YouTube

Youtube 댓글 감성분석

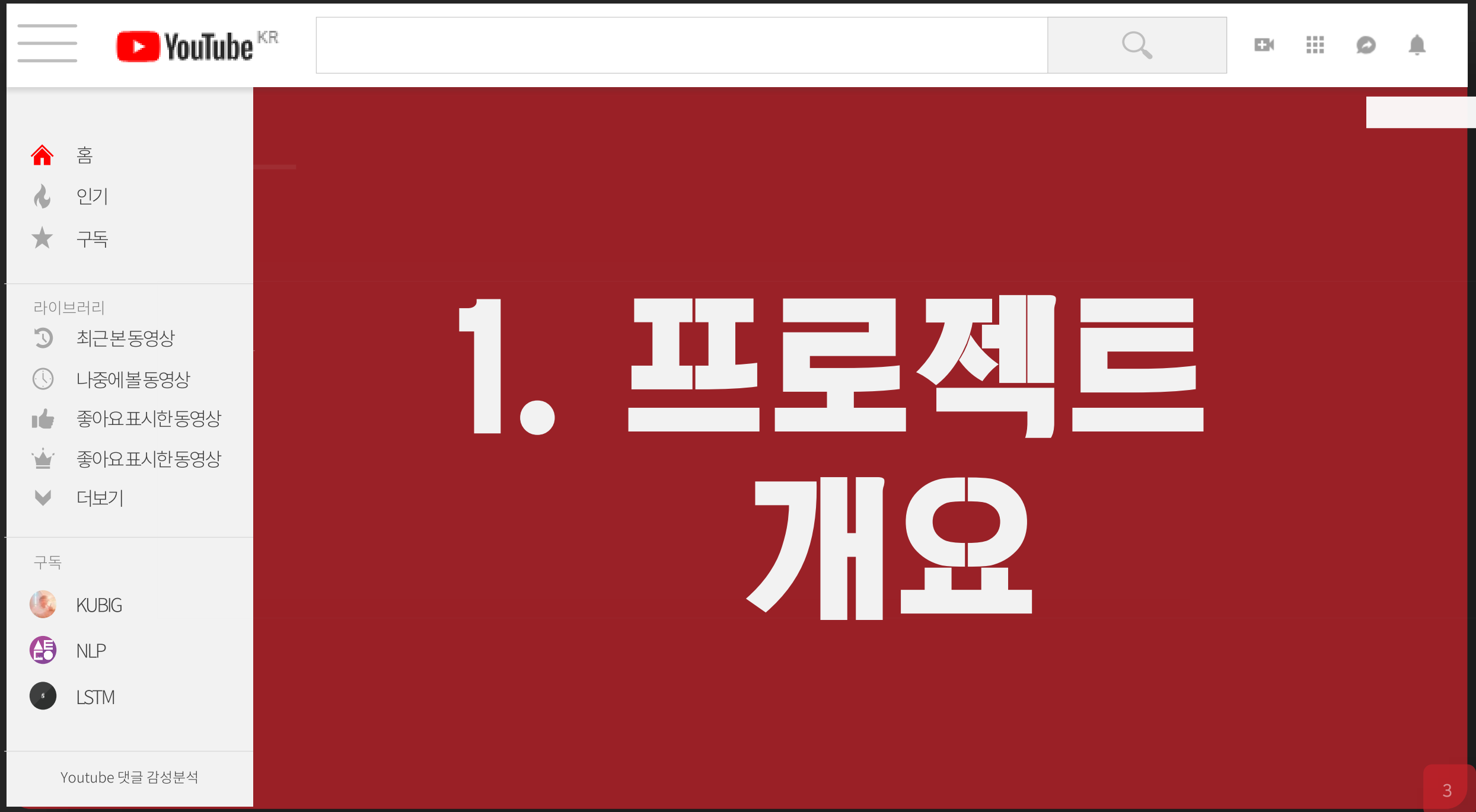
N L P 단 기 p r o j e c t

유정아(조장), 명재성 , 이가영, 이나윤, 임효진, 원윤정

Youtube 댓글 감성분석

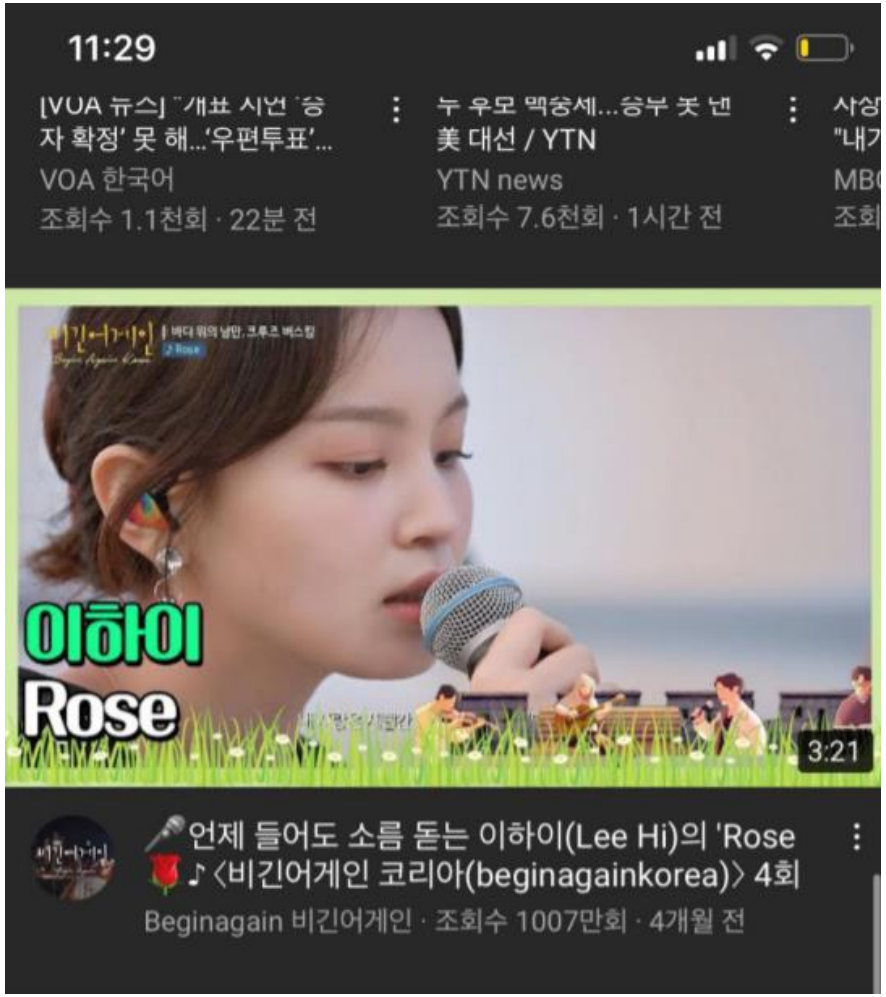
Contents

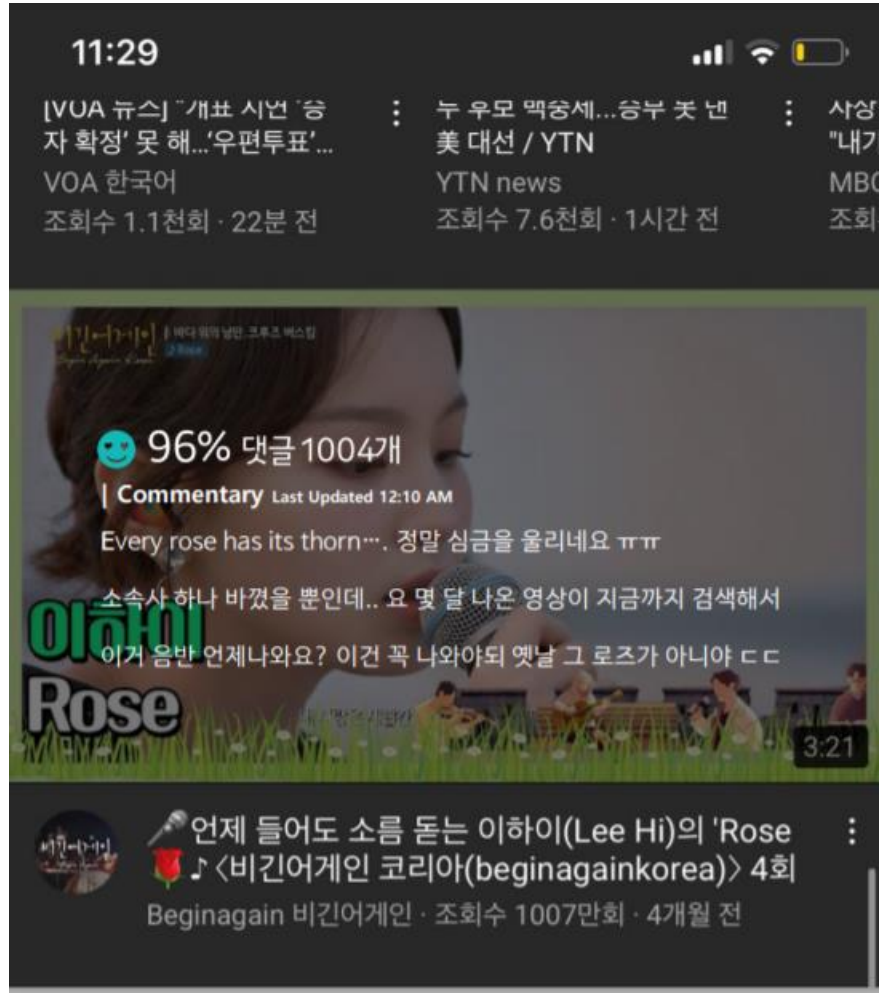
1. 프로젝트 개요
2. NLP 스터디
3. 크롤링
4. Dataset 구성
5. 전처리
6. 모델링
7. 모델 비교
8. 마무리


YouTube KR
홈
인기
구독
라이브러리
최근 본 동영상
나중에 볼 동영상
좋아요 표시한 동영상
좋아요 표시한 동영상
더보기
구독
KUBIG
NLP
LSTM
Youtube 댓글 감성분석

1. 프로젝트 개요

1. 프로젝트 개요


11:29
[VOA 뉴스] "개표 시작 '승자 확정' 못 해...'우편투표'...
VOA 한국어
조회수 1.1천회 · 22분 전
누 우모 맥쿵세...승부 못 낸 美 대선 / YTN
YTN news
조회수 7.6천회 · 1시간 전
이하이
Rose
3:21
언제 들어도 소름 돋는 이하이(Lee Hi)의 'Rose 🌹♪〈비긴어게인 코리아(beginagainkorea)〉 4회
Beginagain 비긴어게인 · 조회수 1007만회 · 4개월 전


11:29
[VOA 뉴스] "개표 시작 '승자 확정' 못 해...'우편투표'...
VOA 한국어
조회수 1.1천회 · 22분 전
누 우모 맥쿵세...승부 못 낸 美 대선 / YTN
YTN news
조회수 7.6천회 · 1시간 전
96% 댓글 1004개
| Commentary Last Updated 12:10 AM
Every rose has its thorn…. 정말 심금을 울리네요 ㅠㅠ
소속사 하나 바꼈을 뿐인데.. 요 몇 달 나온 영상이 지금까지 검색해서
이거 음반 언제나와요? 이건 꼭 나와야되 옛날 그 로즈가 아니야 ㄷㄷ
이하이
Rose
3:21
언제 들어도 소름 돋는 이하이(Lee Hi)의 'Rose 🌹♪〈비긴어게인 코리아(beginagainkorea)〉 4회
Beginagain 비긴어게인 · 조회수 1007만회 · 4개월 전

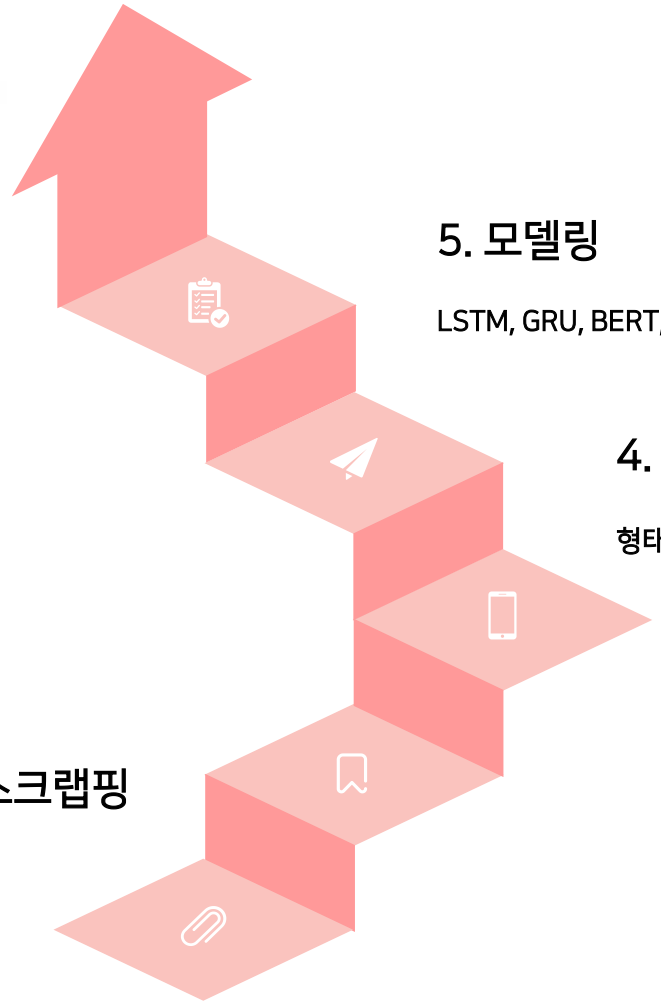
1. 프로젝트 개요

GOAL

감성 분석 이진 분류 목표

Youtube 댓글에 대한

긍정/ 부정 분류하는 모델 구축



5. 모델링

LSTM, GRU, BERT, Bi-LSTM+Attention, CNN+LSTM, CNN+Gru 등 이용

4. 전처리

형태소 분석기 Khaiii, 워드 임베딩 Skip-gram 이용

3. 데이터셋 구성

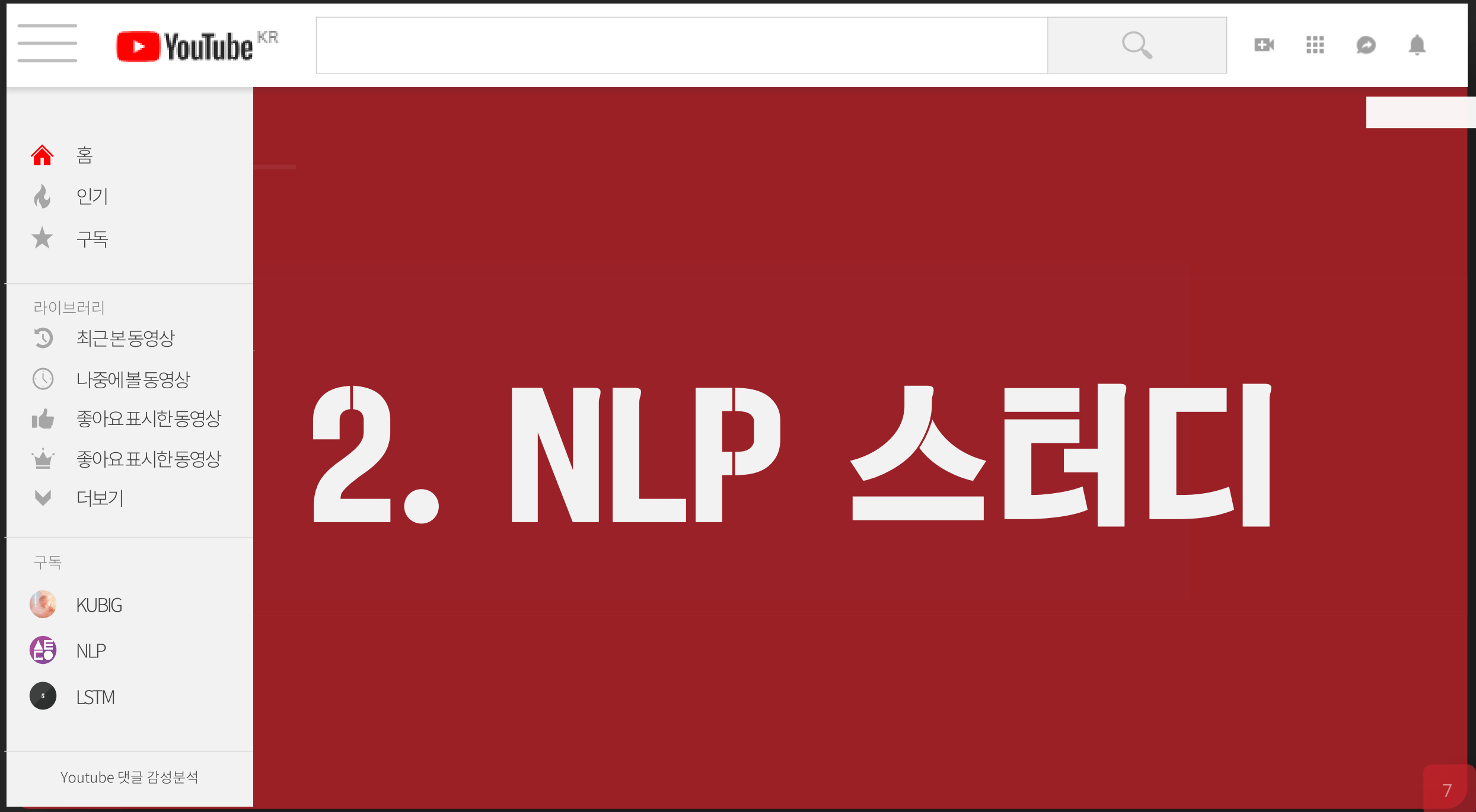
네이버 영화 리뷰 + 유튜브 크롤링 데이터

2. 크롤링 및 스크랩핑

1. NLP 스터디

1. 프로젝트 개요

SUN	MON	TUE	WED	THU	FRI	SAT
				29	30	31
				첫번째 zoom 회의		
1	2	3	4	5	6	7
주제 결정				NLP 프로젝트 개요 발표		
8	9	10	11	12	13	14
	<딥러닝을 이용한 자연어 처리 입문> 스터디 &			RNN을 이용한 텍스트 분류 연습		
15	16	17	18	19	20	21
크롤링 만들기	유튜브 댓글 스크랩핑		전처리/ 모델링 서치팀 준비	두번째 zoom 회의		
22	23	24	25	26	27	28
모델링 및 발표 준비				NLP 프로젝트 최종 발표		
29	30					


YouTube KR
홈
인기
구독
라이브러리
최근 본 동영상
나중에 볼 동영상
좋아요 표시한 동영상
좋아요 표시한 동영상
더보기
구독
KUBIG
NLP
LSTM
Youtube 댓글 감성분석
2. NLP 스터디

2. NLP 스터디

- <딥러닝을 이용한 자연어 처리 입문> Wiki docs

  : 텍스트 전처리, 언어모델, 워드 embedding 등 학습 및 발표

  : RNN을 이용한 텍스트 분류- 네이버 영화/ 쇼핑 리뷰 감성분석 (GRU, LSTM 학습.)

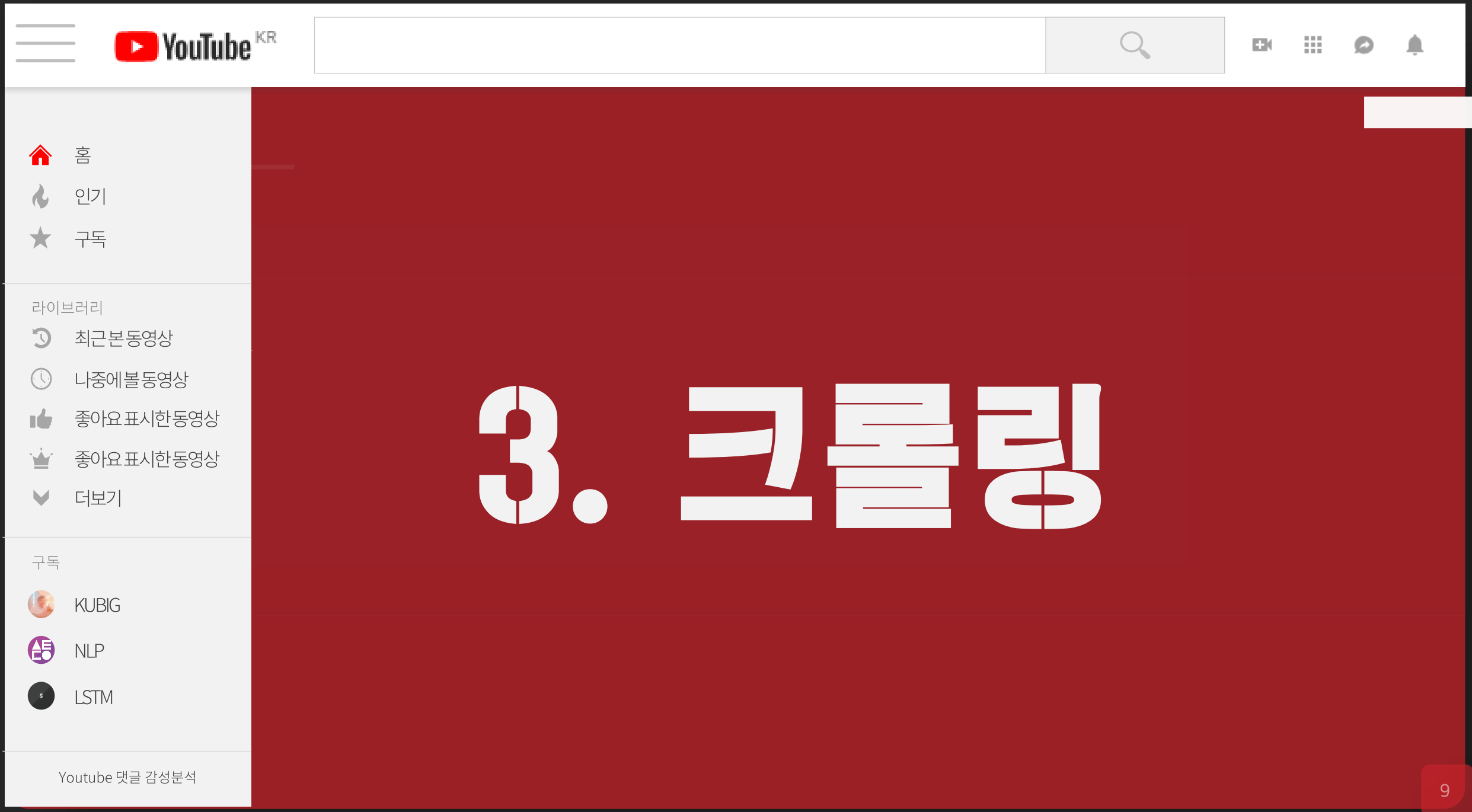
02텍스트전처리_원윤...

03언어모델_이가영.pdf

04카운트 기반의 단어 ...

09순환신경망_임효진....

10워드임베딩_유정아....


YouTube KR
홈
인기
구독
라이브러리
최근 본 동영상
나중에 볼 동영상
좋아요 표시한 동영상
좋아요 표시한 동영상
더보기
구독
KUBIG
NLP
LSTM
Youtube 댓글 감성분석
3. 크롤링

3. 크롤링

```
#원하는 유투브 url 넣어주기
urls = [ 'https://www.youtube.com/watch?v=5OfcbyjdMSI']
html_sources = []

#urls list에 있는 url의 html 소스들을 for 루프를 이용해 가져오기
for i in range (0,1):

    #chromedriver.exe 는 따로 다운 받아서 path 지정해줘야 함. 위에 경로 복사한거 넣으세요.
    driver = webdriver.Chrome('chromedriver',options=options)
    driver.get(urls[i])

    last_page_height = driver.execute_script("return document.documentElement.scrollHeight")

    while True:
        driver.execute_script("window.scrollTo(0, document.documentElement.scrollHeight);")
        time.sleep(3.0)
        new_page_height = driver.execute_script("return document.documentElement.scrollHeight")

        if new_page_height == last_page_height:
            break
        last_page_height = new_page_height

    html_source = driver.page_source
    html_sources.append(html_source)
    print("DONE") #html_source 하나씩 가져와서 더해질 때마다 DONE 출력

    driver.quit()
```

```
def get_user_IDs_and_comments(html_sources):
    my_dataframes = []
    for html in html_sources:

        soup = BeautifulSoup(html, 'lxml')

        #youtube_user_IDs = soup.select('div#header-author > a > span')

        youtube_comments = soup.select('yt-formatted-string#content-text')

        #str_youtube_userIDs = []
        str_youtube_comments = []

        for i in range(len(youtube_comments)):
            #str_tmp = str(youtube_user_IDs[i].text)

            #str_tmp = str_tmp.replace('₩n', '')
            #str_tmp = str_tmp.replace('₩t', '')
            #str_tmp = str_tmp.replace('                ','')
            #str_youtube_userIDs.append(str_tmp)

            str_tmp = str(youtube_comments[i].text)
            str_tmp = str_tmp.replace('₩n', '')
            str_tmp = str_tmp.replace('₩t', '')
            str_tmp = str_tmp.replace('                ', '')

            str_youtube_comments.append(str_tmp)

        pd_data = {"Comment":str_youtube_comments}

        youtube_pd = pd.DataFrame(pd_data)

        my_dataframes.append(youtube_pd)

    return my_dataframes

my_dataframes = get_user_IDs_and_comments(html_sources)
my_dataframes
```

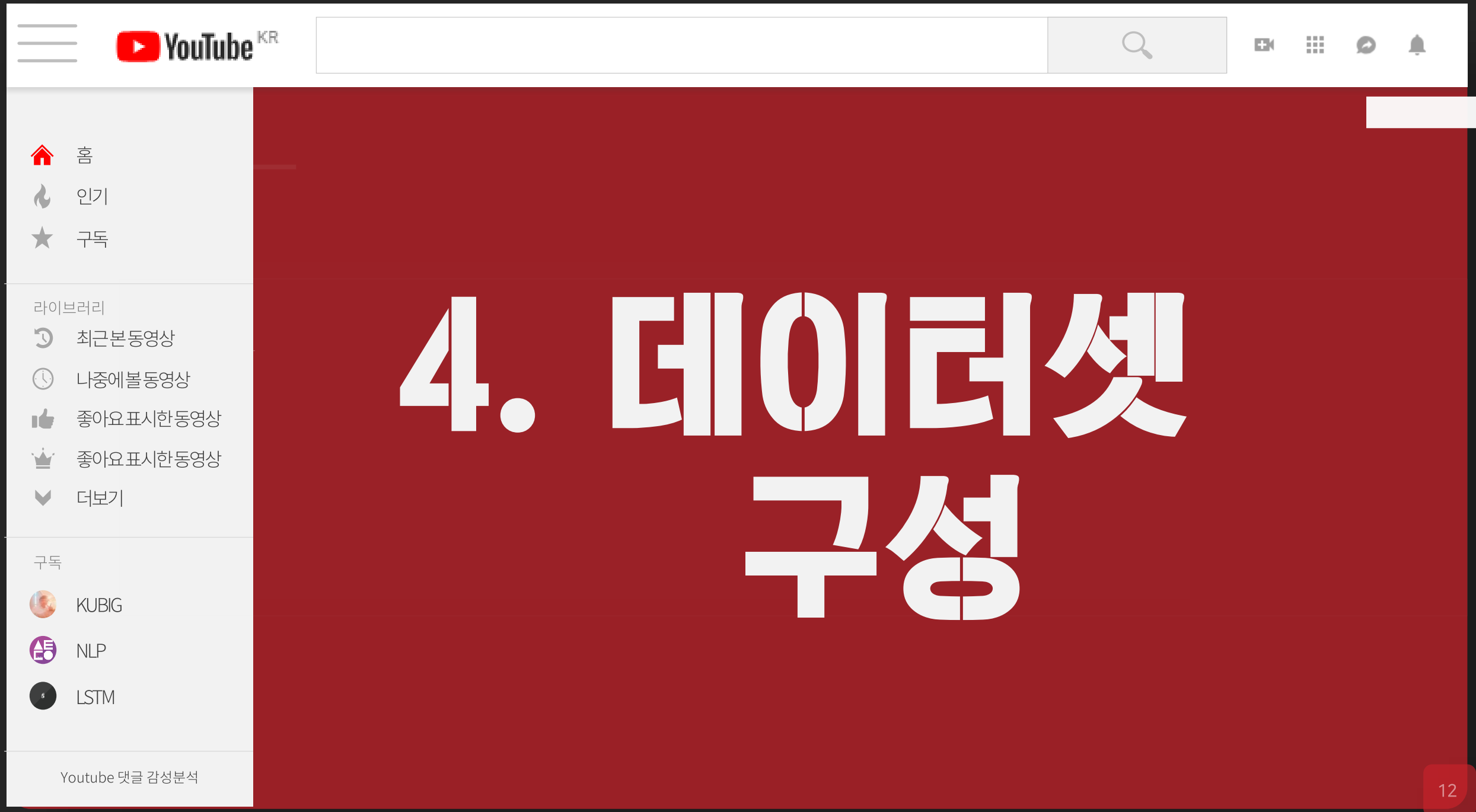
: 유튜브 댓글 크롤링 (Made by. 조장님!)

3. 크롤링

fx | Label

	A	B
1	Label	Comment
2	0	혁신은 민간주도, 정부는 혁신조력 이라고 말은 항상 하면서 왜 공무원을 더 뽑아서 정부를 비대하게 만드는지 좀 이해가 안됩니다.
3		4차산업혁명시대에 주입식교육의 인재는 필요없다. 그것은 인공지능이 더 잘하니까 말이다. 앞으로 대학교는 의미가 없는시대가 온다. 결국 다양한 인재가 필요하다는 말
4	1	좋은 말씀 해주신 위원장님, 인터뷰 준비해준 EO 모두 감사합니다😁😁
5	1	정말 좋은 영상입니다.
6	1	인재는 성과로 평가 받는다! 꼭 기억해야겠습니다😁 좋은 인터뷰 감사합니다!
7	1	이번에 컴업2019 에서 실시간 방송하신거 올려주신다구 하셨었죠?😆(기대만빵) 태용님 질문도 너무 너무 좋고 답변도 한문장 한문장 너무 너무 도움이 되서 꼭 편집 많이 없
8	1	역시 이번에도 좋은 영상!!!!
9	1	정리 잘해주심. 저장했어유.
10	1	좋은 영상 감사합니다!
11	1	해외에서 공부중입니다. 동의하고 변화가 생기길바랍니다
12	1	장병규 위원장님이자 선배님, 영상에서 뵈게 될 줄 몰랐네요.항상 감사드립니다.
13	1	공감하는 영상입니다... 특히나 교육쪽에 대해서는 예전부터 제가 생각해왔던 이야기를 하시는걸 보고 크게 공감했습니다. 아직 풀어나갈 숙제가 많지만 위원장님 같은 분들(
14	1	영상 보며 많은 울림이 있습니다.민간 주도, 그 자체가 혁신임을 입증해주셨습니다.
15	1	읽어볼게요~ 감사합니다~
16	1	다 맞는 말 같아요 근무제도 획일적이기 때문에 많은 불만이 나오는거 같고 특히 대학의 이상적인 모습이 위원장님이 말하신 모습같아요
17		성과가 이제 큰 잣대가 될것 같네요!뭔가뜨끔합니다
18	1	태용님 존경합니다.행동하는 사람이 가장 위대한 사람이라 생각합니다.!!
19	0	글 너무 빨리 넘어가네요,,
20	0	혁신의 기회가 있음에도 정치질에 기회를 잃게 되는게 너무 안따까울 따름입니다.
21	0	소오올찍히 나라가 뭔가 추진하면 되는게 없다. 부정적 네트워크만 뻗칠 뿐이다.. 기술/산업 분야는 그냥 운동장 조성만 잘 해줘도 정부 역할은 다 한거다고 보면 된다... 거시
22	0	4차사업자시고 님들아 제발 애플페이나 되게해보세요. 엑티브x도 좀 없애주시구요. 공인인증서도 없애주시구요. 해외안나가보시것도 아니고, 위의 상황들 없어져도 세상 잘
23	1	좋은 영상 감사합니다.
24		전통적인 관료(공무원)이랑 프로그래머(4차산업) 일도 같이 하고 싶은데, 너무 욕심이 많은걸까요? 시간이 부족할수도 있을거 같은데, 두마리 토끼를 잡고 싶당~~!! ㅋㅋ나는
25	0	대학 이름만 따지는 한국사회에서 대학 다양화가 과연 재대로 될까 의문이네요.. 카이스트 포항공대와 같은 출신대학을 염두하신것은 아닌지.. 아마 대학의 다양화가 아니라
26		'민간'의 이름으로 등장하는 주체들이 누구인지도 중요하다고 생각합니다. 저는 이미 '민간'이 '서민'은 아닌 상태인 것 같고, 끝없는 탐욕에 국가의 방향을 내맡기는 위험을 내

: **크롤링**을 이용하여 총 3505개의 **유튜브 댓글 긍정 1, 부정 0**으로 Labeling
(중립은 NA값으로 한 번에 제거)


YouTube KR
홈
인기
구독
라이브러리
최근 본 동영상
나중에 볼 동영상
좋아요 표시한 동영상
좋아요 표시한 동영상
더보기
구독
KUBIG
NLP
LSTM
Youtube 댓글 감성분석

4. 데이터셋 구성

4. 데이터셋 구성

=

NAVER 영화

NSMC Training set
100,000개

=

NAVER 영화

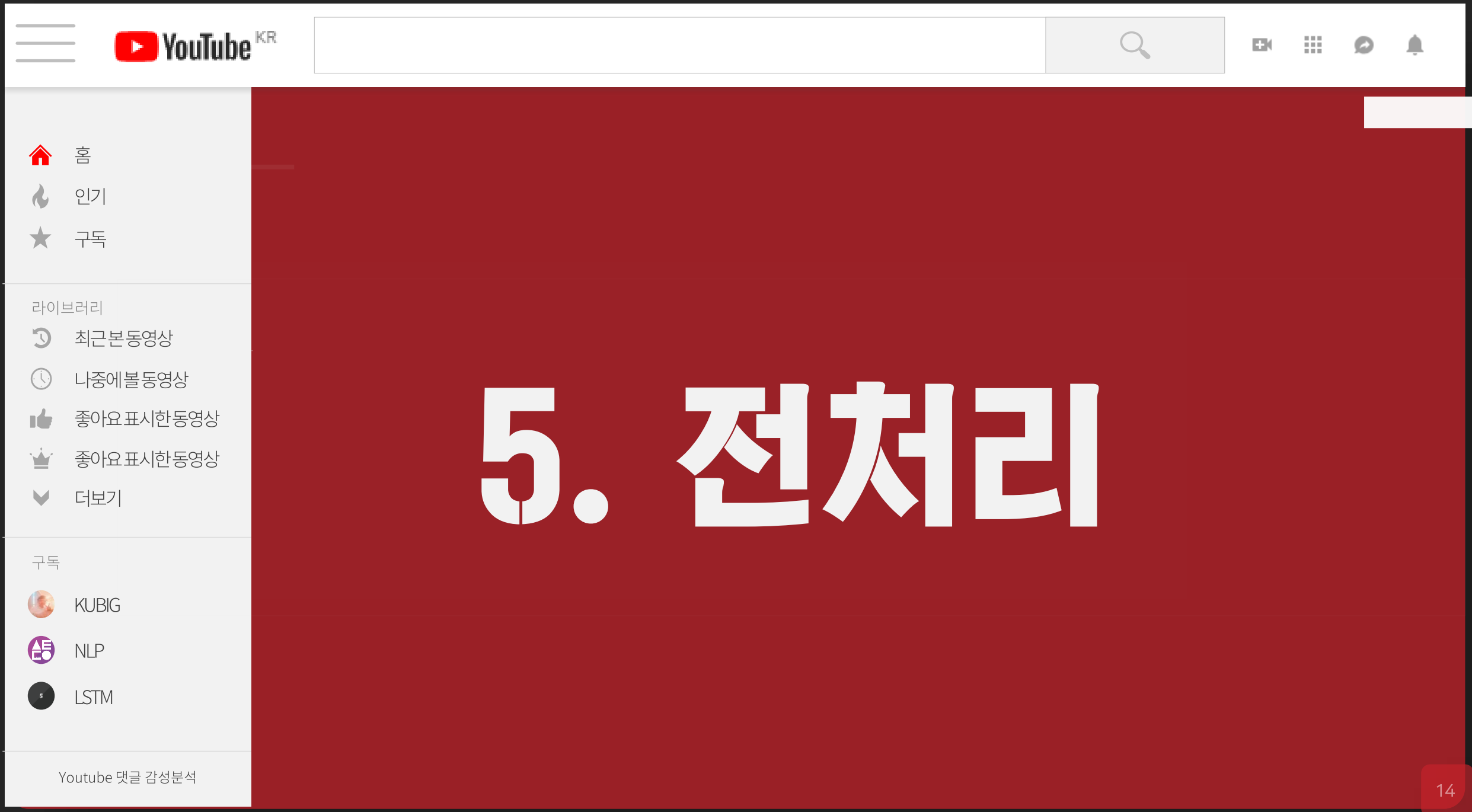
NSMC Test set
50,000개

+

YouTube KR

직접 크롤링한
유튜브 댓글 labeled data 3505개

: 최종 학습 데이터 100,000개, 테스트 데이터 53505개


YouTube KR
홈
인기
구독
라이브러리
최근 본 동영상
나중에 볼 동영상
좋아요 표시한 동영상
좋아요 표시한 동영상
더보기
구독
KUBIG
NLP
LSTM
Youtube 댓글 감성분석
5. 전처리

5. 전처리

- EDA, 중복값 제거

```
[ ]  test_data.drop_duplicates(subset=['document'], inplace=True)
```

```
[ ]  test_data = test_data.dropna(how = 'any')
```

- 한글만 남기기 (+특수부호 제거)

```
[ ]  test_data['document'] = test_data['document'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
```

```
[ ]  punct = "/-'?!.,#$%&'()*+-/:;<=>@[₩₩]^_`{|}~" + '""“”’' + '∞θ÷α•à−β∅³π‘₹´°£€₩×™√²—–&'
     punct_mapping = {"‘": "'", "₹": "e", "´": "'", "°": "", "€": "e", "™": "tm", "√": " sqrt ", "×": "x", "²": "2", "—": "-", "–": "-", "’": "'", "_": "-", "`": "'", '“': '"'}
```

```
[ ]  def clean_punc(text, punct, mapping):
         for p in mapping:
             text = text.replace(p, mapping[p])

         for p in punct:
             text = text.replace(p, f' {p} ')

         specials = {'₩u200b': ' ', '…': ' ... ', '₩ufeff': '', 'करना': '', 'है': ''}
         for s in specials:
             text = text.replace(s, specials[s])

         return text.strip()
```

```
[ ]  cleaned_corpus = []
     for sent in test_data['document']:
         cleaned_corpus.append(clean_punc(sent, punct, punct_mapping))
```

```
[ ]  def clean_text(texts):
         corpus = []
         for i in range(0, len(texts)):
             review = re.sub(r'[@%₩₩*=()/~#&₩+á?₩xc3₩xa1₩-₩|₩.₩:₩;₩!₩-₩,₩_₩~₩$₩'₩"]', '',str(texts[i]))
             review = re.sub(r'₩d+','', str(texts[i]))
             review = review.lower()
             review = re.sub(r'₩s+', ' ', review)
             review = re.sub(r'<[^>]+>','',review)
             review = re.sub(r'₩s+', ' ', review)
             review = re.sub(r"^₩s+", '', review)
             review = re.sub(r'₩s+$', '', review)
             corpus.append(review)
         return corpus
```

```
[ ]  basic_preprocessed_corpus = clean_text(cleaned_corpus)
```

5. 전처리

- 띄어쓰기(PyKoSpacing)

```
[ ]  !pip install git+https://github.com/haven-jeon/PyKoSpacing.git
```

```
[ ]  from pykospacing import spacing
```

```
[ ]  corpus=[]
     for sent in basic_preprocessed_corpus:
         spaced_text = spacing(sent)
         corpus.append(spaced_text)
```

```
[ ]  spaced_corpus=corpus
```

: PyKoSpacing

(한국어 띄어쓰기 패키지로, 띄어쓰기가 되어있지 않은 문장을
띄어쓰기를 한 문장으로 변환해준다.)

- 맞춤법검사(Py-hanspell, soynlp)

```
[ ]  !pip install git+https://github.com/ssut/py-hanspell.git
```

```
[ ]  from hanspell import spell_checker
```

```
[ ]  !pip install soynlp
```

```
[ ]  from soynlp.normalizer import *
```

```
[ ]  corpus=[]
     for sent in spaced_corpus:
       corpus.append(str(sent))
```

```
[ ]  aa = spaced_corpus[0:10000]
```

```
[ ]  corpus=[]
     for sent in aa:
       spelled_sent = spell_checker.check(sent)
       checked_sent = spelled_sent.checked
       normalized_sent = repeat_normalize(checked_sent)
       corpus.append(normalized_sent)

     preprcessed_corpus=corpus
```

Py-Hanspell, soynlp :

(Py-hanspell: 네이버 한글 맞춤법 검사를 바탕으로 만들어진 패키지)
(Soynlp: 품사 태깅, 단어 토큰화 등을 지원하는 단어 토크나이저)

```
[ ]  aa= pd.DataFrame(aa)
     aa.to_csv("test_1.csv", index=False)
     from google.colab import files
     files.download('test_1.csv')
```

5. 전처리

- 형태소 분석 및 품사 tagging

```
[10] api = KhaiiiApi()
```

```
[11] significant_tags = ['NNG', 'NNP', 'NNB', 'VV', 'VA', 'VX', 'MAG', 'MAJ', 'XSV', 'XSA']
```

```
[12] def pos_text(texts):
         corpus = []
         for sent in texts:
             pos_tagged = ''
             for word in api.analyze(sent):
                 for morph in word.morphs:
                     if morph.tag in significant_tags:
                         pos_tagged += morph.lex + '/' + morph.tag + ' '
             corpus.append(pos_tagged.strip())
         return corpus
```

```
[13] tagged_corpus1=pos_text(dat1["document"])
```

```
[14] tagged_corpus2=pos_text(dat2["document"])
```

```
[15] tagged_corpus3=pos_text(dat3["document"])
```

```
[16] tagged_corpus4=pos_text(dat4["document"])
```

: Khaiii, Mecab을 이용하여
형태소 분석 및 품사 tagging 진행.

Khaiii (Kakao Hangul Analyzer III)로 결정.

```
[17] p1 = re.compile('[가-힣A-Za-z0-9]+/NN. [가-힣A-Za-z0-9]+/XS.')
     p2 = re.compile('[가-힣A-Za-z0-9]+/NN. [가-힣A-Za-z0-9]+/XSA [가-힣A-Za-z0-9]+/VX')
     p3 = re.compile('[가-힣A-Za-z0-9]+/VV')
     p4 = re.compile('[가-힣A-Za-z0-9]+/VX')
```

```
[18] tag_corp=tagged_corpus1+tagged_corpus2+tagged_corpus3+tagged_corpus4
```

5. 전처리

- 워드 임베딩

```
[37] from gensim.models import Word2Vec
     model = Word2Vec(sentences=removed_stopword_corpus, size=100, window=5, min_count=5, workers=4, sg=1)
```

```
[38] model.wv.vectors.shape
```

```
     (10620, 100)
```

```
[39] from gensim.models import KeyedVectors
     model.wv.save_word2vec_format('khaii_skipgram')
```

: Skip-gram, Cbow, Keras Tokenizer를

모두 사용해본 후, Skip-gram으로 결정.

5. 전처리

- 워드 임베딩

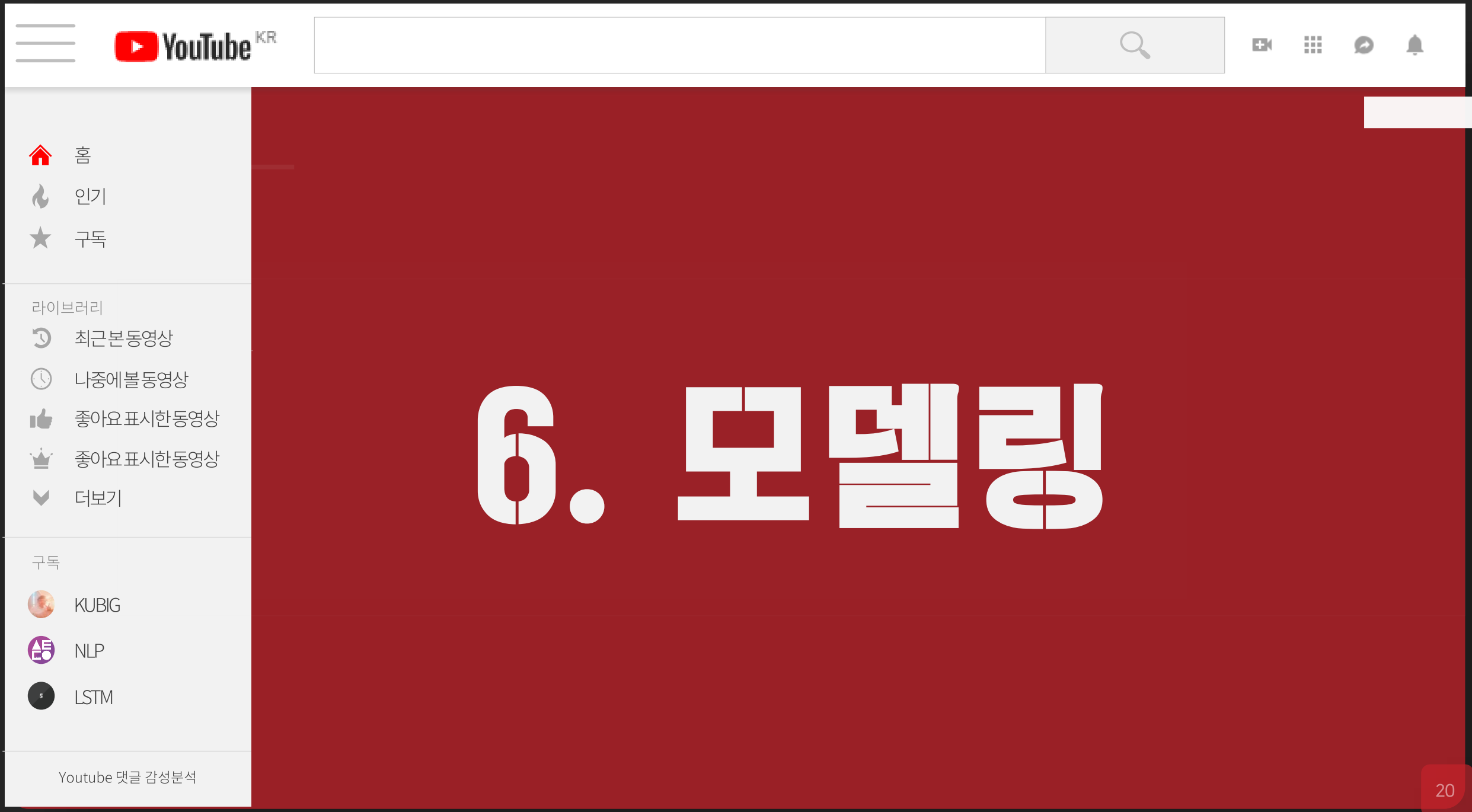
```
[ ]  model.wv.doesnt_match("멜로 호러 여자 남자 키스".split())
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: Call to deprecated `wv`
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.6/dist-packages/gensim/models/keyedvectors.py:895: FutureWarning: arrays to stack mus
  vectors = vstack(self.word_vec(word, use_norm=True) for word in used_words).astype(REAL)
/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of the second argu
  if np.issubdtype(vec.dtype, np.int):
'호러'
```

```
[ ]  model.wv.most_similar("잼")
```

: Skip-gram으로 W2V를 돌려본 모습

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: Call to deprecated `wv`
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of the second argu
  if np.issubdtype(vec.dtype, np.int):
[('꿀', 0.8075603246688843),
 ('잼임', 0.7585766315460205),
 ('허니잼', 0.7512593269348145),
 ('개꿀', 0.7318193912506104),
 ('개잼', 0.724430501461029),
 ('좋', 0.6969881057739258),
 ('잼이', 0.6955959796905518),
 ('재밋', 0.6942168474197388),
 ('좋잼', 0.6913444399833679),
 ('씀', 0.689312219619751)]
```


YouTube KR
홈
인기
구독
라이브러리
최근 본 동영상
나중에 볼 동영상
좋아요 표시한 동영상
좋아요 표시한 동영상
더보기
구독
KUBIG
NLP
LSTM
Youtube 댓글 감성분석
6. 모델링

6. 모델링

Bert

: 사전 훈련 언어모델로,
사전 훈련 전 word piece별 embedding을 사용하는 model

```
[ ]  # 분류를 위한 BERT 모델 생성
     model = BertForSequenceClassification.from_pretrained("bert-base-multilingual-cased", num_labels=2)
     model.cuda()
```

```
[ ]  # 옵티마이저 설정
     optimizer = AdamW(model.parameters(),
                       lr = 2e-5, # 학습률
                       eps = 1e-8 # 0으로 나누는 것을 방지하기 위한 epsilon 값
                     )

     # 에폭수
     epochs = 3

     # 총 훈련 스텝 : 배치반복 횟수 * 에폭
     total_steps = len(train_dataloader) * epochs

     # 처음에 학습률을 조금씩 변화시키는 스케줄러 생성
     scheduler = get_linear_schedule_with_warmup(optimizer,
                                                 num_warmup_steps = 0,
                                                 num_training_steps = total_steps)
```

Bi-LSTM + attention

: 순방향, 역방향 lstm을 동시 사용,
해당 시점에 따라 연관 비율 다르게 해주는 model

```
[128] from tensorflow.keras.layers import Dense, Embedding, Bidirectional, LSTM, Concatenate, Dropout
      from tensorflow.keras import Input, Model
      from tensorflow.keras import optimizers
      import os
```

```
[129] sequence_input = Input(shape=(max_len,), dtype='int32')
      embedded_sequences = Embedding(vocab_size, 128, input_length=max_len, mask_zero = True)(sequence_input)
```

```
[130] lstm = Bidirectional(LSTM(64, dropout=0.5, return_sequences = True))(embedded_sequences)
```

```
[131] lstm, forward_h, forward_c, backward_h, backward_c = Bidirectional ₩
        (LSTM(64, dropout=0.5, return_sequences=True, return_state=True))(lstm)
```

```
[132] print(lstm.shape, forward_h.shape, forward_c.shape, backward_h.shape, backward_c.shape)

      (None, 35, 128) (None, 64) (None, 64) (None, 64) (None, 64)
```

```
[133] state_h = Concatenate()([forward_h, backward_h]) # 은닉 상태
      state_c = Concatenate()([forward_c, backward_c]) # 셀 상태
```

```
[134] attention = BahdanauAttention(64) # 가중치 크기 정의
      context_vector, attention_weights = attention(lstm, state_h)
```

```
[135] dense1 = Dense(20, activation="relu")(context_vector)
      dropout = Dropout(0.5)(dense1)
      output = Dense(1, activation="sigmoid")(dropout)
      model = Model(inputs=sequence_input, outputs=output)
```

6. 모델링

GRU

: 사전 훈련된 언어모델로, padding한 트레인데이터에 대해,
embedding matrix를 구하여 훈련시키는 model

```
[ ] from gensim.models import Word2Vec
    model = Word2Vec(sentences=removed_stopword_corpus, size=200, window=5, min_count=3, workers=4, sg=1)
```

```
[ ] embedding_matrix = np.zeros((vocab_size, 200))
```

```
[ ] def get_vector(word):
        if word in model:
            return model[word]
        else:
            return None
```

```
[ ] for word, i in t.word_index.items():
        temp = get_vector(word)
        if temp is not None:
            embedding_matrix[i] = temp
```

```
[ ] from tensorflow.keras.layers import Embedding, Dense, GRU, Dropout
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.models import load_model
    from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

```
[ ] model = Sequential()
    e = Embedding(vocab_size, 200, weights=[embedding_matrix], input_length=max_len, trainable=False)
    model.add(e)
    model.add(GRU(128))
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.3))
    model.add(Dense(1, activation="sigmoid"))
    es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
    mc = ModelCheckpoint('best_model', monitor='val_acc', mode='max', verbose=1, save_best_only=True)
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

```
[ ] history = model.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size=60, validation_split=0.2)
```

CNN+GRU

: CNN의 속도와 경량함을
RNN의 순서 감지 능력과 결합한 model

```
[ ] model_hybrid= Sequential()
    e = Embedding(vocab_size, 200, weights =[embedding_matrix], input_length=max_len, mask_zero=True, trainable=True)
    model_hybrid.add(e)
    model_hybrid.add(Conv1D(32,kernel_size=3,padding='same',activation='relu'))
    model_hybrid.add(MaxPooling1D(pool_size=3))
    model_hybrid.add(Dropout(0.3))
    model_hybrid.add(Conv1D(64,kernel_size=3,padding='same',activation='relu'))
    model_hybrid.add(MaxPooling1D(pool_size=3))
    model_hybrid.add(Dropout(0.35))
    model_hybrid.add(Conv1D(128,kernel_size=3,padding='same',activation='relu'))
    model_hybrid.add(MaxPooling1D(pool_size=3))
    model_hybrid.add(Dropout(0.4))
    model_hybrid.add(GRU(50,return_sequences=True))
    model_hybrid.add(Dropout(0.25))
    model_hybrid.add(Flatten())
    model_hybrid.add(Dense(128,activation='relu'))
    model_hybrid.add(Dropout(0.45))
    model_hybrid.add(Dense(1,activation='sigmoid'))

    model_hybrid.summary()
```

```
[ ] model_hybrid.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
[ ] history3=model_hybrid.fit(X_train, y_train, epochs=5, callbacks=[es3, mc3], batch_size=500, validation_split=0.2)
```

6. 모델링

CNN+LSTM

: CNN의 convolution layer를 적용하여 단어의 지역 특징을 추출하고, pooling layer에서 LSTM을 이용하여 공간 및 시간적 특징을 동시에 고려

```
model_cnnLSTM = Sequential()
e = Embedding(vocab_size, 200, weights =[embedding_matrix], input_length=max_len, mask_zero=True, trainable=True)
model_cnnLSTM.add(e)
model_cnnLSTM.add(Conv1D(128, 5, strides=1, padding='valid', activation='relu'))
model_cnnLSTM.add(MaxPooling1D(pool_size=4))
model_cnnLSTM.add(Dropout(0.25))
model_cnnLSTM.add(Bidirectional(LSTM(128)))
model_cnnLSTM.add(Dropout(0.25))
model_cnnLSTM.add(Dense(1, activation='sigmoid'))
model_cnnLSTM.summary()
```

```
model_cnnLSTM.compile(optimizer='Adam', loss='binary_crossentropy', metrics=['acc'])
history = model_cnnLSTM.fit(X_train, y_train, epochs=30, callbacks=[es, mc], batch_size=32, validation_split=0.2)
```

CNN

```
model = Sequential()
model.add(Embedding(vocab_size, 200, embeddings_initializer = Constant(embedding_matrix),
                    input_length = max_len, trainable = True))
model.add(Conv1D(16, 7, activation='relu'))
model.add(MaxPooling1D(3))
model.add(Conv1D(16, 7, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

```
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics=['acc'])
```

```
history2=model.fit(X_train, y_train, epochs=15, callbacks=[es2, mc2], batch_size=512, validation_split=0.2)
```

6. 모델링

LSTM

: RNN의 각 유닛에 이전 데이터의 정보를 저장하고 있는 메모리셀 함수
& 메모리셀을 유지/업데이트할지 결정하는 게이트 함수를 추가한 model

```
[ ]  X_train=pad_sequences(X_encoded, maxlen=max_len, padding='post')
     y_train=np.array(y_train)
     print(X_train)
     embedding_matrix = np.zeros((vocab_size, 100))

     def get_vector(word):
         if word in model:
             return model[word]
         else:
             return None

     for word, i in t.word_index.items():
         temp = get_vector(word)
         if temp is not None:
             embedding_matrix[i] = temp
```

```
[ ]  from keras.layers import Embedding, Dense, LSTM, Flatten
     from keras.models import Sequential
     from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
     from tensorflow.keras.models import load_model
```

```
[ ]  model3 = Sequential()
     model3.add(Embedding(vocab_size, 100, weights=[embedding_matrix],input_length=max_len, trainable=False))
     model3.add(LSTM(256))
     model3.add(Dense(1, activation='sigmoid'))
```

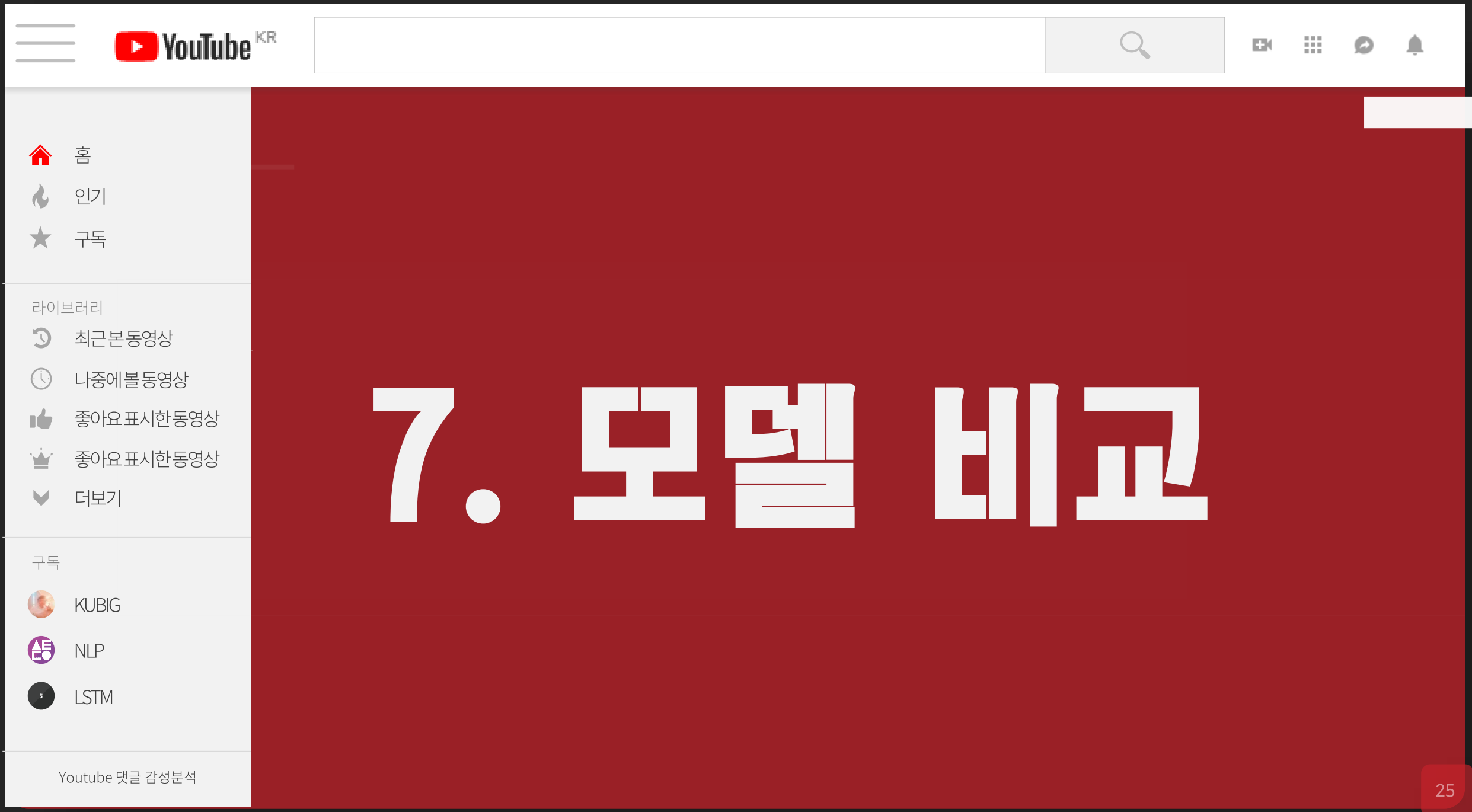
```
[ ]  es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
     mc = ModelCheckpoint('best_model.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)
```

```
[ ]  model3.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
     history = model3.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size=256, validation_split=0.2)
```

GRU+relu

```
model_g = Sequential()
e = Embedding(vocab_size, 200, weights =[embedding_matrix], input_length=max_len, mask_zero=True, trainable=False)
model_g.add(e)
model_g.add(GRU(128))
model_g.add(Dense(128, activation='relu'))
model_g.add(Dropout(0.2))
model_g.add(Dense(3, activation = 'softmax'))
```

```
model_g.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model_g.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size=60, validation_split=0.2)
```


YouTube KR
홈
인기
구독
라이브러리
최근 본 동영상
나중에 볼 동영상
좋아요 표시한 동영상
좋아요 표시한 동영상
더보기
구독
KUBIG
NLP
LSTM
Youtube 댓글 감성분석
7. 모델 비교

7. 모델 비교

Test data (네이버 영화리뷰 50000개+유튜브 댓글 3505개)에 대한 Model들의 accuracy(%)

Model	BERT	Bi-LSTM+A	CNN+GRU	CNN
accuracy	85	81.22	79.10	78.51
Model	CNN+LSTM	LSTM	GRU+RELU	GRU
accuracy	70.66	54.73	51.25	48.75

7. 모델 비교

Cnn & GRU (80% accuracy)

```
[165] lets_predict_2('노답이네 이게 뭐냐...')
```

83.32% 확률로 부정 리뷰입니다.

```
[163] lets_predict_2('반하겠다 잘생김 한도 초과!!!')
```

82.94% 확률로 부정 리뷰입니다.

```
[164] lets_predict_2('색감이 대박~')
```

85.58% 확률로 긍정 리뷰입니다.

```
[144] lets_predict_2('무조건 2탄 나와야된다')
```

53.62% 확률로 긍정 리뷰입니다.

```
[166] lets_predict_2('남편이 빌런이네 아내 불쌍해ㅠㅠ')
```

99.35% 확률로 부정 리뷰입니다.

```
[175] lets_predict_2('이 영상 보지 않았을 때로 돌아갈래...')
```

87.21% 확률로 부정 리뷰입니다.

```
[160] lets_predict_2('평생 구독할게요 고맙습니다')
```

90.06% 확률로 긍정 리뷰입니다.

```
[178] lets_predict_2('집사님 귀여워요..저도 고양이..')
```

64.87% 확률로 긍정 리뷰입니다.

```
[162] lets_predict_2('이 드라마 너무 기대된다')
```

90.94% 확률로 긍정 리뷰입니다.

7. 모델 비교

Bert (85% accuracy)

```
[63] logits = test_sentences(['노답이네 이게 뭐냐'])
     prob(logits)
```

약 87.39% 확률로 부정 리뷰입니다.

```
[64] logits = test_sentences(['최악이다 영상 왜이래'])
     prob(logits)
```

약 95.64% 확률로 부정 리뷰입니다.

```
[65] logits = test_sentences(['반하겠다 잘생김 한도 초과'])
     prob(logits)
```

약 90.49% 확률로 부정 리뷰입니다.

```
[66] logits = test_sentences(['색감이 대박'])
     prob(logits)
```

약 82.47% 확률로 긍정 리뷰입니다.

```
[67] logits = test_sentences(['무조건 2탄 나와야된다'])
     prob(logits)
```

약 57.03% 확률로 부정 리뷰입니다.

```
[68] logits = test_sentences(['남편이 빌런이네 아내 불쌍해'])
     prob(logits)
```

약 51.44% 확률로 부정 리뷰입니다.

```
[69] logits = test_sentences(['엄마랑 보다가 사레 걸리는줄'])
     prob(logits)
```

약 80.10% 확률로 부정 리뷰입니다.

```
[70] logits = test_sentences(['평생 구독할게요 감사합니다'])
     prob(logits)
```

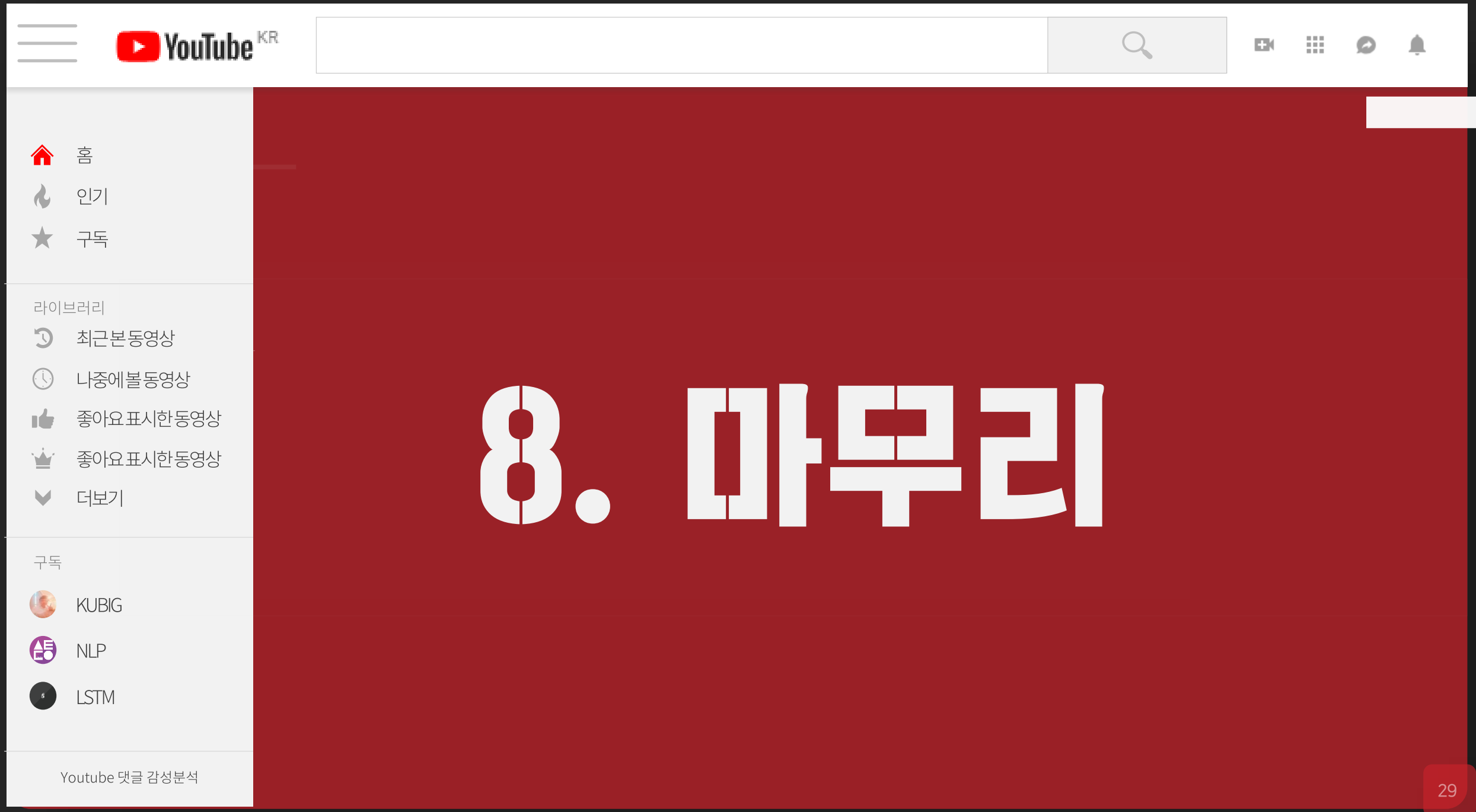
약 90.07% 확률로 긍정 리뷰입니다.

```
[71] logits = test_sentences(['작가님 이러시면 너무 사랑합니다'])
     prob(logits)
```

약 88.03% 확률로 긍정 리뷰입니다.

```
[72] logits = test_sentences(['전처리팀 너무 고생 많았어요'])
     prob(logits)
```

약 72.84% 확률로 긍정 리뷰입니다.


YouTube KR
홈
인기
구독
라이브러리
최근 본 동영상
나중에 볼 동영상
좋아요 표시한 동영상
좋아요 표시한 동영상
더보기
구독
KUBIG
NLP
LSTM
Youtube 댓글 감성분석
8. 마무리

8. 마무리 및 느낀 점

아쉬운 점

- 다양한 전처리 방식과 모델에 대한 개념이 어떻게 코드로 구현되는지 팀원 모두가 공부하기엔 무리가 있었다.
- 기간이 짧아 충분한 사전 준비 없이 준비하다 보니 오류 발생이 잦았다.
- 전과정에서 인터넷 참고 후의 빠른 실행을 하다 보니, 창의적인 모델 구축에는 무리가 있었다.
- 다 같이 오프라인으로 밥..한..번도 못 먹었다.
- W2V size가 작았던 점. 한국어 특성(교착어)상 전처리에 있어 매우 까다로웠고, Skip-gram 성능을 높이기 어려웠다.

좋았던 점

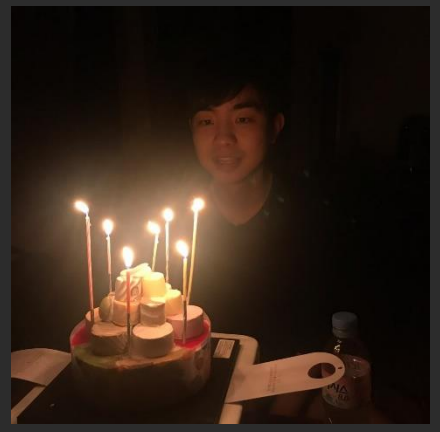
- 팀원분들이 성실하시고 똑똑하셔서 많이 배웠다.
- 전처리에서 온갖 오류를 상대하면서 맷집과 인내심이 강해졌다. (매우 다사다난했다..)
- 직접 데이터 수집, 전처리, 모델링의 과정을 거쳐 뜻 깊었다 .

팀 구성원 소개

12기 임효진

12기 원윤정

11기 유정아



11기 명재성

12기 이가영

12기 이나윤

감사합니다!