강서구청장 보궐선거 예측모델

Mayor of Gangseo-gu election prediction model

YouTube

YoonSeok_Choi

목차

주제 선정 이유

한국대학신문 | 2016.11.12.

빅데이터 분석해 '트럼프 당선' 정확히 예측

이러한 표본오차가 결국 잘못된 **선거**결과를 **예측**하게 합니다. 하지만
그렇지 않습니다. 하루에도 수만 명, 수십만 명의 유권자들이 그들도

디지털타임스 | 16면 TOP | 2016.03.09. | 네이버뉴스

빅데이터로 총선결과 **예측** 더 정확해진다

2014년 6월 4일 지방**선거**에서 **빅데이터 예측** 서비스를 실시한 국내
와이즈넛은 예상한 11개 지역 중 7개 지역 당선자를 맞췄다. 회사는

디지털타임스 | 14면 TOP | 2015.12.08. | 네이버뉴스

내년 총선 **예측**? **빅데이터**는 알고 있다

선거 예측 도구로 활용... 성능개선 솔루션 '주목' 정보·모바일 계층 증

앞두고 **선거 예측** 도구로 **빅데이터** 솔루션이 주목받고 있다. **빅데이터** 솔루션은 이전에도 활용돼 왔지만, 수...

아이뉴스24 | 2015.11.19. | 네이버뉴스

SNS와 만난 **선거**, 초유의 **빅데이터** 전쟁 예고

SNS **빅데이터** 분석은 특히 **선거**판세 **예측**에서 큰 영향력을 발휘했다. 전국 단위
선거들 가운데 대표적인 경우가 지난해 6·4 지방**선거**다. **빅데이터** 분석 업체 스토...

디지털타임스 | 10면 3단 | 2014.05.26. | 네이버뉴스

지방**선거** 판세 `**빅데이터**`로 본다

이미 해외에서는 **빅데이터**가 **선거 예측** 도구로 활용되고 있으며, **빅데이터**를 어떻게 활용하는지가 **선거**의 당락
을 결정할 정도로 중요해지고 있다. 지난 2012년 미국 대통령 **선거**에서 오바마 대통령 **선거**캠프는 유권자 분...

- **미국 44대 대통령 버락 오바마 선거 전략을 빅데이터에 기반 수립.**

- **2023 강서구청장 보궐선거 국내 대다수가 사용하는 유튜브 데이터를 기반으로 예측하고자 함.**

KBS PiCK | 2023.05.18. | 네이버뉴스

'청와대 감찰무마 폭로' **김태우** 유죄 **확정**...**구청장**직 상실

그대로 **확정**했습니다. 김 **구청장**은 대법원 선고 뒤 입장문을 통해 "정치적 재판에
의해 제가 잠시 강서**구청장**직에서 물러나더라도, 진실은 왜곡될 수 없다"며 "저...

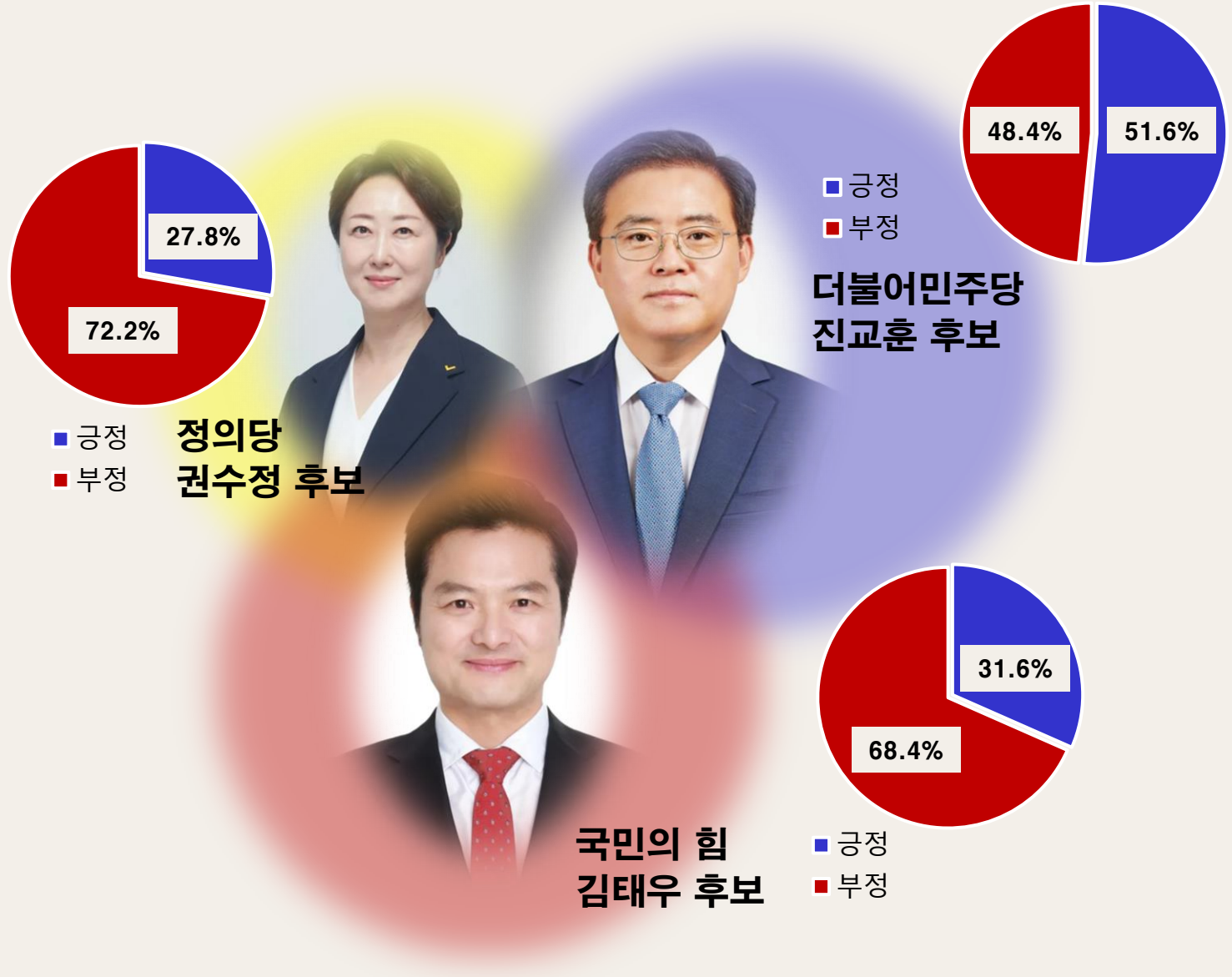
인천일보 | 2023.05.18.

"조국 감찰 무마" **김태우**, 징역형 **확정**...**구청장**직 상실

기소된 **김태우** 서울 강서**구청장**에게 징역형의 집행유예가 **확정**됨에 따
서울 **구청장** 중 첫 번째 공백이 발생했다. 이에 따라 10월 보궐선거...

'청와대 감찰무마 폭로' **김태우**, **구청장**직 상실... "조국... 뉴데일리
수사관→조국 저격수→**구청장**→당선무효..... 뉴스1 | 2023.05.18.
"조국이 유죄면 나는 무죄다"...**구**... 헤럴드경제 PiCK | 2023.05.18.
'靑 감찰무마 폭로' **김태우** 징역형 **확정**..."구정 차질 우... 청년일보

프로젝트 결과



YouTube

영상 수 + 댓글 수

댓글 내용의 긍정/부정

더불어민주당
진교훈 후보의 당선 예측

데이터 소스

projectCHOI 유튜브 크롤링

Code Blame 106 lines (82 loc) · 4 KB Code 55% faster with Gi

```python
# 유튜브 크롤링
#Youtube API Key를 받아야 함.
#Youtube API Key : ##Key##

#pip 유튜브 키를 사용해 크롤링 모듈을 가져온다.

import os #운영 체제 관련 작업을 시작한다.

#google의 API 사용을 위한 코드,
import google_auth_oauthlib.flow # API OAuth 인증
import googleapiclient.discovery # API 서비스 호출
import googleapiclient.errors    # API 오류 처리

import pandas as pd #데이터 처리

import time #하나의 작업이 끝나면 10초의 딜레이를 건다. 아래에 ti

#작업 시작~유튜브를 불러 오는 것.
# YouTube API 키를 설정
api_service_name = "youtube"
api_version = "v3"
api_key = "##Key##"  # 여기에 API 키를 넣자.

# YouTube API 클라이언트를 생성
youtube = googleapiclient.discovery.build(
    api_service_name, api_version, developerKey=api_key)

#크롤링1 하나의 검색어
```

```python

    # YouTube 동영상 검색을 위한 요청을 만듭니다
search_request = youtube.search().list(
        q=search_query,
        type="video", #type을 비디오만
        part="id", #part는 비디오 안에 id만
        maxResults=300 #maxResults 이건 동영상의 갯수
    )

    # API 요청을 실행하고 응답을 받습니다
search_response = search_request.execute()

    #여기서 부터는 for문 사용
    # 검색 결과를 데이터프레임으로 변환합니다
video_data = []
for item in search_response["items"]:
    video_id = item["id"]["videoId"]

    #동영상 정보 요청
    #https://www.youtube.com/watch?v=9zc57ArYET4
    #https://www.youtube.com/watch?v=는 비디오 [뒤에 코드] 비디오 아이디
    video_request = youtube.videos().list(
        part="snippet,statistics",
        id=video_id
    )

    # API 요청을 실행하고 응답을 받습니다
    try:
        comments_response = comments_request.execute()
    except googleapiclient.errors.HttpError as e:
        error_content = e.content.decode("utf-8")
        if e.resp.status == 403 and 'commentsDisabled' in error_content:
            # 해당 동영상은 댓글이 비활성화되어 있음
            print(f"댓글이 비활성화된 동영상: {video_id}")
            continue  # 다음 동영상 처리로 넘어갑니다
        else:
            # 다른 오류 처리
            raise e


    # 가져온 댓글 정보를 리스트에 저장합니다
```

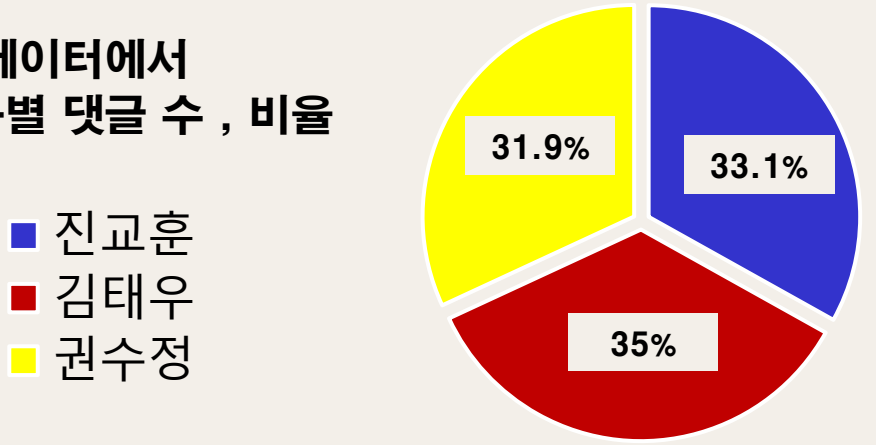
- 데이터 소스

Youtube API, GoogleAPIclient 를 활용

- Crawling

동영상 제목, 게시일, 영상 좋아요 수, 댓글, 작성자, 댓글 작성일, 댓글 좋아요 수

전체 데이터에서
후보자별 댓글 수 , 비율



후보자	동영상 수	댓글 수	비율
진교훈	49개	918개	33.1%
김태우	50개	986개	35%
권수정	47개	712개	31.9%

데이터 전처리 및 EDA

projectCHOI 전처리 과정

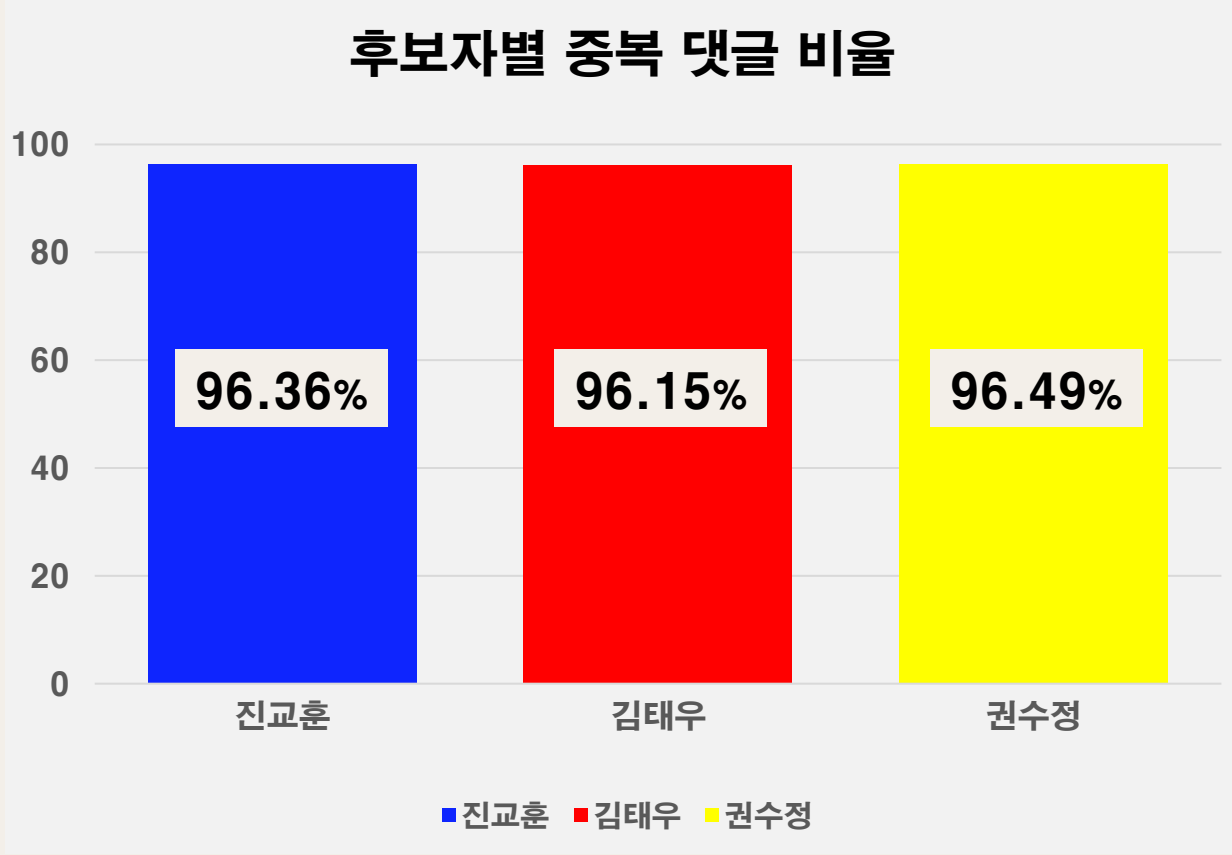
Code Blame 123 lines (95 loc) · 4.2 KB Code 55% faster with GitHub Copilot Raw

```python
import pandas as pd

# 후보자 목록
candis = ['진교훈', '김태우', '권수정']

# 각 후보자의 결과 데이터 CSV 파일 읽기 및 기술적 분석 출력
for candi in candis:
    # CSV 파일 읽기
    file_path = f"/content/drive/MyDrive/SeSAC_AI 전문인력 양성/Project/2023년 강서구청장 보궐선거 예
    df = pd.read_csv(file_path)

    # 기술적 통계 출력
    print(df.describe())
    print()

# 각 후보자의 엑셀 파일 읽기 및 CSV로 변환
for candi in candis:
    # 엑셀 파일 읽기
    excel_file_path = f"/content/drive/MyDrive/SeSAC_AI 전문인력 양성/Project/Raw Data/comment_{candi
    excel_data = pd.read_excel(excel_file_path)

    # CSV 파일로 저장
    csv_file_path = f"/content/drive/MyDrive/SeSAC_AI 전문인력 양성/Project/Raw Data/comment_{candi}.
    excel_data.to_csv(csv_file_path, index=False)

# 변환 완료 메시지
print('성공적으로 변환되었습니다.')

# 각 후보자별 댓글 데이터 CSV 파일 읽기
df_jin = pd.read_csv("/content/drive/MyDrive/SeSAC_AI 전문인력 양성/Project/Raw Data/comment_진교훈.c
df_kim = pd.read_csv("/content/drive/MyDrive/SeSAC_AI 전문인력 양성/Project/Raw Data/comment_김태우.c
df_kwon = pd.read_csv("/content/drive/MyDrive/SeSAC_AI 전문인력 양성/Project/Raw Data/comment_권수정.

# 기술적 분석 출력
print(df_jin.describe())
print(df_kim.describe())
print(df_kwon.describe())
```

```python

# 기술적 분석 재출력
print(df_jin.describe())
print(df_kim.describe())
print(df_kwon.describe())

# Pandas 출력 옵션 설정
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

# 'comment_length' 열을 기준으로 정렬하여 상위 10개 행 출력
sorted_df_jin = df_jin.nlargest(80, 'comment_length', keep='all')
print(sorted_df_jin)

# 특정 값 선택
selected_value = df_jin.loc[[11782, 1736, 862], 'comment'] # 특정 인덱스의 'comment' 열 값 선택
print(selected_value)

# 중복된 행의 개수 계산 및 출력
duplicate_rows_jin = df_jin['comment'].duplicated().sum()
duplicate_rows_kim = df_kim['comment'].duplicated().sum()
duplicate_rows_kwon = df_kwon['comment'].duplicated().sum()

print(f'jin 중복된 행의 개수: {duplicate_rows_jin}')
print(f'kim 중복된 행의 개수: {duplicate_rows_kim}')
print(f'kwon 중복된 행의 개수: {duplicate_rows_kwon}')

import pandas as pd

# 후보자별 결과 데이터 CSV 파일 읽기
df_jin = pd.read_csv("/content/drive/MyDrive/SeSAC_AI 전문인력 양성/Project/2023년 강서구청장 보궐선거 예측분석/Raw Data/results_진교훈.csv")
df_kim = pd.read_csv("/content/drive/MyDrive/SeSAC_AI 전문인력 양성/Project/2023년 강서구청장 보궐선거 예측분석/Raw Data/results_김태우.csv")
df_kwon = pd.read_csv("/content/drive/MyDrive/SeSAC_AI 전문인력 양성/Project/2023년 강서구청장 보궐선거 예측분석/Raw Data/results_권수정.csv")

# 각 데이터프레임의 기술적 분석 출력
print("진교훈")
print(df_jin.describe())
print()

print("김태우")
print(df_kim.describe())
print()
```

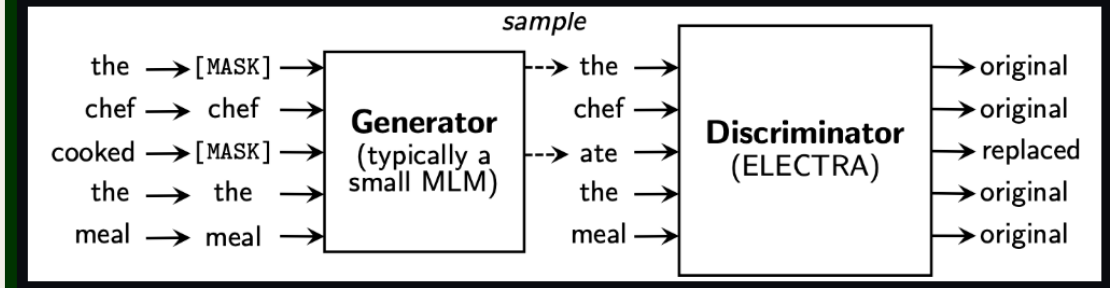
- **중복 댓글 제거**

동일 작성자의 동일 댓글을 제거,
분석의 신뢰도를 높이고자 함,



모델 선택 및 EDA

KoELECTRA



ELECTRA는 `Replaced Token Detection`, 즉 generator에서 나온 token을 보고 discriminator에서 "real" token인지 "fake" token인지 판별하는 방법으로 학습을 합니다. 이 방법은 모든 input token에 대해 학습할 수 있다는 장점을 가지며, BERT 등과 비교했을 때 더 좋은 성능을 보였습니다.

KoELECTRA는 **34GB의 한국어 text**로 학습하였고, 이를 통해 나온 `KoELECTRA-Base`와 `KoELECTRA-Small` 두 가지 모델을 배포하게 되었습니다.

또한 KoELECTRA는 **Wordpiece 사용, 모델 s3 업로드** 등을 통해 OS 상관없이 `Transformers` 라이브러리만 설치하면 곧바로 사용할 수 있습니다.

- **모델**
  HuggingFace KoElectra-small을 이용

- **모델특징**
  사람이 선거를 바라보는 것이 영화 보듯이 댓글을 작성한다는 가정하에 예측모델을 사용

- **데이터셋 구조**
  15만개 Train Data와 5만개의 Test Data로 구성
  일부 Data NaN, 중복 데이터 존재

- **전처리**
  HuggingFace에서 제공하는 AutoTokenizer 이용

project_3_강서구시장 선거예측모델 / projecr3_VS_ 딥러닝 기반 NLP 모델.py

암호 탓하다

```python
# 필요한 라이브러리 임포트 설치
import pandas as pd
import torch
from torch.nn import functional as F
from torch.utils.data import DataLoader, Dataset
from transformers import AutoTokenizer, ElectraForSequenceClassification, AdamW
from tqdm.notebook import tqdm

# GPU 사용 설정
device = torch.device("cuda")

# NSMC 데이터셋을 처리하기 위한 클래스 정의
class NSMCDataset(Dataset):
    def __init__(self, csv_file):
        # 데이터셋 로드 및 전처리
        self.dataset = pd.read_csv(csv_file, sep='\t').dropna(axis=0)
        self.dataset.drop_duplicates(subset=['document'], inplace=True)
        self.tokenizer = AutoTokenizer.from_pretrained("monologg/koelectra-small-v2-discriminator")
        print(self.dataset.describe())
```

project_3_강서구시장 선거예측모델 / projecr3_VS_ 딥러닝 기반 NLP 모델.py

암호 탓하다

```python
# 훈련 및 테스트 데이터셋 로드
train_dataset = NSMCDataset("ratings_train.txt")
test_dataset = NSMCDataset("ratings_test.txt")

# 모델 정의 및 초기화
model = ElectraForSequenceClassification.from_pretrained("monologg/koelectra-base-v3-discriminator").to(de

# 모델 상태 로드
model.load_state_dict(torch.load("model.pt"))

# 모델 학습 설정
epochs = 5
batch_size = 16
optimizer = AdamW(model.parameters(), lr=5e-6)
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=16, shuffle=True)

# 모델 학습 과정
losses = []
```

project_3_강서구시장 선거예측모델 / projecr3_VS_ 딥러닝 기반 NLP 모델.py

암호 탓하다

```python
        accuracies.append(correct.float() / total)
        print("Train Loss:", total_loss, "Accuracy:", correct.float() / total)

# 모델 평가
model.eval()
test_correct = 0
test_total = 0

for input_ids_batch, attention_masks_batch, y_batch in tqdm(test_loader):
    y_batch = y_batch.to(device)
    y_pred = model(input_ids_batch.to(device), attention_mask=attention_masks_batch.to(device))[0]
    _, predicted = torch.max(y_pred, 1)
    test_correct += (predicted == y_batch).sum()
    test_total += len(y_batch)

print("Accuracy:", test_correct.float() / test_total)

# 모델 저장
torch.save(model.state_dict(), "model.pt")
```

모델 학습 및 평가

```
1 losses, accuracies
```

```
([4651.6763158887625,
  3617.4758188501,
  3179.780110117048,
  2830.5081308037043,
  2554.9124857839197],
 [tensor(0.7406, device='cuda:0'),
  tensor(0.8154, device='cuda:0'),
  tensor(0.8438, device='cuda:0'),
  tensor(0.8645, device='cuda:0'),
  tensor(0.8806, device='cuda:0')])
```

```
1 df_jin[["댓글", "예측 감정"]]
```

	댓글	예측 감정
0	이죄명을 위한₩n투표는₩n나라를 망치는일이다₩n김태우 후보님이₩n꼭 승리해야...	긍정
1	김에게 정의를 보여줍시다.진교훈님 당선 응원합니다	긍정
2	진교훈 응원합니다	긍정
3	진교훈후보님. 승리바랍니다. 힘차게응원하겠습니다	긍정
4	응원할게요 꼭승리합니다	긍정
...	...	...
913	내년 총선 선거 민주당 인간들 선거하지 않습니다. 우 👎👎...	긍정
914	이것다진교훈강서구청장당선을추카함다!?~	부정
915	김태우는 공약을 국가 예산을 끌어 올 수 있다, 대통령한테 전화만 하면 된다. 서울...	부정
916	봉투당은 절대 안됩니다₩n나라를 망치는 문이기에	부정
917	진교훈승리하라	부정

918 rows × 2 columns

```
 1 model.eval()
 2
 3 test_correct = 0
 4 test_total = 0
 5
 6 for input_ids_batch, attention_masks_batch, y_batch in tqdm(test_loader):
 7   y_batch = y_batch.to(device)
 8   y_pred = model(input_ids_batch.to(device), attention_mask=attention_masks_batch.to(device))[0]
 9   _, predicted = torch.max(y_pred, 1)
10   test_correct += (predicted == y_batch).sum()
11   test_total += len(y_batch)
12
13 print("Accuracy:", test_correct.float() / test_total)
```

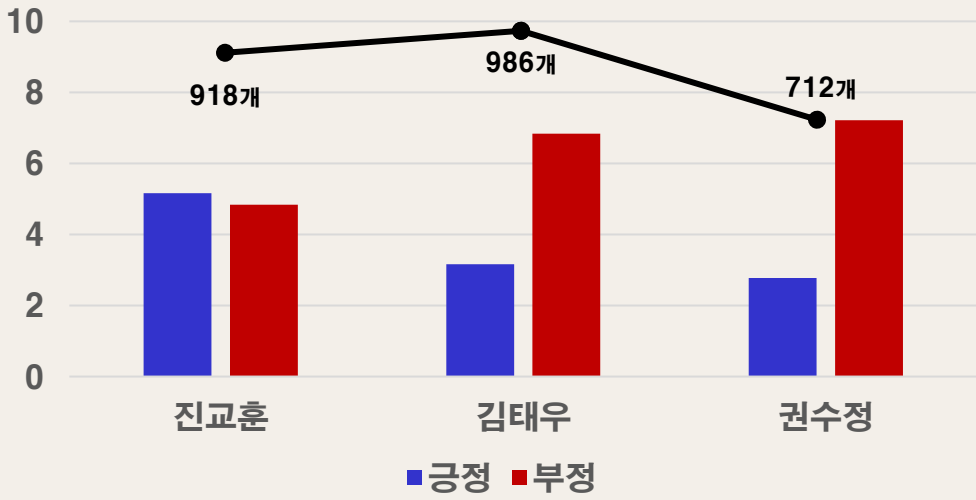
```
  0%|          | 0/3073 [00:00<?, ?it/s]
Accuracy: tensor(0.8505, device='cuda:0')
```

▪ 모델
HuggingFace KoElectra-small을 이용

▪ 모델 과정
1. 손실(losses)은 각 순환이 진행됨에 따라 지속적으로 감소
2. 정확도(accuracies)는 순환이 진행됨에 따라 예측 성능이 향상되고 있다.

▪ 모델 결과
모델의 테스트 데이터셋에 대한 정확도는 약 85.05%

▪ 댓글 처리 결과
댓글의 전처리 과정을 끝낸 뒤 예측 감정 정리



제한점

▪ 제한점

1.영상에 다른 후보자의 응원 댓글을 달아도 " 긍정 " 으로 분류함.
2.선거에 영향을 미치는 다양한 변수를 반영하지는 못함.

▪ 향후 개선 방향

1. 후보자의 이름까지 구분하여 " 긍정/부정 " 을 분류할 수 있도록 모델개선.→ KcElectra
2. 다양한 변수 반영
   여론조사 결과, 뉴스 기사 댓글…등등.

참고문헌

참고	세부
기술적 지원	자연어처리 교제, 파이썬 기초 교제, ChatGPT
데이터 수집	GoogleAPIclient, YouTube
시각화 작업	Matplotlib, PPT, pyLDAvis, Goole Looker Studio
참고 웹페이지	HuggingFace, Github # https://monologg.kr/2020/05/02/koelectra-part1/ # https://github.com/monologg/KoELECTRA