



K-Learning AI 솔루션

Early Bird 팀

김예지 김희진 류소리 최지원 허진욱

Early Bird 조직도



최지원(팀장)



김예지(팀원)



김희진(팀원)



류소리(팀원)



허진욱(팀원)

목차



Skill set



서비스 기획



서비스 기획

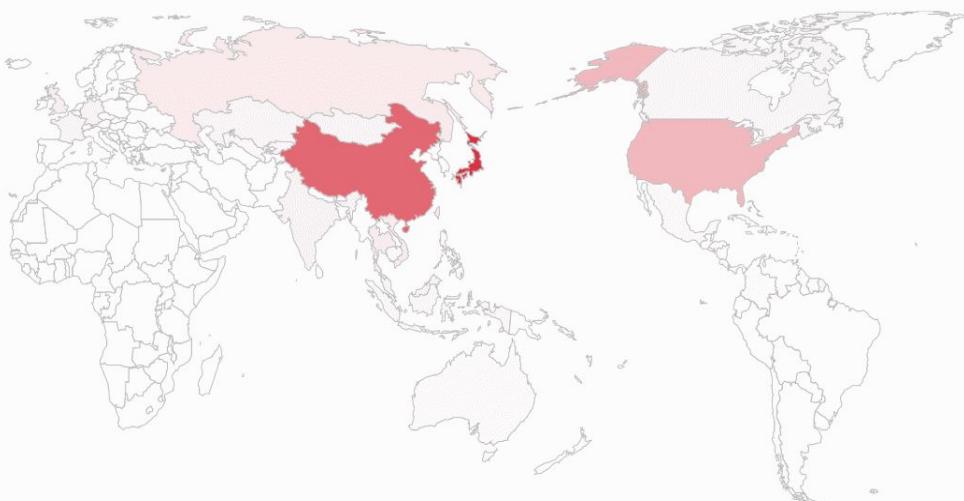
세계 한국학현황 지도

한국학 강좌운영 해외대학 총 : 107개국 1408개처

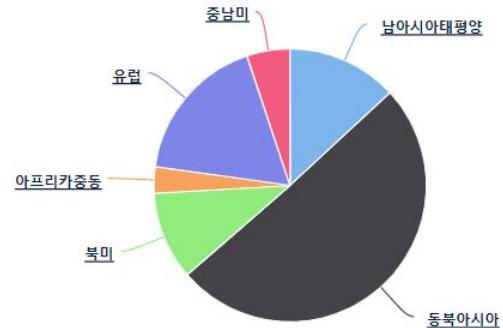


세계 한국학현황 지도 이용 안내

- 세계 한국학현황 지도의 대학 개수 및 정보는 한국국제교류재단에서 업데이트 하는 자료가 실시간으로 반영되고 있습니다.
- 전체 세계지도에서 한국학 강의 제공 대학 수가 많은 국가일수록 색도가 짙게 표시됩니다.
- 국가 위치에 마우스를 올려두시면 국가색이 하늘색으로 바뀌며 좌측 하단에 국가별 한국학 보유 대학 개수가 나타납니다.
- 국가를 클릭하시면 해당 국가의 지도가 확대되어, 대학별 핀포인트가 표시됩니다.
- 핀포인트에 마우스를 올려두시면 간략한 대학 정보가 표시되며, 클릭하시면 해당 대학의 한국학 현황 정보 팝업이 나타납니다.
- 마우스휠을 통한 지도 확대 및 축소가 가능하며, 드래그하여 위치를 이동하실 수 있습니다.
- 국가별 지도 화면에서 좌측 상단의 국가분 아이콘을 클릭하시면 세계전도 화면으로 이동합니다.



전체



서비스 기획



(서울=연합뉴스) 문다영 기자 = "현재 오프라인 한국어 교육으로는 외국인 학습자의 폭발적 증기를 감당하기 어려운 상황입니다."

'온라인 세종학당' 공동연구원이자 첫 사업책임자를 맡았던 김지형 경희사이버대 한국어문화학과 교수는 11일 연합뉴스 인터뷰에서 이렇게 말했다.

일반적인 학습 방법 : 전 세계에서 세종학당을 통해 한국어와 한국 문화 공부

→ 꾸준히 증가하여 15년간 약 **110배 수강생 증가** & 30만명이 넘는 외국인 한국어 능력 시험에 응시

문제 제기 : 폭발적으로 늘어나는 한국어 수요 감당하기 어려움

→ 교원 공급 등 수요를 따라가지 못하고 있는 상황

서비스 기획

키워드

뉴스, K-문화, 한국어 공부

대상

한국어를 공부하는 외국인

설명

외국인들을 위한 한국어 단계별 학습 AI 솔루션

한국어 실력을 **Level 1 ~ Level 3** 으로 단계를 나눔.

학습자가 공부에 흥미를 잃지 않고, 한국어 실력이 증진될 수 있도록 솔루션을 제시.

서비스 기획

Level 1

k-문화 인기 상승

→ k-pop 가사를 활용하여 공부

Level 2

일상 대화 공부는 필수

→ 일상 대화 형식을 공부

Level 3

뉴스를 이용한 스마트 학습 거의 無

→ 뉴스 내용으로 공부하여 심화 공부



서비스 기획

문제 정의

- ❑ 프로그램
- ❑ 시나리오 구성

데이터셋 구축

- ❑ 문장유형 분류
- ❑ 데이터 탐색
- ❑ 클래스 정의
- ❑ 데이터 전처리

모델 개발

- ❑ 불용어 정리
- ❑ 데이터 증강

서비스 구현

- ❑ HTML
- ❑ Flask
- ❑ Bootstrap

데이터 탐색 및 전처리



DACON

문장 유형 분류 AI 경진대회 데이터



수많은 글들을 AI 모델을 활용해 학습하고, 빠르게 분류할 수 있다면

우리는 더 정교하게 분류된 정보를 얻고,

이를 통해 언어가 쓰이는 모든 영역에서

보다 사용자 친화적인 서비스를 경험할 수 있게 될 것을 기대



유형

사실형
추론형
대화형
예측형



시제

과거
현재
미래



극성

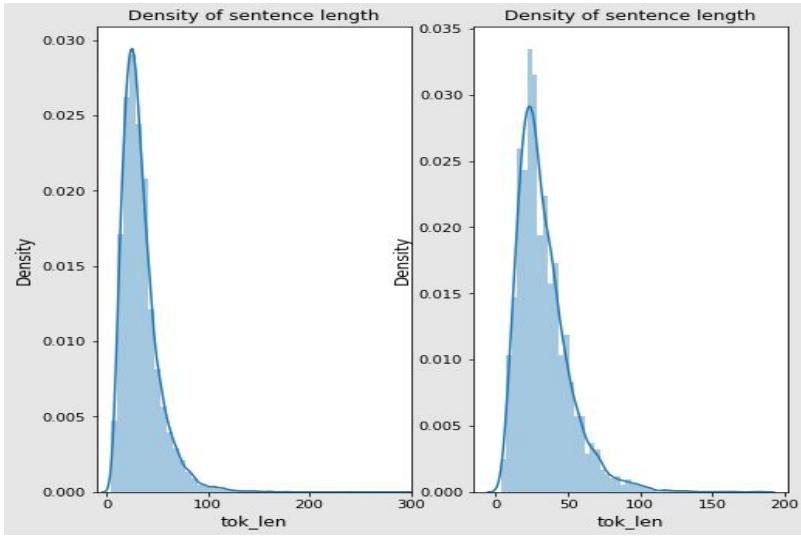
긍정
부정
미정



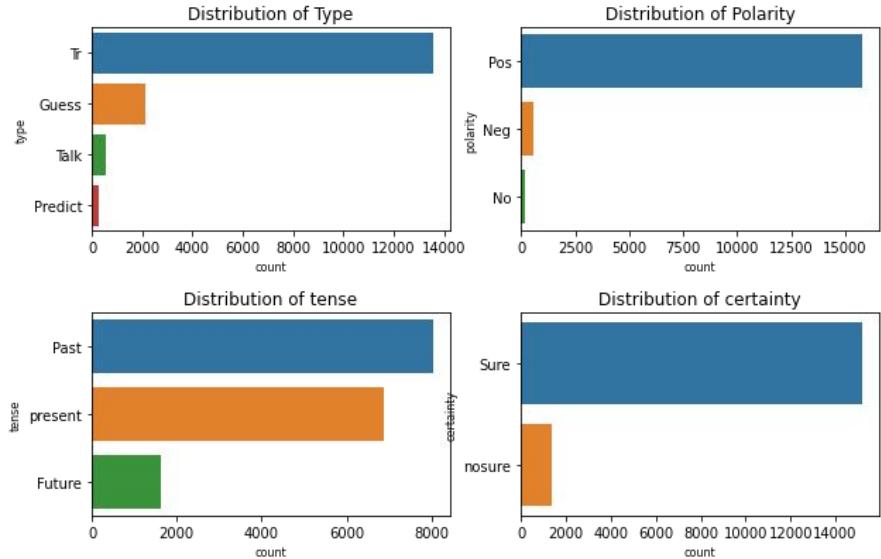
확실성

확실
불확실

데이터 탐색



90% 이상의 데이터들의
문장의 길이가 100을 넘지 않음



라벨 시각화 → 심한 불균형
결측치 x 맞춤법 신경 x

데이터 탐색

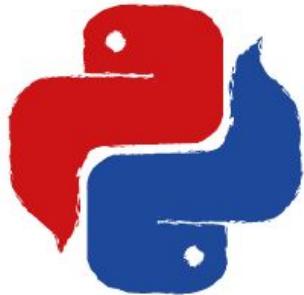


문제점 → 어떤 기준으로 이러한 카테고리들이 붙게 되었는지 **의아한 부분**이 많음.

어미 분석

시제 레이블을 제외한 나머지 레이블에서는 구분이 어려운 부분이 있음.

Kiwi 분석



Soynlp



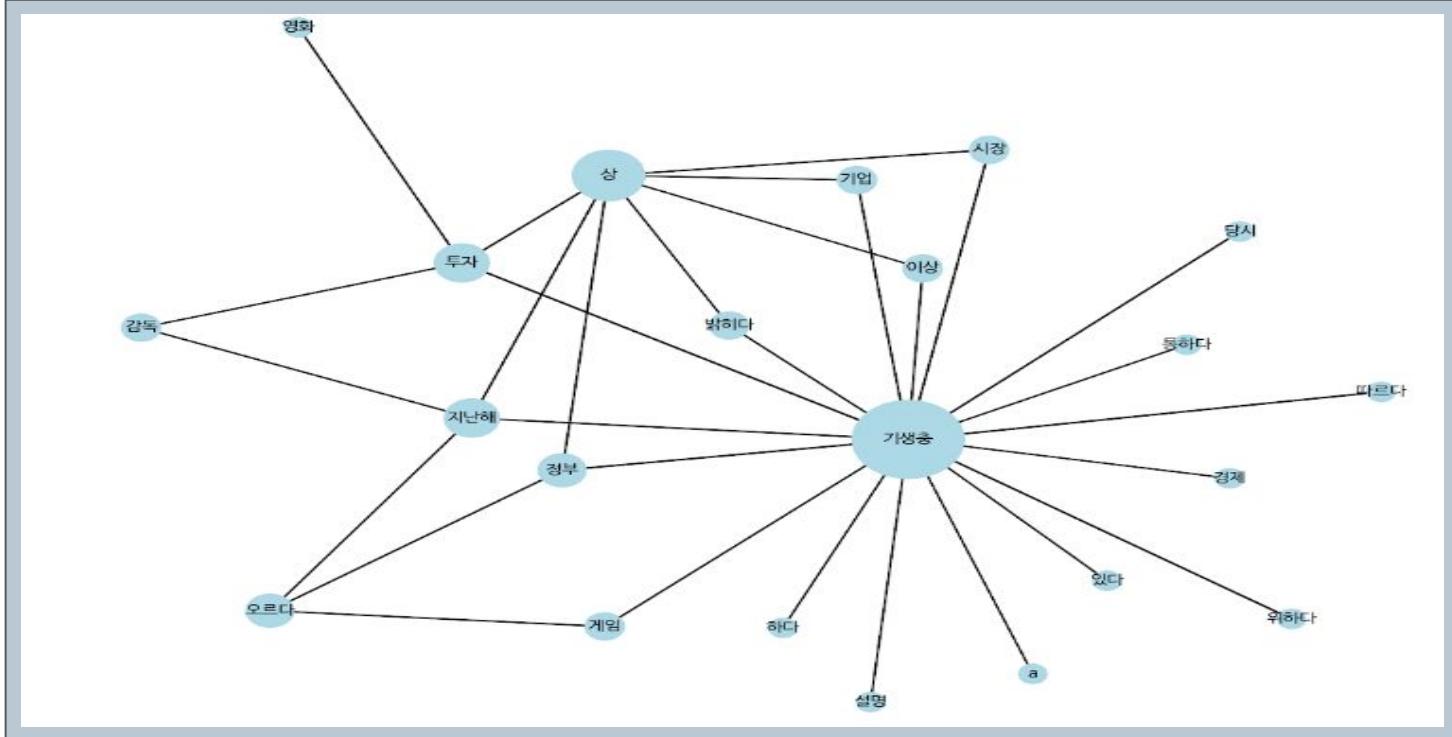
키위
/NNP

KoNLPy

한국어 형태소 분석기

사용자 사전과
형태소분석 없이 토큰화함.

동시출현 네트워크



영화, 한국, 미국, 중국 등 키워드 → 데이터 내용이 편중되어 있음.

하지만 주관적인 판단 → 데이터 탐색에 논리성이 부족함.

논문 + LDA 분석

DBpiA

2021년 한국컴퓨터종합학술대회 논문집

텍스트 전처리를 위한 불용어 목록의 구축

황인서¹⁾, 변석우²⁾, 우근¹⁾

부산대학교 정보컴퓨터공학과

²⁾경성대학교 소프트웨어학과

inseowang@pusan.ac.kr, swbyun@ks.ac.kr, woogyun@pusan.ac.kr

Building a List of Stopwords for Text Preprocessing

Inseow Wang¹⁾, Seokwoo Byun²⁾, Gyun Woo¹⁾

¹⁾School of Information Computer Engineering Pusan National University,

²⁾School of Software Engineering Kyungseong University

요약

한국어 자연어 처리에서 표준화된 불용어 목록은 미비한 상태다. 본격적인 자연어 처리를 위해 텍스트 데이터를 전처리하는 단계는 필수적인데, 유의미한 토큰만을 남겨두기 위해 불용어의 제거는 중요하다. 본 논문에서는 특정 도메인의 텍스트 데이터 집합에서 불용어 목록을 구축하여 단어 집합을 생성하고, 구축한 불용어 목록을 사용하여 세 개의 다른 도메인에 존재하는 텍스트 데이터 집합의 불용어를 정제한다. 구축한 불용어 목록을 네이버 영화, 새종교 포스, 청연대 국민청원 데이터 집합에 적용한 결과, 각각 88.50%, 84.65%, 77.34%의 정제률을 보였다.



LDA란, 문서의 집합으로부터 어떤 토픽이 존재하는지 알아내기 위한 알고리즘

LDA → 각 원들의 거리로 토픽의 유사도를 살펴 볼 수 있음.

레이블 별 정의 정리

클래스별 정의					
A1:E1	B	C	D	E	
	사실형	대화형	추론형	예측형	
5	< 어미 > -다 -는다 -ㄴ다 -했다 이다 였다 쳤었다 한다 었다 있다 않았다 한다	< 어미 > ~요(해요) ~죠(하죠 이죠) ~래(그래? 할래?) ~해(생각해?) ~까?(될까? 무엇일까?) ~나?(하나?) ~야 ~입니다 ~합니다 ~습니다	< 어미 > ~될 것입니다 ~것입니다 ~할 수도 없다 ~전혀 ~되다 ~할 수 있다 ~될 것이다 제공하다 제출하였다 제출하였다	< 어미 > ~길다 ~것이다 ~할 수 있다 ~될 것이다 제출된다 제회이다 제인이다 제여진다 전망이다	
6	문장의 유형	< 문맥 - 네트워크 분석 > - 과거 특징이랑 비슷함 (현재랑도 좀 섞인듯?)	< 문맥 - 네트워크 분석 > - 비트코인, 코인, 스윙의 대화가 많았던듯	< 문맥 - 네트워크 분석 > - 이유 따르다 통하다 경우	< 문맥 - 네트워크 분석 > - 보이다 선보이다 - 예정 예상 전망 - 미래 비슷 - 물을이랑 산업쪽으로 언급됨. - 국내 계획 등을
7	< LDA > - 년도를 사실로 분리하는거 같다. - 전문 용어를 사실형으로 분리 - 고유 명사가 보인다. ex) 우리은행, 저축은행, 메이플스토리	< LDA > - 따옴표가 보임 - 숫자가 보임 ex) 870원, 9곳이었습니다. - 어디일까, 살펴보자 이런 대화형식? 으로 끝난다.	< LDA > - 다른 것보다 나타난 수치가 낮다. - 서울형으로 끝나는게 좀 보인다.	< LDA > - 예정이다, 전망이다, 계획이다. 가 확실히 보인다. - 일기예보, 해상에서는, 물풀탄, 미세먼지 - 수치가 보임	
8					

3차례 데이터 탐색 → 비슷한 특징들이 발견 되었음.

사용된 모델과 기술



사용 기술



TTS

텍스트 → 오디오



Crawling

웹페이지에서
데이터 긁어오기



DALL·E 2

인공지능이
자동으로 그림을 생성

사용 모델

분류 모델

KLUE-RoBERTa-small

Masked 모델

klue/bert-base('fill-mask')

STS 모델

Huffon/sentence-klue-roberta-base

T5 문서 요약 모델

eenzeenee/t5-small-korean-summarization

QA 모델

multi-qa-MiniLM-L6-cos-v1

질문 생성 모델

Sehong/kobert-Question Generation

MRC 모델

bespin-global/klue-bert-base-aihub-mrc

Klue 란

- 한국어 언어 모델의 공정한 평가를 위한 목적으로 만들어진 벤치마크
- 8개 종류 데이터셋으로 구성
 - 뉴스 헤드라인 분류(Topic Classification)
 - 문장 유사도 비교(Sentence Textual Similarity)
 - 자연어 추론(Natural Language Inference)
 - 개체명 인식(Named Entity Recognition)
 - 관계 추출(Relation Extraction)
 - 형태소 및 의존 구문 분석(Dependency Parsing)
 - 기계 독해 이해(Machine Reading Comprehension)
 - 대화 상태 추적(Dialog State Tracking)

Klue-RoBERTa model

KLUE-RoBERTa-small

Model	Embedding size	Hidden size	# Layers	# Heads
KLUE-BERT-base	768	768	12	12
KLUE-RoBERTa-small	768	768	6	12
KLUE-RoBERTa-base	768	768	12	12
KLUE-RoBERTa-large	1024	1024	24	16

KLUE-RoBERTa-small

```
import random
import pandas as pd
import numpy as np
import os

from sklearn.model_selection import train_test_split
# from sklearn.feature_extraction.text import TfidfVectorizer # baseline code용 tfidef vectorirzer(또는 tokenizer)
from sklearn import preprocessing
from sklearn.metrics import f1_score

from transformers import AutoModel, AutoTokenizer # 사용하고자 하는 모델, 토크나이저 적용시 필요
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader

from tqdm.auto import tqdm # process bar 표시용

import warnings
warnings.filterwarnings(action='ignore')
```

KLUE-RoBERTa-small



```
pretrained_model = 'klue/roberta-small'

base_model = AutoModel.from_pretrained(pretrained_model)
tokenizer = AutoTokenizer.from_pretrained(pretrained_model)

# 2. Label Encoding (유형, 극성, 시제, 확실성) - 이거는 베이스라인 코드 써도 될듯
type_le = preprocessing.LabelEncoder()
train["유형"] = type_le.fit_transform(train["유형"].values)
val[ "유형" ] = type_le.transform(val[ "유형" ].values)
df[ "유형" ] = type_le.transform(df[ "유형" ].values)

polarity_le = preprocessing.LabelEncoder()
train["극성"] = polarity_le.fit_transform(train[ "극성" ].values)
val[ "극성" ] = polarity_le.transform(val[ "극성" ].values)
df[ "극성" ] = polarity_le.transform(df[ "극성" ].values)

tense_le = preprocessing.LabelEncoder()
train["시제"] = tense_le.fit_transform(train[ "시제" ].values)
val[ "시제" ] = tense_le.transform(val[ "시제" ].values)
df[ "시제" ] = tense_le.transform(df[ "시제" ].values)

certainty_le = preprocessing.LabelEncoder()
train["확실성"] = certainty_le.fit_transform(train[ "확실성" ].values)
val[ "확실성" ] = certainty_le.transform(val[ "확실성" ].values)
df[ "확실성" ] = certainty_le.transform(df[ "확실성" ].values)
```

KLUE-RoBERTa-small



```
class CustomDataset(Dataset):
    def __init__(self, tokenized_text, labels= None):
        self.texts = tokenized_text
        self.labels = labels

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, index):
        text = self.texts[index]

        if self.labels is not None:
            st_type = self.labels['type'][index]
            st_polarity = self.labels['polarity'][index]
            st_tense = self.labels['tense'][index]
            st_certainty = self.labels['certainty'][index]

            return text, st_type, st_polarity, st_tense, st_certainty
        else:
            return text, torch.Tensor([-1,-1,-1,-1]), torch.Tensor([-1,-1,-1]), torch.Tensor([-1,-1,-1]), torch.Tensor([-1,-1])

train_dataset = CustomDataset(train_tokenized_texts, train_labels)
train_loader = DataLoader(train_dataset, batch_size = CFG['BATCH_SIZE'], shuffle=True, num_workers=0)

val_dataset = CustomDataset(val_tokenized_texts, val_labels)
val_loader = DataLoader(val_dataset, batch_size = CFG['BATCH_SIZE'], shuffle=False, num_workers=0)
```

KLUE-RoBERTa-small



```
class CustomModel(nn.Module):
    def __init__(self):
        super(CustomModel, self).__init__()
        self.base_model = AutoModel.from_pretrained(pretrained_model).to(device)
        try:
            self.out = self.base_model.encoder.layer[-1].output.dense.out_features
        except:
            self.out = 768

        self.type_classifier = nn.Sequential(
            nn.Dropout(p=0.2),
            nn.Linear(in_features=self.out, out_features=4),      # 유형 : 문장의 유형 (사실형, 추론형, 대화형, 예측형)
            # nn.Softmax(dim=1)
        )
        self.polarity_classifier = nn.Sequential(
            nn.Dropout(p=0.2),
            nn.Linear(in_features=self.out, out_features=3),      # 극성 : 문장의 극성 (긍정, 부정, 미정)
            # nn.Softmax(dim=1)
        )
        self.tense_classifier = nn.Sequential(
            nn.Dropout(p=0.2),
            nn.Linear(in_features=self.out, out_features=3),      # 시제 : 문장의 시제 (과거, 현재, 미래)
            # nn.Softmax(dim=1)
        )
        self.certainty_classifier = nn.Sequential(
            nn.Dropout(p=0.2),
            nn.Linear(in_features=self.out, out_features=2),      # 확실성 : 문장의 확실성 (확실, 불확실)
            # nn.Softmax(dim=1)
        )

    def forward(self, input_ids, attention_mask, labels=None, token_type_ids=None):
        x = self.base_model(input_ids=input_ids, attention_mask=attention_mask)[0]

        # x = self.linear(x)
        # 문장 유형, 극성, 시제, 확실성을 각각 분류
        type_output = self.type_classifier(x[:,0,:].view(-1,self.out))
        polarity_output = self.polarity_classifier(x[:,0,:].view(-1,self.out))
        tense_output = self.tense_classifier(x[:,0,:].view(-1,self.out))
        certainty_output = self.certainty_classifier(x[:,0,:].view(-1,self.out))

        return type_output, polarity_output, tense_output, certainty_output
```

KLUE-RoBERTa-small

```
def train(model, optimizer, train_loader, val_loader, scheduler, device):
    model.to(device)

    criterion = {
        'type' : nn.CrossEntropyLoss().to(device),
        'polarity' : nn.CrossEntropyLoss().to(device),
        'tense' : nn.CrossEntropyLoss().to(device),
        'certainty' : nn.CrossEntropyLoss().to(device)
    }

    best_loss = np.inf
    best_model = None

    for epoch in range(1, CFG['EPOCHS']+1):
        model.train()
        train_loss = []
        for sentence, type_label, polarity_label, tense_label, certainty_label in tqdm(iter(train_loader)): # 각각의 train data iteration
            # sentence = sentence.to(device)

            # attention_mask = train_input['attention_mask'].to(device)
            # input_ids = train_input['input_ids'].squeeze(1).to(device)

            input_ids = sentence.input_ids.squeeze(1).to(device)
            attention_mask = sentence.attention_mask.to(device)

            type_label = type_label.to(device)
            polarity_label = polarity_label.to(device)
            tense_label = tense_label.to(device)
            certainty_label = certainty_label.to(device)
            optimizer.zero_grad() # 초기화

            type_logit, polarity_logit, tense_logit, certainty_logit = model(input_ids,attention_mask) # 모델 적용

            loss = criterion['type'](type_logit, type_label) + \
                criterion['polarity'](polarity_logit, polarity_label) + \
                criterion['tense'](tense_logit, tense_label) + \
                criterion['certainty'](certainty_logit, certainty_label)

            loss.backward() # 역전파

            optimizer.step() # 한단계 진행

            train_loss.append(loss.item()) # loss 합산

        val_loss, val_type_f1, val_polarity_f1, val_tense_f1, val_certainty_f1, f1_mult = validation(model, val_loader, criterion, device)

        print(f'Epoch : {epoch} F1 : {(f1_mult:.5f)} Train Loss : {(np.mean(train_loss):.5f)} Val Loss : {(val_loss:.5f)} \
        유형 F1 : {(val_type_f1:.5f)} 맥정 F1 : {(val_polarity_f1:.5f)} 시제 F1 : {(val_tense_f1:.5f)} 확실성 F1 : {(val_certainty_f1:.5f)}')

        if scheduler is not None:
            scheduler.step(val_loss)

        if best_loss > val_loss:
            best_loss = val_loss
            best_model = model

    return best_model
```



KLUE-RoBERTa-small

```
import gc
# del model
gc.collect()
torch.cuda.empty_cache() # gpu 지원관리

model = CustomModel()
model.to(device)

model.eval()
optimizer = torch.optim.Adam(params = model.parameters(), lr = CFG["LEARNING_RATE"])
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', factor=0.5, patience=2, threshold_mode='abs', min_lr=1e-8, verbose=True)

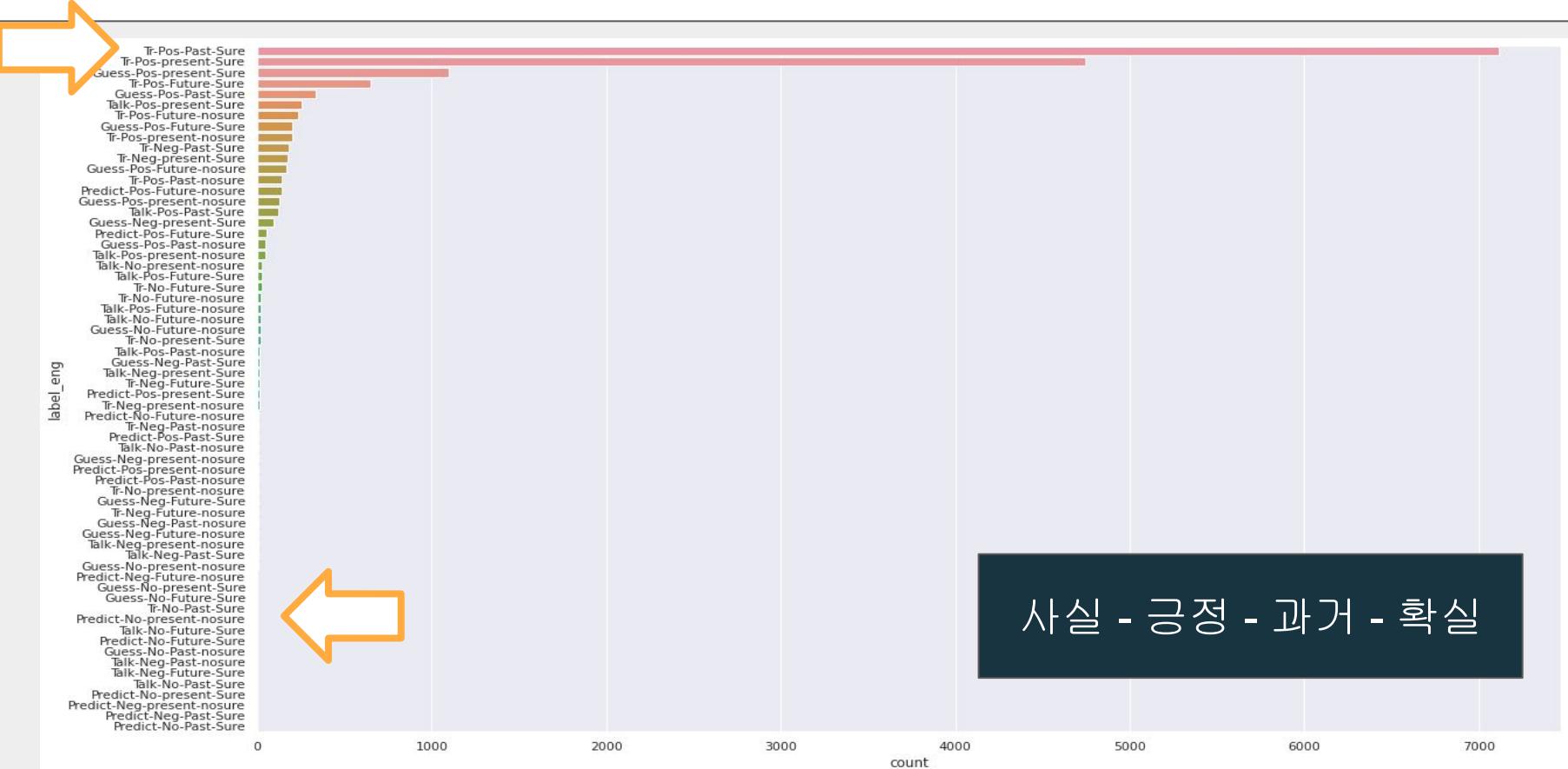
infer_model = train(model, optimizer, train_loader, val_loader, scheduler, device)
```

100%  1618/1618 [12:29<00:00, 2.20it/s]

100%  405/405 [01:02<00:00, 6.54it/s]

Epoch : [10] F1 : [0.83162] Train Loss : [0.00002] Val Loss : [0.00006] 유형 F1 : [0.92850] 긍성 F1 : [0.99846] 시제 F1 : [0.93579] 확실성 F1 : [0.95859]
Epoch 00010: reducing learning rate of group 0 to 2.5000e-06.

모델 정확도 향상 - Augmentation



모델 정확도 향상 - Augmentation

```
# https://github.com/catSirup/KorEDA/blob/master/eda.py
def swap_word(new_words):
    random_idx_1 = random.randint(0, len(new_words)-1)
    random_idx_2 = random_idx_1
    counter = 0

    while random_idx_2 == random_idx_1:
        random_idx_2 = random.randint(0, len(new_words)-1)
        counter += 1
        if counter > 3:
            return new_words

    new_words[random_idx_1], new_words[random_idx_2] = new_words[random_idx_2], new_words[random_idx_1]
    return new_words

def random_swap(words, n):
    new_words = words.copy()
    for _ in range(n):
        new_words = swap_word(new_words)
    return new_words

def text_aug(sentence, alpha_rs = 0.1, num_aug=3):
    words = sentence.split(' ')
    words = [word for word in words if word != ""]
    num_words = len(words)

    augmented_sentences = []
    num_new_per_technique = num_aug

    n_rs = max(1, int(alpha_rs*num_words))

    for _ in range(num_new_per_technique):
        a_words = random_swap(words, n_rs)
        augmented_sentences.append(" ".join(a_words))

    augmented_sentences = [sentence for sentence in augmented_sentences]
    random.shuffle(augmented_sentences)

    if num_aug >= 1:
        augmented_sentences = augmented_sentences[:num_aug]
    else:
        keep_prob = num_aug / len(augmented_sentences)
        augmented_sentences = [s for s in augmented_sentences if random.uniform(0, 1) < keep_prob]
    return augmented_sentences

aug = df['문장'].apply(lambda x: text_aug(x))
```

- ▶ 랜덤 교체 (Random Swap)
- ▶ 랜덤 삭제 (Random Deletion)
- ▶ 불용어 처리

모델 정확도 향상 - KFold & Ensemble

```
# 제공된 학습데이터를 학습 / 검증 데이터셋으로 재 분할

''' stratified수행시 좋아질 수 있음
k-fold로 변형 필요 '''

folds = StratifiedKFold(n_splits=CFG[ 'FOLD' ], random_state=CFG[ 'SEED' ], shuffle=True)

df[ 'kfold' ] = -1
for i in range(5):
    df_idx, valid_idx = list(folds.split(df.values, df[ 'label' ]))[i]
    valid = df.iloc[valid_idx]

    df.loc[df[df.ID.isin(valid.ID) == True].index.to_list(), 'kfold'] = i
```

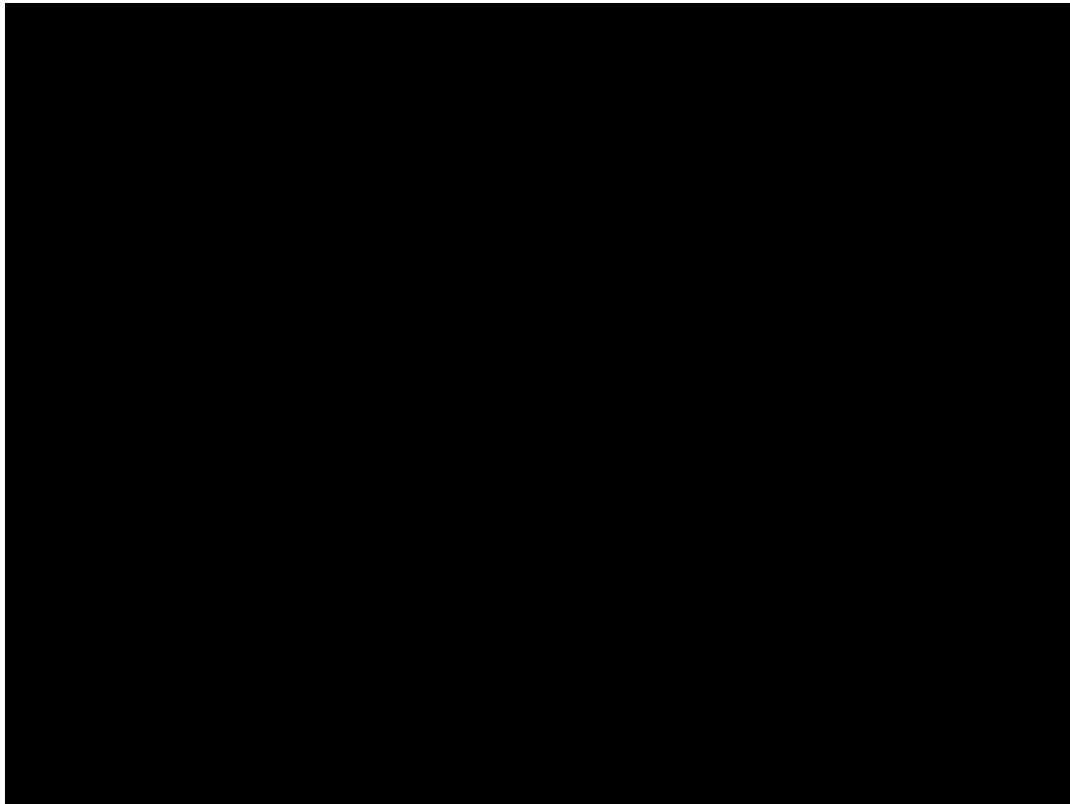
정확도가 향상 ↑

803258	제출용_klue-roberta-base_BatchSize_32_Epoch_10.csv 데이터 증식 有 + epochs 10 + learning rate 1e-5, batch size 32 edit	2023-02-08 14:59:12	0.7193384879 0.7420068224	<input type="checkbox"/>
799376	test.csv klue-Roberta_small_base_model epoch 4 batch size 256 edit	2023-01-27 14:17:17	0.7168554327 0.7256542856	<input checked="" type="checkbox"/>



서비스 제작

시연 영상



Level 1. k-pop 가사 공부

TTS

Masked
Model

DALL·E 2

노래 가사 받아쓰기

- Q1. 다음 들려주는 가사를 듣고
한글로 적어보세요.

빈칸 학습하기

- Q2. 다음 보기 중 빈칸에 들어갈 말로
가장 어울리지 않은 단어를 골라주세요

이미지에 맞는 문장 찾기

- Q3. 다음 이미지와 맞는
문장을 골라주세요.

Level 2. 일상 대화 공부

Classification
Model

STS
Model

QA
Model

문장 유형 선택하기

- Q1. 아래 문장을 보고
유형을 맞춰주세요.

비슷한 문장 선택하기

- Q2. 다음 문장을 읽고 문맥상 가장
뜻이 비슷한 보기를 골라주세요.

로봇이랑 대화하기

- Q3. 위로봇과 일상적인 대화를
나눠보세요.

Level 3. 뉴스 기사 공부

Classification
Model

T5
Model

Classification
Model

질문 생성
&
MRC Model

카테고리 선택하기

Q1. 이 뉴스는 다음 중 어떤 분류에 속할까요?

문장 요약하기

Q2. 본문 문장을 요약해 보세요.

문장 유형 선택하기

Q3. 뉴스 본문 중 하나의 문장이 주어집니다.
문장 유형을 맞춰 보세요.

Q&A

Q4. 질문이 나오면 답변해 보세요.

실제 적용 및 피드백





서비스 기대효과 및 개선점

서비스 기대효과



회화학습

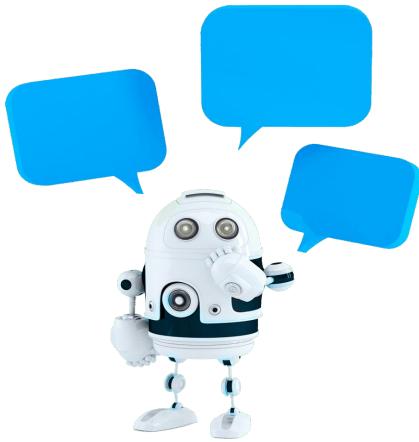


문화 체험



실력 증진

프로젝트 개선점



기술 개선점

- ▷ 모델 정확도 개선
- ▷ 최적화 작업
- ▷ 챗봇 구현



서비스 개선점

- ▷ 다양한 콘텐츠

소감



<지원>

평소에도 자연어에 대한 관심이 많았는데 이번 프로젝트를 통해서 모델 구현, 데이터 전처리에 대한 고민 등등 많은 공부를 할 수 있어서 정말 재밌고 즐거웠습니다.



<희진>

이번 프로젝트를 통해 모델 구축, 여러 모델 활용 방법, 웹 구현 등 여러 기술들을 공부할 수 있는 계기가 되었고, 좋은 팀원들을 만나 프로젝트를 잘 마무리한 것 같습니다.



<예지>

다양한 모델들과 패키지들 공부하고 웹 구현도 도전해볼 수 있어서 뜻깊은 프로젝트였습니다. 감사했습니다~



<소리>

다 같이 모여 모델 공부 할때가 어제 같은데 모델들과 웹까지 많이 배웠네요!
즐거웠습니다. 감사해요.



<진욱>

모델부터 웹 시연까지 전반적으로 한 번 씩 다뤄본것이 좋았어요. 다들 좋은곳에 취직하시고 건강하시길

Citation

<https://dacon.io/competitions/official/236037/overview/description>

<https://klue-benchmark.com/>

<https://huqqingface.co/klue/roberta-small>

<https://huqqingface.co/klue/bert-base>

<https://huggingface.co/Sehong/kobart-QuestionGeneration>

<https://huggingface.co/bespin-global/klue-bert-base-aihub-mrc>

<https://huggingface.co/Huffon/sentence-klue-roberta-base>

<https://huqqingface.co/eenzeenee/t5-small-korean-summarization>

<https://html5up.net/phantom>

<https://www.yna.co.kr/view/AKR20211010052400004>

https://tbs.seoul.kr/news/newsView.do?typ_800=4&idx_800=3479756&seq_800=20472426

https://f.hubspotusercontent40.net/hubfs/364394/CTPE%20Blog%20-%202011_03%20-%20Common%20Obstacles%20to%20Business%20Process%20Improvement%20Index.jpg

KF통계센터

<https://konlpy-ko.readthedocs.io/ko/v0.4.3/>

<https://www.aitimes.com/news/articleView.html?idxno=47005>

<https://www.inflearn.com/course/12%EC%8B%9C%EA%B0%84-%ED%8C%8C%EC%9D%B4%EC%8D%AC-%ED%81%AC%EB%A1%A4%EB%A7%81>

Presentation template by [Slidesgo](#)

Icons by [Flaticon](#)

Images & infographics by [Freepik](#)

Pictures by [Peoplecreations](#) and [Freepik](#)

감사합니다.

QnA
