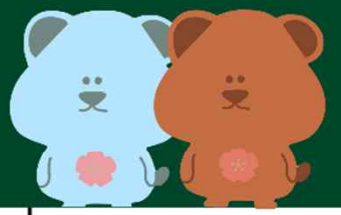
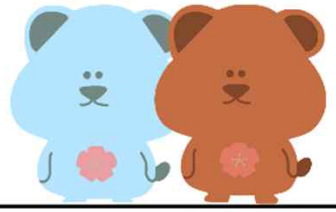


학교 공지사항 분류 프로젝트

최은빈 문가을 이주원 주민서

EWHA EURON

목차



EWHA EURON

프로젝트 소개

EWHA EURON

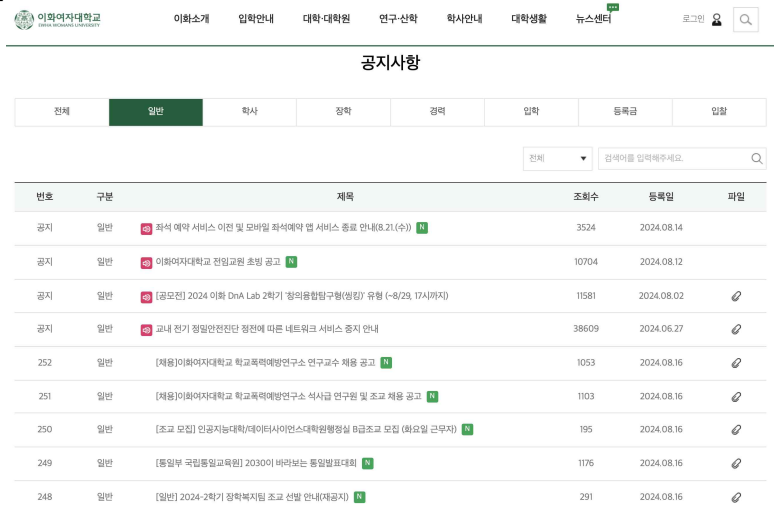
#01 프로젝트 소개

프로젝트의 필요성 및 개요

학교 공지사항에서 학사, 경력, 장학 등 카테고리의 경계가 모호하여 사용자 입장에서 원하는 정보를 빠르고 정확하게 얻지 못하는 점에서 불편을 느낌.

특히, 홈페이지의 주 사용층인 학생의 주요 관심사는 **스펙을 쌓을 수 있는 활동에 대한 공지사항**임.

공지사항을 스펙이 되는 것(1)과 그렇지 않은 것(0)으로 분류할 수 있는 모델을 구축하고자 함



데이터 크롤링

EWHA EURON

#02 데이터 크롤링

Train Data – 9782개

- 이화여대, 홍익대, 성균관대, 연세대, 경희대, 숭실대

Test Data – 976개

- 서울대, 숙명여대, 국민대

* 대학 선택 기준 - 서울 소재 대학 중 임의로 선택

EWHA EURON

#02 데이터 크롤링

```python
# 헤더 정의 -> 웹 페이지에 접근할 수 있도록
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'}

titles = []
categories = []

def title_scrapping(start_page, end_page):

    for page in range(start_page, end_page + 1):

        print(f"👉{page} 페이지")

        # offset 계산 (0부터 10단위로 증가)
        offset = (page - 1) * 10
        url = f'https://www.skku.edu/skku/campus/skk_comm/notice01.do?mode=list&&articleLimit=10&article.offset={offset}'

        # request + html 소스 가져오기
        html = requests.get(url, headers = headers)

        # bs 객체 생성
        soup = BeautifulSoup(html.text, 'html.parser')
        # dt 태그 + class_ 인 것을 모두 찾기
        board_tags = soup.find_all('dt', class_ = "board-list-content-title")

        # title
        for board_tag in board_tags:
            a_tag = board_tag.find('a', href = True)
            title = a_tag.text.strip()
            titles.append(title)

            category_tag = board_tag.find('span', class_ = "c-board-list-category")
            category = category_tag.text.strip().replace('[', '').replace(']', '')
            categories.append(category)

        for idx in range(offset, offset + 10):
            text = titles[idx]
            print(text)

            detail = categories[idx]
            print(detail)
        print('\n')
```

title scrapping 함수 정의
-> BeautifulSoup 라이브러리를 사용해
원하는 페이지의 원하는 부분을 크롤링

```python
title_scrapping(1, 250)
```

```
👉34 페이지
LS드림사이언스클래스 20기 대학생 멘토 모집 안내
채용/모집
2024년 경기도 GenAI·공공데이터 창업경진대회 개최 안내
행사/세미나
스페셜올림픽코리아 <2024 국제 스페셜 뮤직&아트 페스티벌> 메이트 및 자원봉사자 모집 안내
채용/모집
[의료인공지능융합인재양성사업단] 2024년 대한의료정보학회 춘계학술대회 참가 안내(~6/2)
행사/세미나
2024-2025 불가리아 정부초청 장학생 선발 안내
채용/모집
블룸버그 터미널 이용교육 신청 안내
행사/세미나
제 19회 소셜이노베이션 해외학자 초청 집중 특강(일시: 2024년 6월 11일 (화) 오후 3시~5시)
행사/세미나
성균인문동양학아카데미(S-AHA) 명사특강 개최, 강연 신청 안내
행사/세미나
2024년 소선나눔기금 과학기술분야 대학(원)생 발굴 장학금 신청 안내
장학
[반도체소부장혁신융합대학사업단] 차세대반도체와 반도체소부장이 함께하는 TCAD를 이용한 공정,소자 Simulation 특강 모집 안내
채용/모집
```

EDA

EWHA EURON

#03 EDA

(1) 한글 폰트 설치

- 나눔 폰트

```
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  fonts-nanum
```

#03 EDA

(2) 결측치 확인 & 중복 제거

- 결측치 없음
- 중복된 공지사항 확인 및 제거

```
print(df[df.duplicated()])
```

```
                                                 title  label
101                2019학년도 겨울 계절제수업 폐강 과목 안내(1차)      0
147                                학사경고자 학사지도 안내       0
267              졸업앨범 미촬영자 앨범비 환불신청 안내(신촌캠 학부생)      0
279                                학생 상해보험 제도 안내       0
379              졸업앨범 미촬영자 앨범비 환불신청 안내(신촌캠 학부생)      0
...                                                ...    ...
9707                          생활관(우정원) 기숙사 조교 모집      0
9713                 국제처 (국제)글로벌교육지원팀 계약직 채용 공고       1
9715                 입학처 2023-1학기 하계방학 근로학생 모집 ★      0
9734                  ★국제C 재무회계팀 기간제 근로자 채용 공고★       1
9757  [추천채용] [포항공과대학교기술지주(주)] 관리부문 신입사원 추천채용 공고      1

[359 rows x 2 columns]
```

```
df.drop_duplicates(inplace = True)
```

```
df
```

	title	label
0	재학생 영어진단평가 신청 및 일정 안내	0
1	울산대학교 2019년도 2학기 국내교환대학 학점교류 수학 안내 만료	0
2	2019학년도 여름계절제 학부 강의정보 공유를 위한 설문 실시 안내 만료	0
3	울산과학기술원 2019학년도 2학기 국내대학 학점교류 수학 안내 만료	0
4	한국과학기술원 2019학년도 2학기 국내교환 학점교류 수학 안내 만료	0
...	...	...
9777	[추천채용] [김.장 법률사무소] 비서,빌링 정규직 추천채용 안내	1
9778	예술디자인대학 시각디자인학과 조교 채용 공고	0
9779	재무회계팀(국제) 근로학생 모집합니다.	0
9780	생활관(제2기숙사)기숙사 조교 모집 안내	0
9781	[국가근로]23-1학기 칠보청소년문화의집(교외) 국가근로 장학생 모집	0

9423 rows × 2 columns

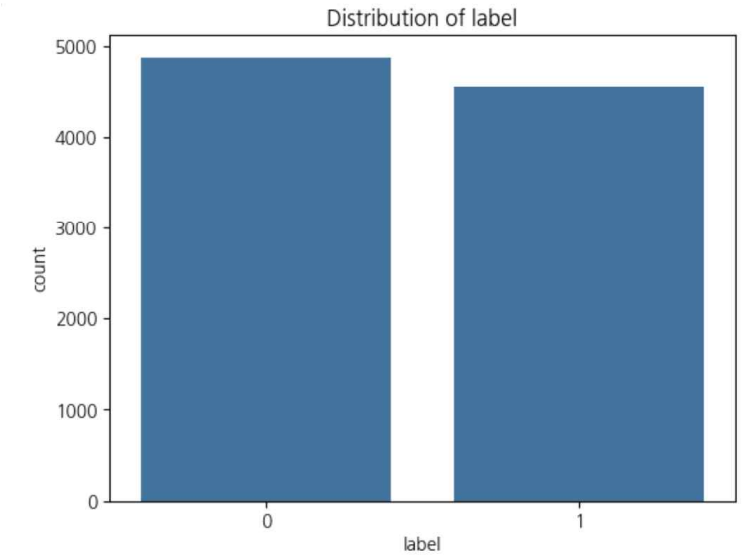
EWHA EURON

#03 EDA

(3) class balance 확인

- Train data의 class imbalance는 smote/ oversampling / undersampling과 같은 데이터 조작을 필요로 할 수 있음 -> 확인해야 할 중요한 요소
- 0(4873) / 1(4550) -> **balanced class distribution**

```
#label 분포 확인
sns.countplot(x='label', data=df)
plt.title('Distribution of label')
plt.show()
```

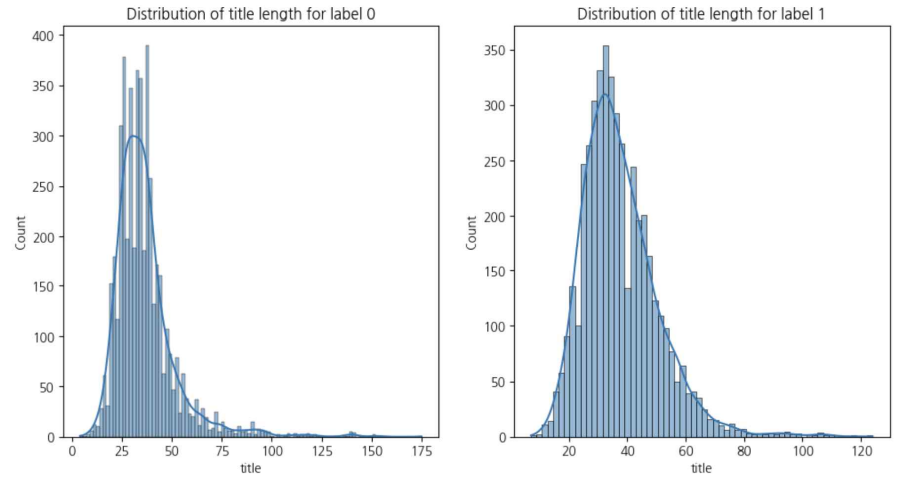


#03 EDA

(4) title length 파악

- label = 0과 label = 1 각각의 텍스트 길이의 전반적인 분포를 파악
- 클래스에 따라 텍스트 길이가 크게 다르지 않음을 확인

```
fig, ax = plt.subplots(1, 2, figsize=(12, 6))

#label= 0일 때의 텍스트 길이 분포
sns.histplot(df[df['label'] == 0]['title'].str.len(), kde=True, ax = ax[0])
ax[0].set_title('Distribution of title length for label 0')

#label= 1일 때의 텍스트 길이 분포
sns.histplot(df[df['label'] == 1]['title'].str.len(), kde=True, ax = ax[1])
ax[1].set_title('Distribution of title length for label 1')

plt.show()
```

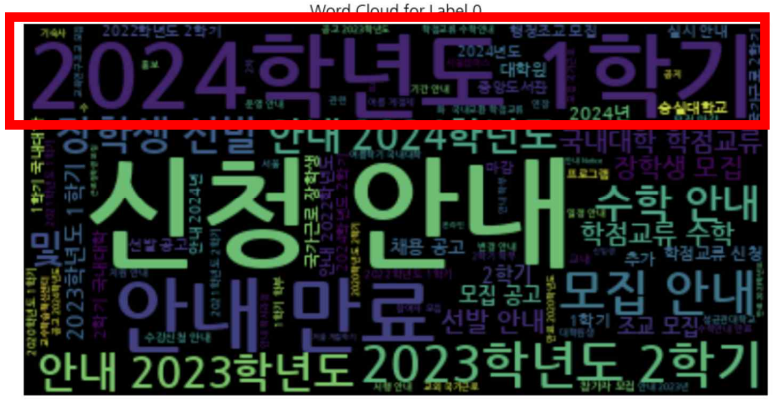


#03 EDA

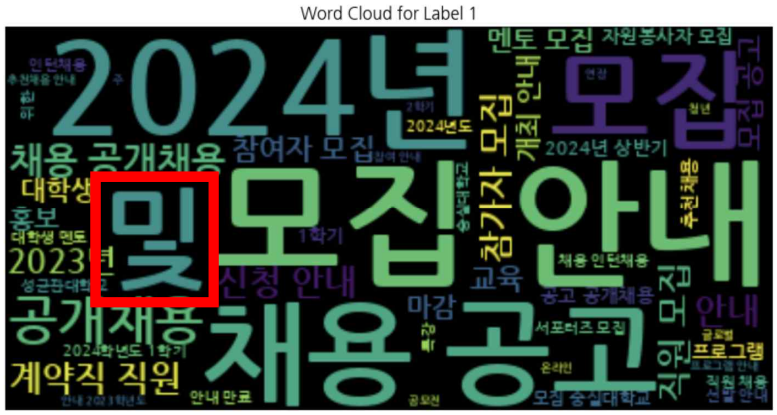
(5) Word Cloud (ver. stopword 제거 안 함)

- label = 0의 word cloud & label = 1의 word cloud
- stopword (불용어) 제거 전이므로 분류에 관련 없는 숫자, 대학 이름, 접속어 등등이 포함되어 있음.

```
#label = 0 의 word cloud
text_label_0 = ' '.join(df[df['label'] == 0]['title'].astype(str))
wordcloud_label_0 = WordCloud(font_path=font_path, background_color='black').generate(text_label_0)

plt.figure(figsize=(10, 6))
plt.imshow(wordcloud_label_0, interpolation='bilinear')
plt.title('Word Cloud for Label 0')
plt.axis('off')

plt.show()
```




```
#label = 1 의 word cloud
text_label_1 = ' '.join(df[df['label'] == 1]['title'].astype(str))
wordcloud_label_1 = WordCloud(font_path=font_path, background_color='black').generate(text_label_1)

plt.figure(figsize=(10, 6))
plt.imshow(wordcloud_label_1, interpolation='bilinear')
plt.title('Word Cloud for Label 1')
plt.axis('off')

plt.show()
```



전처리

EWHA EURON

#04 전처리

(1) 출현 빈도가 낮은 단어 제거

```
# 모든 단어의 빈도수 계산

all_words = ' '.join(df['title']).split()
word_counts = Counter(all_words)

# 빈도수가 3번 이하인 단어 제거
def filter_infrequent_words(text, word_counts, threshold=3):
    words = text.split()
    filtered_words = [word for word in words if word_counts[word] > threshold]
    return ' '.join(filtered_words)

# 'title' 열에 함수 적용
df['title'] = df['title'].apply(lambda x: filter_infrequent_words(x, word_counts))
```

(2) 크롤링 해온 학교 명을 95% 정도 drop ➡ 특정 학교명으로 학습 되는 일 방지

```
school_words = ['신촌', '이화여자대학교', '이대', '이화여대' '이화', '이화인','성균관대학교', '성대',
                '성균', '성균인', '홍익대학교', '홍대', '홍익', '홍익인', '숭실대학교', '숭실대',
                '숭실', '숭실인', '연세대학교', '연대', '연세', '연세인', '경희', '경희대', '서강']

probability = 0.95

# 단어 제거 함수
def probabilistic_remove(word, text, probability):
    if np.random.rand() < probability:
        return text.replace(word, '')
    return text

# 각 행에 대해 school words를 95% 확률로 제거
for school_word in school_words:
    df['title'] = df['title'].apply(lambda x: probabilistic_remove(school_word, x, probability))
```

EWHA EURON

#04 전처리

(3) title에 영어와 한국어만 남기기

	title
0	재학생 신청 및 일정 안내
1	울산대학교 2019년도 2학기 학점교류 수학 안내 만료
2	2019학년도 여름계절제 학부 강의정보 공유를 위한 설문 실시 안내 만료
3	울산과학기술원 2019학년도 2학기 국내대학 학점교류 수학 안내 만료
4	한국과학기술원 2019학년도 2학기 국내교환 학점교류 수학 안내 만료
...	...
9777	[추천채용]정규직 추천채용 안내
9778	예술디자인대학 조교 채용 공고
9779	근로학생 모집합니다.
9780	조교 모집 안내
9781	국가근로 장학생 모집

9782 rows × 1 columns

	title
0	재학생 신청 및 일정 안내
1	울산대학교 년도 학기 학점교류 수학 안내 만료
2	학년도 여름계절제 학부 강의정보 공유를 위한 설문 실시 안내 만료
3	울산과학기술원 학년도 학기 국내대학 학점교류 수학 안내 만료
4	한국과학기술원 학년도 학기 국내교환 학점교류 수학 안내 만료
...	...
9777	추천채용 정규직 추천채용 안내
9778	예술디자인대학 조교 채용 공고
9779	근로학생 모집합니다
9780	조교 모집 안내
9781	국가근로 장학생 모집

9782 rows × 1 columns

#04 전처리

(4) Custom words 와 stop words 정의

'장학생'과 같은 주요 키워드를 '장', '학생' 으로 잘못 인식하는 등의 문제 발생.

➡ custom words를 정의해 사용자 지정 사전 제작

재학생	NNG	
학부생	NNG	
타대생	NNG	
수강	NNG	
이수	NNG	
워크샵	NNG	
전형	NNG	
예정자	NNG	
신입생	NNG	
수강생	NNG	
수강신청	NNG	
비대면	NNG	
국가직	NNG	
철회	NNG	
미납자	NNG	
앰배서더	NNG	
미취득자	NNG	
미개설	NNG	
미참여	NNG	

(5) 한 글자 단어들은 drop 하되 '팀 ' 과 '랩 ' 은 남겨두기

	title	label	processed_title
547	총무팀 국제캠퍼스 셔틀버스 따른 장소 변경	0	총무 팀 국제 캠퍼스 셔틀버스 장소 변경
1276	년 대학원생 창업경진대회 개최 및 참가팀 모집	0	대학원 창업 경진 대회 개최 참가 팀 모집
1349	x 대학원생 창업경진대회 참가팀 모집	1	대학원 창업 경진 대회 참가 팀 모집
1409	더 동아리 지원 사업 선정 팀 결과 발표 만료	1	동아리 지원 사업 선정 팀 결과 발표
1428	학생인재개발팀 대학일자리플러스센터 여름방학 아카데미	1	학생 인재 개발 팀 대학 일자리 플러스 센터 여름 방학 아카데미
...	...	...	...
9676	학기 국제교류팀 조교 모집	0	국제 교류 팀 조교 모집
9694	학사지원팀 학기 조교모집 공고	0	학사 지원 팀 조교 모집 공고
9714	글로벌봉사팀 캠페인 in the khu	1	글로벌 봉사 팀 캠페인 khu
9731	글로벌봉사팀 하계 국가근로 학생	0	글로벌 봉사 팀 하계 국가 근로 학생
9734	재무회계팀 기간제 근로자 채용 공고	1	재무회계 팀 기간 근로자 채용 공고

298 rows × 3 columns

모델링

EWHA EURON

#05 모델링(1) – Machine Learning Model

임베딩(Embedding)

ⅰ) N-gram 벡터화
: 단어의 앞뒤 맥락 정보를 더 잘 파악하기 위해 n-gram vectorize 수행

ⅱ) Kobert Tokenizer
: 한국어의 형태소 분석을 반영하여 의미 있는 임베딩을 생성할 수 있음

```
#중요도 up 단어 플래그
high_priority_words = ['학회', '프로그램', '서포터즈', '인턴', '공모전', '교환', '실습', '취업',
                       '면접', '자소서', '자격증', '고시', '트랙', '교수', '조교', '대학원생', '팀플', '교양', 'ChatGPT', '챗지피티',
                      '계절', '채플', 'mbti', 'MBTI']


df['high_priority_flag'] = df['processed_title'].apply(lambda x: 1 if any(word in x for word in high_priority_words) else 0)
X_high_priority = df['high_priority_flag'].values.reshape(-1, 1)

# 모든 특징 결합
X_combined = np.hstack((X_high_priority, X_ngrams, X_embeddings))

# 라벨
y = df['label'].values
```

임베딩 및 중요도를 높인 단어들의 특징을 X_combine 이라는 변수로 스태킹

#05 모델링(1) – Machine Learning Model

Random Forest

```python
# 랜덤 포레스트 모델 학습 3
RF_3 = RandomForestClassifier(n_estimators=300, max_depth = None, min_samples_split = 2, min_samples_leaf = 1,
                              max_features = 'log2', random_state=36)
RF_3.fit(X_train, y_train)

# 예측 및 평가
y_pred = RF_3.predict(X_test)

# 평가 지표 출력
print(f'Accuracy: {accuracy_score(y_test, y_pred): .4f}')
print(f'Precision: {precision_score(y_test, y_pred, average="weighted"): .4f}')
print(f'Recall: {recall_score(y_test, y_pred, average="weighted"): .4f}')
print(f'F1 Score: {f1_score(y_test, y_pred, average="weighted"): .4f}')
print(classification_report(y_test, y_pred))
```

<일반 버전>

```
Accuracy:  0.8456
Precision:  0.8478
Recall:  0.8456
F1 Score:  0.8457
```

=

```python
RF_3 = RandomForestClassifier(n_estimators=300, max_depth = None, min_samples_split = 2, min_samples_leaf = 1,
                              max_features = 'log2', random_state=36)
RF_3.fit(X_train_weighted, y_train)

# 예측 및 평가
y_pred = RF_3.predict(X_test_weighted)

# 평가 지표 출력
print(f'Accuracy: {accuracy_score(y_test, y_pred): .4f}')
print(f'Precision: {precision_score(y_test, y_pred, average="weighted"): .4f}')
print(f'Recall: {recall_score(y_test, y_pred, average="weighted"): .4f}')
print(f'F1 Score: {f1_score(y_test, y_pred, average="weighted"): .4f}')
print(classification_report(y_test, y_pred))
```

<특정 단어 중요도 높인 버전>

```
Accuracy:  0.8456
Precision:  0.8478
Recall:  0.8456
F1 Score:  0.8457
```

EWHA EURON

#05 모델링(1) – Machine Learning Model

Naïve Bayes

(1) Multinomial Naïve Bayes

```
# Multinomial Naive Bayes

# 데이터에 음수가 없도록 0과 1사이로 스케일링
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)

# 모델 훈련
MNB = MultinomialNB()
MNB.fit(X_train, y_train)

# 예측
y_pred = MNB.predict(X_test)
```

<일반

```
Accuracy:  0.8844
Precision:  0.8962
Recall:  0.8844
F1 Score:  0.8841
```

<

```
# 데이터에 음수가 없도록 0과 1사이로 스케일링
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)

# 중요 단어 가중치 높이기
X_train_weighted = np.copy(X_train)
X_test_weighted = np.copy(X_test)

X_train_weighted[:, 0] *= 2.0  # high_priority_flag의 가중치를 2.0으로 설정
X_test_weighted[:, 0] *= 2.0

# 모델 훈련
MNB_2 = MultinomialNB()
MNB_2.fit(X_train_weighted, y_train)

# 예측
y_pred = MNB_2.predict(X_test_weighted)
```

<특정 단어 중요도 높인 버

```
Accuracy:  0.8850
Precision:  0.8966
Recall:  0.8850
F1 Score:  0.8847
```

EWHA EURON

#05 모델링(1) – Machine Learning Model

Naïve Bayes

(2) Bernoulli Naïve Bayes

```
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)

# 모델 훈련
BNB = BernoulliNB()
BNB.fit(X_train_scaled, y_train)

# 예측
y_pred = BNB.predict(X_test_scaled)
```

<일반

```
Accuracy:  0.8942
Precision:  0.9006
Recall:  0.8942
F1 Score:  0.8941
```

=

```
# 중요 단어 가중치 높이기
X_train_weighted = np.copy(X_train_scaled)
X_test_weighted = np.copy(X_test_scaled)

X_train_weighted[:, 0] *= 2.0  # high_priority_flag의 가중치를 2.0으로 설정
X_test_weighted[:, 0] *= 2.0

# 모델 훈련
BNB_2 = BernoulliNB()
BNB_2.fit(X_train_weighted, y_train)

# 예측
y_pred = BNB_2.predict(X_test_weighted)
```

<특정 단어 중요도 높인 버전

```
Accuracy:  0.8942
Precision:  0.9006
Recall:  0.8942
F1 Score:  0.8941
```

EWHA EURON

#05 모델링(1) – Machine Learning Model

Logistic Regression

(1) Plain Version

X_train <- X_embedding만 있는 데이터 사용

```
# 모델 학습
LR_2 = LogisticRegression(max_iter=1000)
LR_2.fit(X_train, y_train)

# 예측 및 평가
y_pred = LR_2.predict(X_test)
```

```
# 중요 단어 가중치 높이기
X_train_weighted = np.copy(X_train)
X_test_weighted = np.copy(X_test)

X_train_weighted[:, 0] *= 2.0  # high_priority_flag의 가중치를 2.0으로 설정
X_test_weighted[:, 0] *= 2.0

# 모델 훈련
LR_3 = LogisticRegression(max_iter=1000)
LR_3.fit(X_train_weighted, y_train)

# 예측
y_pred = LR_3.predict(X_test_weighted)
```

<일반

```
Accuracy:  0.7052
Precision:  0.7185
Recall:  0.7052
F1 Score:  0.7033
```

<특정 단어 중요도 높인 버전>

```
Accuracy:  0.7041
Precision:  0.7172
Recall:  0.7041
F1 Score:  0.7022
```

#05 모델링(1) – Machine Learning Model

Logistic Regression

(2) Optimal hyperparameters with GridSearch

X_train -> X_combined 사용함.(X_embedding, X_ngram, X_high_priority합쳐진 데이터)

```
# 모델 학습
LR_4 = LogisticRegression(max_iter=1000, C=1, penalty='l2', solver='lbfgs')
LR_4.fit(X_train, y_train)

# 예측 및 평가
y_pred = LR_4.predict(X_test)
```

```
# 중요 단어 가중치 높이기
X_train_weighted = np.copy(X_train)
X_test_weighted = np.copy(X_test)

X_train_weighted[:, 0] *= 2.0  # high_priority_flag의 가중치를 2.0으로 설정
X_test_weighted[:, 0] *= 2.0

# 모델 훈련
LR_5 = LogisticRegression(max_iter=1000, C=1, penalty='l2', solver='lbfgs')
LR_5.fit(X_train_weighted, y_train)

# 예측
y_pred = LR_5.predict(X_test_weighted)
```

<일반 버전>

```
Accuracy:  0.9131
Precision:  0.9144
Recall:  0.9131
F1 Score:  0.9131
```

=

<특정 단어 중요도 높인 버전>

```
Accuracy:  0.9131
Precision:  0.9144
Recall:  0.9131
F1 Score:  0.9131
```

#05 모델링(2) – Deep Learning Model

Multilingual BERT

- 다양한 언어로 이루어진 텍스트를 이해할 수 있도록 설계된 BERT -> 일반화된 언어 표현을 학습
- trained with self-supervised fashion (자기지도학습)
- **Pretrained Objectives**
  (1) Masked Language Modeling(MLM): randomly masks 15% of the input words -> learn bidirectional representation of word
  (2) Next Sentence Prediction(NSP): concatenates two masked sentences -> predict whether two sentences are successive or not

- This way, M-BERT learns an inner representation of languages that can be useful in extracting features for each downstream task.

#05 모델링(2) – Deep Learning Model

Multilingual BERT

(1) preprocessed title의 시작에 [CLS], 끝에 [SEP]를 붙임

- [CLS]: classification의 약자. 문장의 시작에 붙임으로써 모델로 하여금 이 위치에서 특징 벡터를 추출하도록 함
- [SEP]: separator의 약자. 문장의 끝에 붙임으로써 모델로 하여금 이 위치에서 문장이 끝나고 새로운 문장이 시작하는 것을 명시함

```
X_train_fixed = ['[CLS]' + str(text) + '[SEP]' for text in X_train]
X_test_fixed = ['[CLS]' + str(text) + '[SEP]' for text in X_test]

print(X_train_fixed)
```

```
['[CLS]세종 캠퍼스 사회봉사 신청 안내[SEP]', '[CLS]공개 채용 서울 과학 종합 대학원 신입 직원 채용 공고[SEP]',
```

#05 모델링(2) – Deep Learning Model

Multilingual BERT

(2) 문장의 단어를 tokenizing

```
# tokenizing using multilingual

tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased', do_lower_case = False)
tokenized_X_train = [tokenizer.tokenize(fixed) for fixed in X_train_fixed]
tokenized_X_test = [tokenizer.tokenize(fixed) for fixed in X_test_fixed]
```

```
['[CLS]', '세', '##종', '캠', '##퍼', '##스', '사', '##회', '##봉', '##사', '신', '##청', '안', '##내', '[SEP]']
```

BERT 토큰화 방법

1. 단어가 단어 집합에 존재한다.
=> 해당 단어를 분리하지 않는다.

2. 단어가 단어 집합에 존재하지 않는다.
⇒ 해당 단어를 서브워드로 분리한다.
=> 해당 단어의 첫번째 서브워드를 제외한 나머지 서브워드들은 앞에 "##"를 붙인 것을 토큰으로 한다.

#05 모델링(2) – Deep Learning Model

Multilingual BERT

(3) Token을 대응하는 id로 나타냄

```
X_train_ids = [tokenizer.convert_tokens_to_ids(fixed) for fixed in tokenized_X_train]
X_test_ids = [tokenizer.convert_tokens_to_ids(fixed) for fixed in tokenized_X_test]
```

```
[101, 9435, 22200, 9795, 68984, 12605, 9405, 14863, 118989, 12945, 9487, 40311, 9521, 31605, 102]
```

#05 모델링(2) – Deep Learning Model

Multilingual BERT

(4) pad sequence를 통해 시퀀스 데이터를 지정한 길이(max len)으로 맞춰줌

```
# 최대 길이가 128로 설정, 나머지 빈 부분은 0으로 패딩해줌
max_len = 128
# pad_sequences: 시퀀스 데이터를 일정한 길이(max_len)로 맞추는 데 사용

X_train_ids = pad_sequences(X_train_ids, maxlen = max_len, dtype = 'long', padding = 'post', truncating = 'post')
X_test_ids = pad_sequences(X_test_ids, maxlen = max_len, dtype = 'long', padding = 'post', truncating = 'post')
```

```
[    101    9435   22200    9795   68984   12605    9405   14863  118989   12945
    9487   40311    9521   31605     102       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0       0       0
       0       0       0       0       0       0       0       0]
```

#05 모델링(2) – Deep Learning Model

Multilingual BERT

(5) 문장의 attention mask를 만듦 -> id가 존재하면 1.0, 패딩이면 0.0으로 mask가 생성됨

```
# masking
mask_train = []
attention_mask_train = []

for id in X_train_ids: # X_train_ids는 이미 토큰화 + 임베딩 + 패딩까지 완료된 상태, id는 한 문장에 들어있는 토큰들의 id list
  mask_train = [float(i>0) for i in id] # 패딩 토큰(0)에는 0.0, 실제 토큰에는 1.0으로 채워진 마스크 생성함으로써 패딩 토큰은 고려하지 않게 됨
  attention_mask_train.append(mask_train) # 각 id list마다 mask가 만들어짐

mask_test = []
attention_mask_test = []

for id in X_test_ids:
  mask_test = [float(i>0) for i in id]
  attention_mask_test.append(mask_test)
```

#05 모델링(2) – Deep Learning Model

Multilingual BERT

(6) Hyperparameter setting: 배치 사이즈, 에폭 수, learning rate scheduler 조절을 통해 최적의 하이퍼파라미터를 찾음

```
# hyperparameter setting

BATCH_SIZE = 32
EPOCHS = 10

STEP_SIZE = 3
GAMMA = 0.1

DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
optimizer = AdamW(model.parameters(), lr = 3e-5, eps = 1e-8)
lr_scheduler = lr_scheduler.StepLR(optimizer, step_size = STEP_SIZE, gamma = GAMMA)
```

```
=====EPOCH 1 / 10======
학습율: 3e-05
-----train-----
training loss: 0.40767
training acc: 0.82074
-----eval-----
eval loss: 0.28807
eval acc: 0.88479
=====EPOCH 2 / 10======
학습율: 3e-05
-----train-----
training loss: 0.25796
training acc: 0.89455
-----eval-----
eval loss: 0.26057
eval acc: 0.88691
=====EPOCH 3 / 10======
학습율: 3e-05
-----train-----
training loss: 0.21890
training acc: 0.91065
-----eval-----
eval loss: 0.26988
eval acc: 0.89915
=====EPOCH 4 / 10======
학습율: 3e-06
-----train-----
training loss: 0.15809
training acc: 0.93451
-----eval-----
eval loss: 0.25190
eval acc: 0.90709
=====EPOCH 5 / 10======
학습율: 3e-06
-----train-----
training loss: 0.14444
training acc: 0.94084
-----eval-----
eval loss: 0.26320
eval acc: 0.90338
=====EPOCH 6 / 10======
학습율: 3e-06
-----train-----
training loss: 0.13271
training acc: 0.94584
-----eval-----
eval loss: 0.27557
eval acc: 0.90444
=====EPOCH 7 / 10======
학습율: 3.0000000000000004e-07
```

최대 eval 정확도: 0.90709

#05 모델링(2) – Deep Learning Model

Multilingual BERT

(7) Overfitting을 막기 위해 early stopping 기법 적용: eval_loss가 세 번 동안이나 개선되지 않은 경우, overfitting이라고 판단하고 훈련 종료

```
best_loss = 10**9
patience_limit = 3
patience_check = 0
```

```
if eval_loss > best_loss: # loss가 개선되지 않은 경우
  patience_check += 1

  if patience_check >= patience_limit: # early stopping 조건 만족(즉, eval loss가 3번 동안이나 개선되지 않는 경우)
    break

else: # eval loss가 개선되는 경우
  best_loss = eval_loss
  patience_check = 0
```

#05 모델링(2) – Deep Learning Model

Multilingual BERT with Adapter Hub

- Adapter란?: LLM의 파라미터를 freeze시키고 모델에 일부 레이어를 삽입해 downstream task에 맞게 조정하는 모듈. 모든 파라미터를 재학습시키는 것이 아니라 필요한 부분만 학습시킴으로써 빠르고 효율적임.

➡ 단순히 hyperparameter만 조절해서 최적화하는 것이 아니라, 최적화를 도울 수 있는 새로운 레이어를 모델에 적용해보고자 함.

#05 모델링(2) – Deep Learning Model

Multilingual BERT with Adapter Hub

- Adapter를 M-BERT에 추가

```
from adapters import AutoAdapterModel

config = BertConfig.from_pretrained("bert-base-multilingual-cased",num_labels=2)
model = AutoAdapterModel.from_pretrained("bert-base-multilingual-cased", config=config)

#Add a new adapter
model.add_adapter("notice", config="seq_bn")

#Add a matching classification head
model.add_classification_head(
    "notice",
    num_labels=2,
    id2label={0:"진로 아님", 1: "진로"}
)

#Activate the adapter
model.train_adapter("notice")
```

```
trainer.train()
```

[1044/1044 04:58, Epoch 6/6]

Step	Training Loss
200	0.490000
400	0.379500
600	0.316600
800	0.306100
1000	0.285400

```
trainer.evaluate()
```

[58/58 00:08]

```
{'eval_loss': 0.2993907034397125,
 'eval_acc': 0.8774298056155507,
 'eval_runtime': 8.4326,
 'eval_samples_per_second': 219.624,
 'eval_steps_per_second': 6.878,
 'epoch': 6.0}
```

EWHA EURON

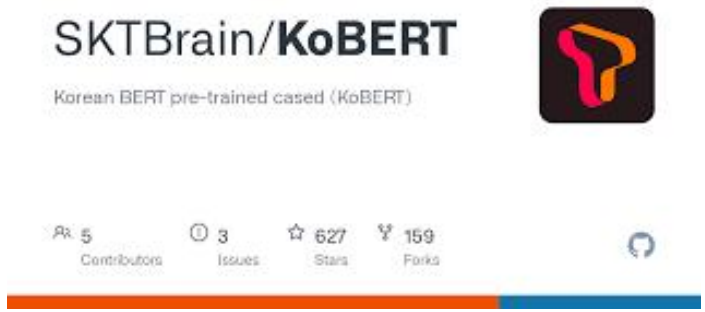
#05 모델링(2) – Deep Learning Model

KOBERT

- 기존 BERT의 한국어 성능을 극복하기 위해 SKT Brain에서 개발한 모델

- 위키피디아나 뉴스 등에서 수집한 수백만개의 한국어 문장의 대규모 말뭉치(Corpus)를 통해 학습됨

- Output Layer를 추가함으로써 언어 특화 모델을 customize할 수 있다는 장점 존재



#05 모델링(2) – Deep Learning Model

KOBERT

(1) Bert Classifier Architecture

```
class BERTClassifier(nn.Module):
    def __init__(self,
                 bert, # bertmodel 입력받음.
                 dr_rate,
                 hidden_size = 768,
                 num_classes=2,   ##클래스 수 조정##
                 params=None):
        super(BERTClassifier, self).__init__()
        self.bert = bert
        self.dr_rate = dr_rate

        self.classifier = nn.Linear(hidden_size , num_classes)
        if dr_rate:
            self.dropout = nn.Dropout(p=dr_rate)

    def gen_attention_mask(self, token_ids, valid_length):
        attention_mask = torch.zeros_like(token_ids)
        for i, v in enumerate(valid_length):
            attention_mask[i][:v] = 1
        return attention_mask.float()

    def forward(self, token_ids, valid_length, segment_ids):
        attention_mask = self.gen_attention_mask(token_ids, valid_length)

        _, pooler = self.bert(input_ids = token_ids, token_type_ids = segment_ids.long(),
                              attention_mask = attention_mask.float().to(token_ids.device),return_dict=False)
        if self.dr_rate:
            out = self.dropout(pooler)
        return self.classifier(out)
```

본 프로젝트의 목적은 Binary classification에 특화된 모델을 만드는 것이므로,
클래스 수는 2로 조정

#05 모델링(2) – Deep Learning Model

KOBERT

(2) get_cosine_schedule_with_warmup 을 활용한 학습률 스케줄러 사용

-학습 초기 단계에서 학습률을 점진적으로 증가시키는 워밍업 단계(warmup steps)를 적용한 후, 코사인 함수 형태로 학습률을 점차적으로 감소시키는 방식

```
def ret_scheduler(train_dataloader, optimizer, epochs):
    # Total number of training steps is [number of batches] x [number of epochs].
    # (Note that this is not the same as the number of training samples).
    total_steps = len(train_dataloader) * epochs
    warmup_step = int(total_steps * warmup_ratio)

    # Create the learning rate scheduler.
    scheduler = get_cosine_schedule_with_warmup(optimizer,
                                                num_warmup_steps = warmup_step,
                                                num_training_steps = total_steps)
    return scheduler
```

Train dataset의 10%를 워밍업의 단계(학습률이 점진적으로 증가하는 구간)로 활용

EWHA EURON

#05 모델링(2) – Deep Learning Model

KOBERT

(3) Layer-wise Learning Rate Decay (LLRD) 적용

-모델의 각 레이어마다 다른 학습률을 적용하는 기법

-상위 레이어는 보다 구체적인 의미 정보/ 태스크 특화 정보를 담고 있음

-하위 레이어는 언어의 기본적인 문법, 어휘, 구조적 정보 등을 담고 있음

-따라서, 상위 레이어에 높은 학습률을 적용 -> 하위 레이어일수록 낮은 학습률 적용

```
def ret_optim_llrd(model, lr_rate):
    # LLRD 적용
    optimizer_grouped_parameters = [
        {'params': model.bert.encoder.layer[:6].parameters(), 'lr': lr_rate * 0.1},
        {'params': model.bert.encoder.layer[6:12].parameters(), 'lr': lr_rate * 0.5},
        {'params': model.classifier.parameters(), 'lr': lr_rate}
    ]

    optimizer = AdamW(optimizer_grouped_parameters, eps=1e-8)
    return optimizer
```

#05 모델링(2) – Deep Learning Model

KOBERT

(4) 최종 모델

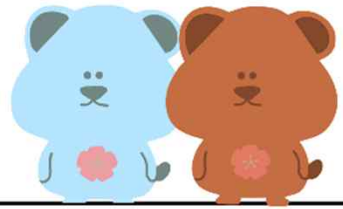
<LLRD 적용 X >

```
evaluate_model_on_test_data(model_plain, test_dataloader, device=torch.device("cuda:0"))

Running Test Evaluation...
  Test Accuracy: 0.8976
  Test Loss: 0.4420
  Test Evaluation took: 0:00:05
{'Test Loss': 0.44201750804980594,
 'Test Accuracy': 0.8976102941176471,
 'Test Time': '0:00:05'}
```

<LLRD 적용 O >

```
evaluate_model_on_test_data(model_llrd, test_dataloader, device=torch.device("cuda:0"))

Running Test Evaluation...
  Test Accuracy: 0.8982
  Test Loss: 0.5367
  Test Evaluation took: 0:00:06
{'Test Loss': 0.5367137044668198,
 'Test Accuracy': 0.8981617647058824,
 'Test Time': '0:00:06'}
```

THANK YOU



EWHA EURON