. 제목
5. 본문 내용
6. 첨부 파일명
7. 첨부 파일 확장자
8. 발송 일시

NN Model

Target

받은 편지함
메일함1
  ㄴ 메일함1-1
메일함2
  ㄴ 메일함 2-1
    ㄴ 메일함 2-1-1
  ㄴ 메일함 2-2
메일함3
  ㄴ 메일함 3-1
  ㄴ 메일함 3-2
...

03 전처리

Feature set 및 Target 정의

Feature set

1. 발신자
2. 수신자 목록
3. 참조 목록
4. 제목
5. 본문 내용
6. 첨부 파일명
7. 첨부 파일 확장자
8. 발송 일시

✓ 받은 편지함 및 받은 편지함 하위 1depth 폴더를 분류될 폴더(Class)로 선정

✓ 받은 편지함 하위 1depth 폴더의 경우 학습 데이터는 하위 폴더들 까지 포함

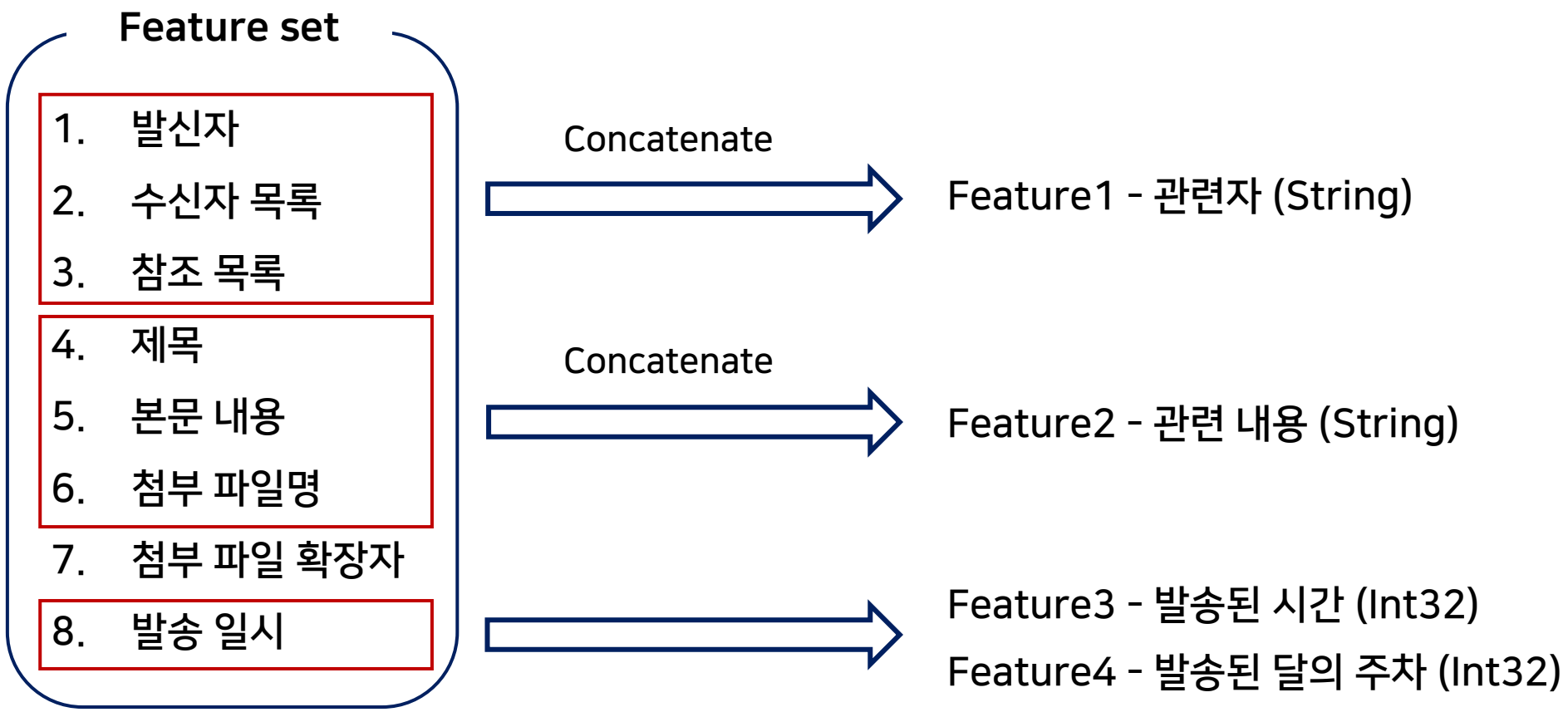
NN Model

메일함3의 학습데이터

Target

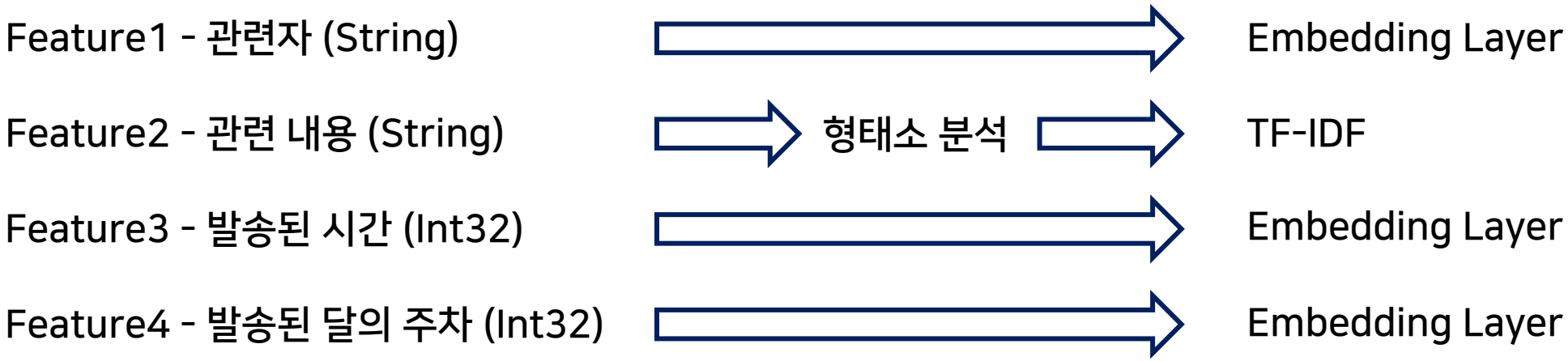
받은 편지함
메일함1
  ㄴ 메일함1-1
메일함2
  ㄴ 메일함 2-1
    ㄴ 메일함 2-1-1
  ㄴ 메일함 2-2
메일함3
  ㄴ 메일함 3-1
  ㄴ 메일함 3-2
...

03 전처리
Feature Engineering



* String의 경우 사전에 특수문자 제거

03 전처리
Feature Engineering



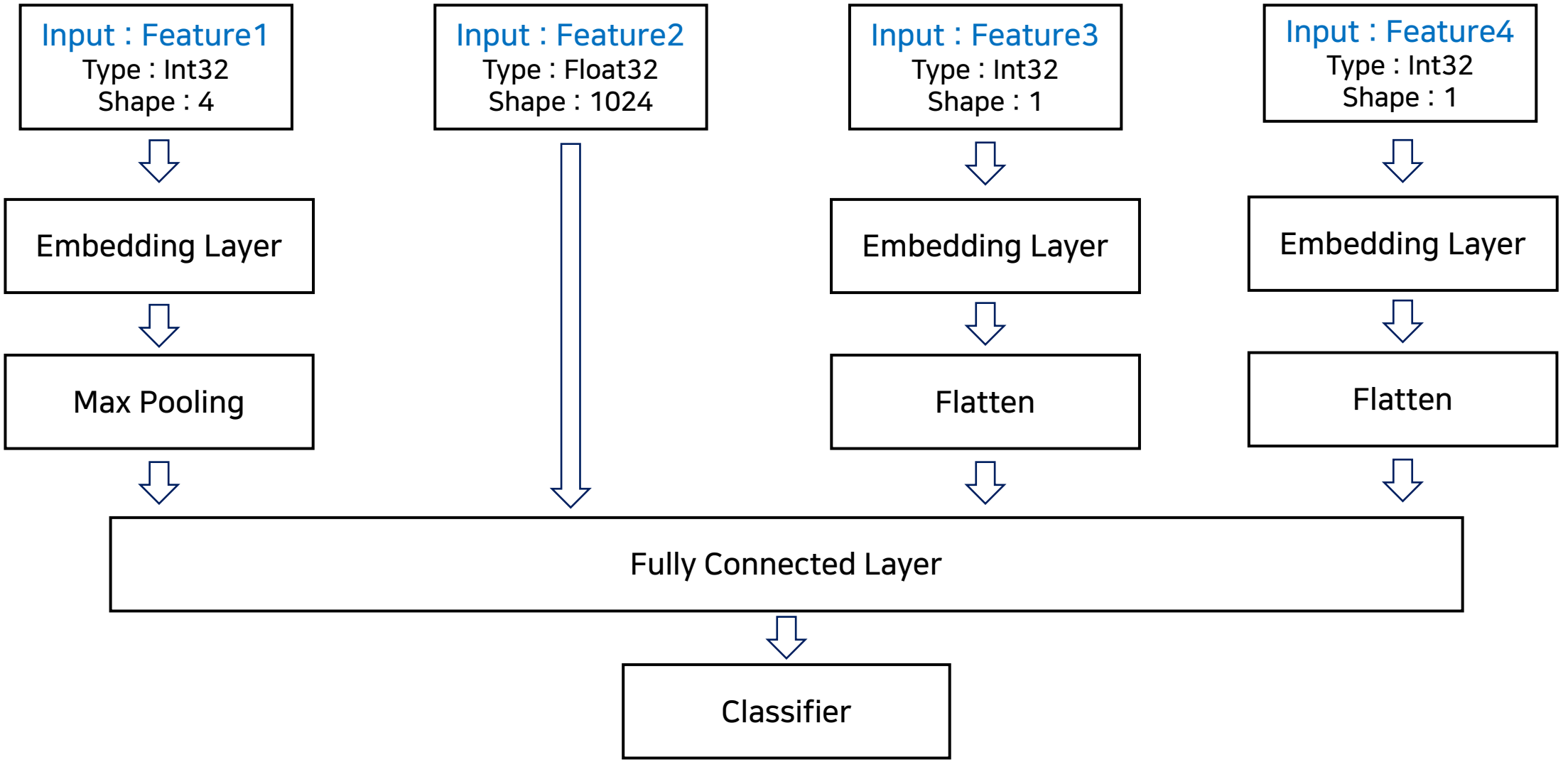
관련 내용(제목 + 본문 내용 + 첨부 파일명)을 TF-IDF로 embedding한 이유

로펌 특성 상 Formal 메일이 다수이기에, 정보가 많지 않은 내용에 대한 중요도를 줄이기 위함

(Ex. 번거로우시겠지만, ~ 부탁드립니다. / ~ 한 점 참고 부탁드립니다.)

* 형태소 분석에는 rhinoMorph 모듈 사용

04 모델링
Model Architecture

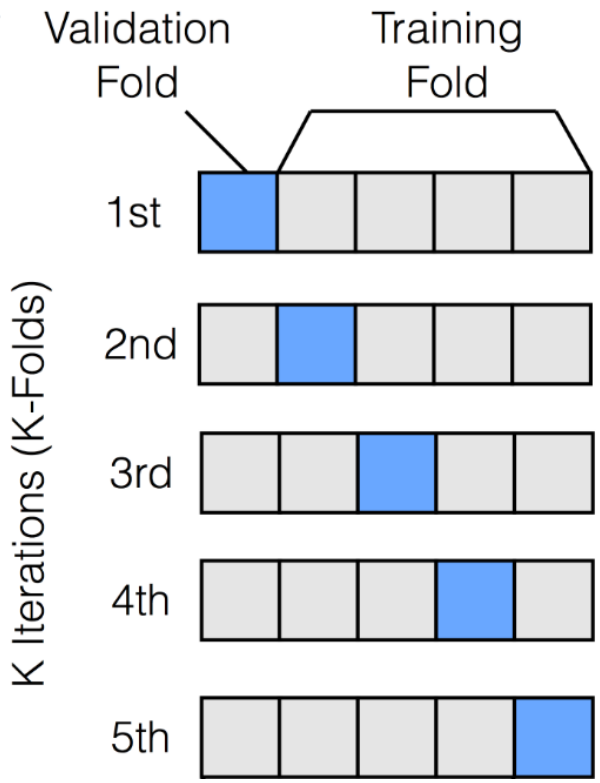


* Feature1(관련자)의 경우 사전에 label encoding 처리

04 모델링

학습 및 추론 방식

- ✓ K-Folds 로 데이터셋을 나눈 후 학습
- ✓ 학습된 K개의 모델의 예측값을 평균하여 추론



모델 하이퍼 파라미터

Optimizer : AdamW

Learning rate : 1e-3

Weight decay : 1e-4

Epochs : 50

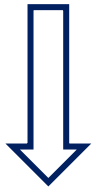
Early stopping epochs : 10

Batch size : 8

K (in training) : 5

05 성능평가

Base Line Model
정확도 : 80% 초반



약 12.5% 성능 개선

모델 고도화
정확도 : 90% 초반

Evaluation Table

	Logloss	Accuracy
fold_0	0.3273	94.32%
fold_1	0.5121	92.95%
fold_2	0.3578	94.99%
fold_3	0.4497	92.71%
fold_4	0.5487	91.80%
Average	0.4391	**93.35%**
Std.	0.0855	0.0115

* 기존 메일이 잘 분류되어 있다고 가정

06 배운 점 & 어려웠던 점

배운 점

- 실제 업무에 딥러닝 기술을 적용해 업무 효율을 올릴 수 있었던 경험이었다.
- TF-IDF가 정말 간단하면서 효율적인 embedding 알고리즘이라는 것을 알았다.
- feature engineering 과정에서 관련 feature들을 묶는 아이디어에 도달하기까지 여러 방법들을 생각해 보았고, 이에 폭넓은 생각을 할 수 있었다.
  (ex. 관련자 : 발신자 + 수신자 목록 + 참조 목록)

06 배운 점 & 어려웠던 점

어려웠던 점

- Outlook의 메일 객체를 Python에서 다루는 방법을 몰라 헤맸지만,
  이 프로젝트를 통해 배울 수 있었다.

- 클래스로 지정된 메일함 하위에 있는 모든 메일 객체들을 로드하는 방법을
  코드로 구현하기 어려웠다. (Recursive 방법을 이용해 해결)

- Outlook 메일함 전체 메일 객체에 대해 반복문을 통해 분류를 하게 되면,
  큐 구조로 인해 적절한 메일함으로 가지 않는 오류가 발생하는데 이 부분을 해결하는 것이 어려웠다.
  (간단히 제일 첫 번째 객체를 n번 꺼네도록 하여 해결)

07 한계점 & 개선 방향

한계점

- 직무마다 주고받는 메일의 특성이 각기 다르므로
  일반화된 성능을 보여주는 단일 모델을 개발하기가 어렵다고 느꼈다.

- 학습데이터 구축하기 위해 사전에 Outlook 메일함이
  분류가 잘 되어 있어야 했었다.

- 참조 목록의 경우 최대 4명 까지만 고려하여 학습했기에,
  참조 목록이 다수로 발송되는 메일들의 특성을 반영하지 못하였다.

07 한계점 & 개선 방향

개선 방향

- 제목과 본문 내용을 하나의 문자열로 보고 학습시켰으나,
  제목은 요약된 정보이므로 본문과 별도로 embedding한 후 학습시킨다.

- 다량의 참조자가 있는 메일의 특성을 반영하게 한다.

- 형태소 분석 이후 SentencePiece Tokenizer 로 한 단계를 더 문자열
  을 분해하여 단어 미만 단위의 특성을 반영하게 한다.
  (Post Tokenizing 단계 추가)

감사합니다

08 Appendix
개발 환경

하드웨어 스펙

CPU : i9-11900K
RAM : 32G

핵심 모듈 버전

Python 3.8.0
Tensorflow 2.6.2

IDE

Pycharm

08 Appendix
구현 코드

```
def get_all_emails(sub_item, cond_time=None):
    if sub_item.Count > 0:
        inbox_dic = pd.DataFrame()
        tmp_subject = []
        tmp_sender_name = []
        tmp_to = []
        tmp_cc = []
        tmp_body = []
        tmp_received_time = []
        tmp_atts = []
        tmp_atts_num = []
        for j in sub_item:...

        inbox_dic["subject"] = tmp_subject
        inbox_dic["subject"].apply(lambda x: x[2:] if x[:2] in ("re", "fw") else x)
        inbox_dic["sender_name"] = tmp_sender_name
        inbox_dic["to"] = tmp_to
        inbox_dic["cc"] = tmp_cc
        inbox_dic["body"] = tmp_body
        inbox_dic["received_time"] = tmp_received_time
        inbox_dic["hour"] = inbox_dic["received_time"].dt.hour.astype("int32")
        inbox_dic["week_of_month"] = inbox_dic["received_time"].apply(week_of_month).astype("int32")
        inbox_dic["atts"] = tmp_atts
        inbox_dic["atts_num"] = tmp_atts_num
        return inbox_dic
def get_all_df(class_folder):
    return_df = dataframe()
    if class_folder.Folders.Count < 1:
        return return_df.append(get_all_emails(class_folder.Items))
    else:
        for i in class_folder.Folders:
            return_df = return_df.append(get_all_df(i))
        return return_df.append(get_all_emails(class_folder.Items))
```

```
text_finder_subject = re.compile('[^ _/A-Za-zㄱ-ㅣ가-힣+]')
text_finder_body = re.compile('[^ _/A-Za-zㄱ-ㅣ가-힣+]')
text_finder_atts = re.compile('[^ ./A-Za-zㄱ-ㅣ가-힣+]')
text_finder = re.compile('[^;A-Za-zㄱ-ㅣ가-힣+]')
rn = rhinoMorph.startRhino()


inbox_dic = dict.fromkeys(class_map["class"])
body_stopwords = [
    "iwl", "yulchoncom", "from", "sent", "pmto", "file", "vs", "re", "f
    "monday", "tuesday", "wednesday", "thursday", "friday", "saturday",
    "january", "february", "march", "april", "may", "june", "july", "au
]

# test_df = get_all_df(inbox.Folders["Z드라이브 - 정기"])
# print(test_df.shape)
# print(test_df.shape[0] == 61 + 66 + 8)
# print(test_df.head())

# 폴더이름 별로 반복
for idx, value in enumerate(tqdm(class_map["class"])):
    # 해당 폴더 및 하위 폴더의 모든 메일을 dataframe 으로 저장 리턴하는 함수 호출
    inbox_dic[value] = get_all_df(inbox.Folders[value])
    # 생서된 dataframe에 클래스 할당
    inbox_dic[value]["class"] = class_map["class_mapped"].iloc[idx]

cond_time = (
    (datetime.now() - DateOffset(days=7)).normalize(),
    pd.to_datetime(datetime.now()).normalize() - DateOffset(seconds=1)
)
```