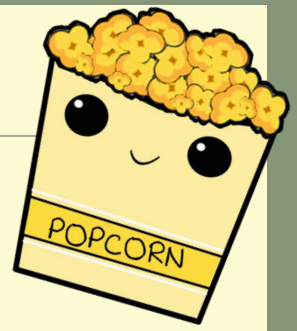
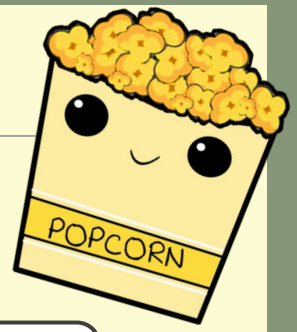
키워드를 활용한 사용자 리뷰 기반 네이버 영화 추천시스템

오프라인 4조 김도균(조장) 박강인 박선홍 반위홍



발표 순서



발표 순서

1. 구현 과정

(1) Crawling

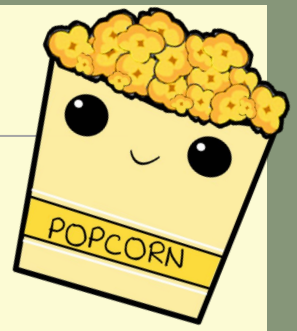
(2) Preprocessing

(3) Modeling

(4) Application

2. Application 시연

3. Q&A

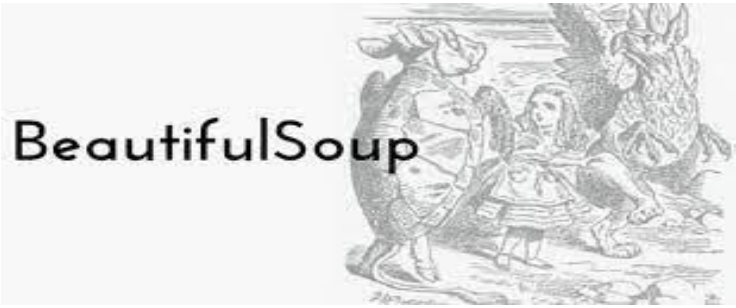


Crawling

DATA crawling(1)

BeautifulSoap

- BeautifulSoup는 웹 상의 가치있는 정보를 빠르게 크롤링 하기위한 도구로써 진입 장벽이 매우 낮고 간결하다.
- urlib2 와 requests로 HTML 소스를 가져와야 웹 사이트 크롤링 가능.

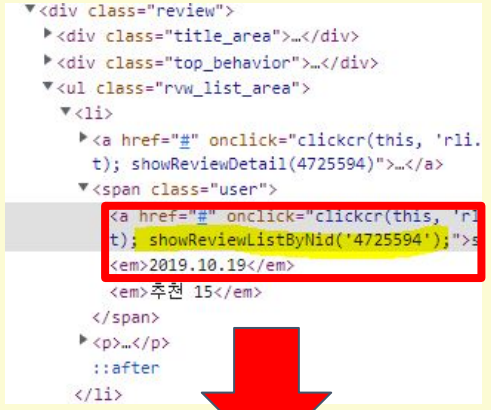


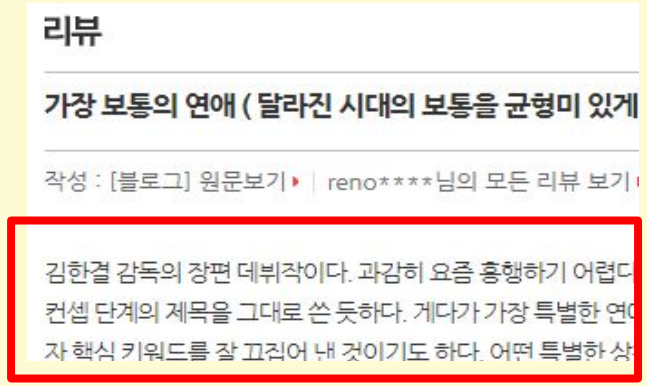
Scrapy

- Scrapy는 파이썬으로 작성 되었으며, spider를 작성해서 크롤링하거나 BeautifulSoup, lxml과 연동해서 사용한다
- BeautifulSoup에서는 지원하지 않는 XPath를 사용해 복잡한 HTML 소스를 쉽게 크롤링 할 수 있음

자료가 적고 빠른 Crawling을 원하면 가볍고 빠른 Beautiful Soup
자료가 방대하고 Xpath를 이용한 Crawling을 원하면 유연한 Scrapy

DATA crawling(2)






```
#cod.append(code)
for page in tqdm(range(1, num//10+2)):
    url = f"https://movie.naver.com/movie/bi/mi/review.na
    gain = requests.get(url)
    html = BeautifulSoup(gain.content, 'html.parser')
    html2 = str(html)
    for j in range(10):
        # 유의미한 내용이 담긴 content까지 접근
        p2 = re.compile(r"showReviewListByNid\('(.*?)'")
        try:
            nim.append(p2.findall(html2)[j])
```

```
parse(self, response):
titles = response.xpath('//*[@id="content"]/div[1]/div[2]/div[1]/h3/a/text(
reviews = [' '.join(line.strip()
        for line in p.xpath('.//text()').extract() # p 태그에서 text를 추출하
        if line.strip() )
    for p in response.xpath('//*[@id="content"]/div[1]/div[4]/div[1]//p')]
```

- html.parser를 이용해 파싱한 html문서생성
- 정규식을 이용하여 리뷰의 수와 리뷰 코드 추출

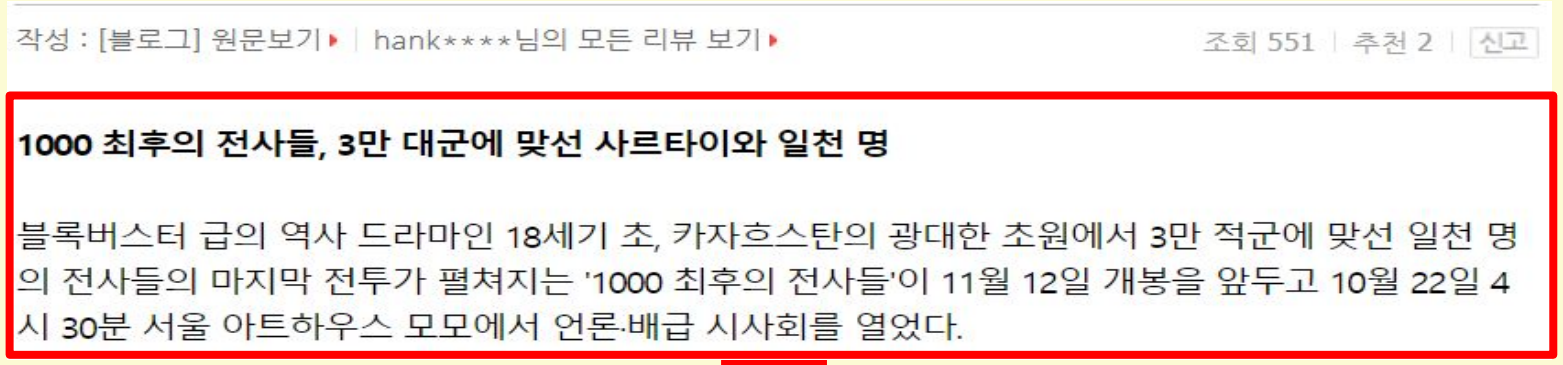
- 리뷰가 있는 Xpath경로 지정(<p>Tag 경로까지)
- 해당 경로에 있는 모든 Text를 한 문장으로 생성

DATA crawling(3)

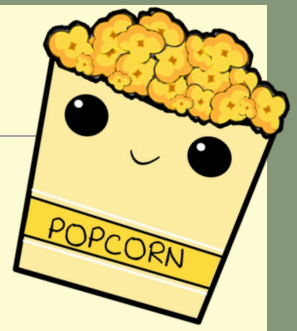
데이터 정의 :

네이버 영화리뷰

총 3643개 데이터 : 14년도~20년도 개봉영화 중에서 블로그 리뷰가 3개이상 100개이하인 데이터 사용



	title
1000 : 최후의 전사들	최후 전사 카자흐스탄 만들 카자흐스탄 하여 호기심 보게 되었다 카자흐스탄 그리 부유한 나라 아니라고 했고 예술 경제력 연관 그러하다는 먹었어도 기대 선입견 이기 모든 구성요소 마음 드는 걸작 만난다는 힘든 최후 전사 카자흐스탄 웅담 일반 웅담 달리 영웅 죽음 난다 카자흐스탄 실제 벌어졌던 전쟁 주제 카자흐스탄 르 타이 부족 약탈 실감 나고 잔인하다 잔인한 전장 남달랐던 사르 타이 구사일생 살아나게 사르 타이가 어린이 젊은이 본격 시작 으례 웅담 그러하듯이 사르 타 료 배신 이겨내고 카자흐스탄 전쟁 승리 이끄는 밀알 영웅 된다는 뼈대 카자흐스탄 자연 생활 대하 되었고 예전 있었다 러브씬 남녀 붙잡는 순진한 러브씬 가족 구 젊은 시절 초원 달리 젊은이 사이 그렇듯이 질투 질투 감정 부작용 주변 망치 최고 러브씬 출정 나서는 카자흐 부족 전사 가운데 사르 타이 사르 타이 죽지 여 만들었구나 들었다 카자흐스탄 어예쁜 외모 가진 김태희 같고 그런다는 농담 섞인 떠오른다 보니 헛말 아닌것 스타 정예 인턴 기자 최후 전사 익스펜더블 참 모으고 카자흐스탄 역사상 최대 예산 투입 최후 전사 카이 전투 묘사 의상 전투 방식 역사 고증 심혈 기울여 세기 기마 전투 생생하게 그렸다 사르 타이 넘는 원 는 익힘으로써 실제 전투 생생함을 담아 있게 최후 전사 실감 전투 이란 지르 국제 영화제 기술 예술 성취 심사 위원 특별상 수상하기도 한편 최후 전사 앞두고 맞선 명의 전사 마지막 전투 펼쳐진다 칭기즈칸 제국 재건 하려는 준가 카자흐 민족 학살 폭정 일삼 는다 가족 몰살 마을 태워지는 목격 사르 타이 용맹한 청년



Preprocessing

Preprocessing(1)

title	review
니키타	니키타,리뷰보기,예전에 본 영화,처음 니키타가 요원으써 임무를 부여 받고 임무 수행 후 사방에 쫓아 오는 경찰들을 뒤로 미리 약속 된 탈출 경로인 창문을 여는 순간 벽으로 꽉 막혔있는 장면..,오늘 왠지 그 장면이 자꾸만 머리에서 떠나지 않는다,추천하였습니다.
니키타	니키타,리뷰보기,,"이름이 뭐야?","니!키!타!",,그녀의 속은 분노로 꽉 차 있다. 부글부글 끓어 조금만 건드려도 뿜어낼 것만 같은 마그마처럼. 무엇이 그렇게 만들었나. 그녀의 광기는 정체모를 기관과 '밥'에 의해 식고 있다. 그녀는 킬러가 되었다. 새로운 신분을 얻었다. '조세핀'이 되었다. 그렇게 다시 세상으로 나왔다. 사랑하는 사람이 생겼다. 그에게 그녀는 '마리'가 된다. 마리는 행복하다. 세상의 다른 면이 보인다. 그러나 그녀는 명령에 의존하는 조세핀이다. 그녀 안의 니키타가 언제 튀어나올지 알 수 없다. 복잡하다.,,그녀의 사랑 '마르코'가 묻는다. '밥'에게.,"이제 그녀의 삶을 찾으면 안 되나요?",,,,,,,,,,,,,,,,,,,,,,추천하였습니다.

	title	review
0	어롱더로드사이드	,,,,영화정보,,,,http://www.imdb.com/title/tt2290113/,,,,예고편,,,,,추천하였습니다.
1	어롱더로드사이드	,,"나와 함께 갈래요?",자신을 둘러싼 모든 것이 불만인 방황하는 청춘 바니. 반면 회색 눈으로 세상을 보지만 항상 밝은 그녀 니나. 둘은 LA에서 열리는 뮤직 페스티벌에 동행하게 된다. 서로 다른 사연과 감성을 가진 이들은 긴 여정 동안 서로의 상처를 음악으로 감싸주며 치유를 얻게 되고 점점 서로를 알아가며 마음을 열게 되는데.....,,영화정보 : http://movie.naver.com/movie/bi/mi/basic.nhn?code=115964,,추천하였습니다.

전처리1

nan값 제거

중복값 제거

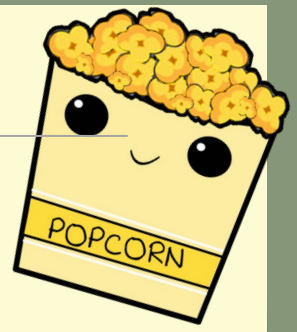
전처리2

review column에서
한글 이외 모두 제거

전처리3

review column에
'추천하였습니다.'제거

Preprocessing(2)



OKT 한글 토큰화

OKT : Open-source Korean Text processor의 약자로 트위터에서 개발한 Twitter 한국어 처리기에서 파생된 오픈소스 한국어 처리기

토큰화

```
['최후', '전사', '카자흐스탄']
['만들', '카자흐스탄', '하여']
['카자흐스탄', '그리', '부유한']
['나라', '아니라고', '했고']
['예술', '경제력', '연관']
['그러하다는', '먹었어도', '기대']
```

품사태깅

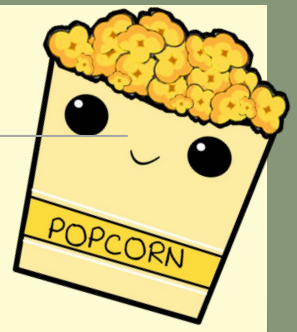
```
[('최후', 'Noun'), ('전사', 'Noun')]
[('카자흐스탄', 'Noun'), ('만들', 'Verb')]
[('카자흐스탄', 'Noun'), ('하여', 'Verb')]
[('호기심', 'Noun'), ('보게', 'Verb')]
[('그리', 'Verb'), ('부유한', 'Adjective')]
[('나라', 'Noun'), ('아니라고', 'Adjective')]
[('했고', 'Verb'), ('예술', 'Noun')]
```

명사, 형용사 추출

	word	class
0	최후	Noun
1	전사	Noun
2	카자흐스탄	Noun
4	카자흐스탄	Noun
6	호기심	Noun
9	카자흐스탄	Noun
11	부유한	Adjective
12	나라	Noun
13	아니라고	Adjective

OKT 문장 토큰화 → 형태소 분석→ 품사태깅(명사, 형용소만 포함) 방식으로 전처리한 3643개의 데이터 저장

Preprocessing(3)



KOMORAN 한글 토큰화

KOMORAN : KOreanMORphologicalANalyer는 자바로 구현된 한국어 형태소 분석기이다.

토큰화

```
['간', '만', '진정']
['하', 'ㄴ', '청소년']
['ㄴ', '제국', '부활']
['오', '았', '습니다']
['가끔', '청소년', '불가']
['도대체', '청부', 'ㄹ']
```

품사태깅

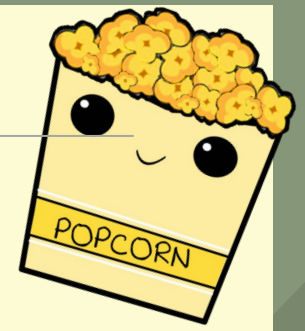
```
('간', 'NNB'), ('만', 'JX')]
('진정', 'XR'), ('하', 'XSA')]
('ㄴ', 'ETM'), ('청소년', 'NNG')
('불가', 'NNG'), ('영화', 'NNG')
('제국', 'NNP'), ('부활', 'NNP')
('오', 'VX'), ('았', 'EP')]
('습니다', 'EC'), ('가끔', 'MAG'
```

명사, 어간 추출

2	진정
5	청소년
6	불가
7	영화
10	제국
11	부활
16	청소년
17	불가
19	청부

KOmoran 문장 코큰화 → 품사태깅(명사, 어간만 포함) 방식으로 전처리한 3643개의 데이터 저장

Preprocessing(4)

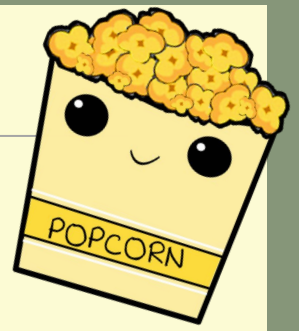


워드 클라우드

불용어 제거

```
stopwords = pd.read_csv('./stopwords_small.csv', index_col=0)
# 불용어를 리스트에 추가하고 실행
a = pd.DataFrame({'stopword':['렉스','년대','메르','일자리','여전히','동영상 첨부','만드는','모습','과정']})
stopwords = pd.concat([stopwords, a],ignore_index=True)
stopwords = stopwords.drop_duplicates()
print(stopwords.duplicated().sum())
stopwords.to_csv('./stopwords_small.csv')
```

워드 클라우드를 참고하여 stopwords small.csv에 불용어를 추가 하면서 저장 하는 방식으로 불용어 파일 생성

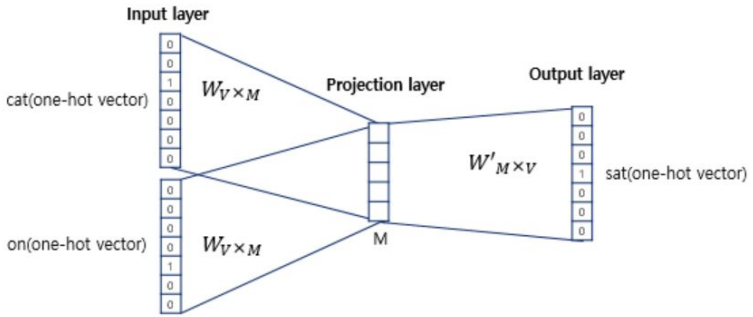


Modeling

Modeling – Word2Vec & TFIDF Model

Word2Vec

- 분산표현을 활용한 2개의 히든레이어를 사용하여 단어를 벡터로 변환해주는 얕은 뉴럴 네트워크 모델



TFIDF
(Term Frequency – Inverse Document Frequency)

- 문서에서 단어가 얼마나 등장하는지를 나타내는 TF값과 전체문서에서 단어가 얼마나 자주 등장하는지 나타내는 DF값의 역수인 IDF값을 곱한 값을 사용하는 모델

$$\mathrm{tf}(t,d) = 0.5 + \frac{0.5 \times \mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}}$$

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Modeling – 유사도 도출(1)

1. Word2Vec 모델 구성

```
for sentence in clean_token_review:
    token = sentence.split(' ')
    cleaned_tokens.append(token)
print(cleaned_tokens[0])
embedding_model = Word2Vec(cleaned_tokens, vector_size=300,
                           window=30, min_count=20,ns_exponent=0.75,
                           workers=-1)
embedding_model.save('./word2VecModel_final.model')
```

2. 키워드 유사도 도출

```
### 유사도 추출
if key_word in embedding_model.wv.index_to_key:
    sim_word = embedding_model.wv.most_similar(key_word, topn=10)
    print(sim_word)
    labels = []
    for label, _ in sim_word:
        labels.append(label)
    print(labels)
    #### 토큰 추출 및 유사도 생성
    for i, word in enumerate(labels):
        sentence += [word] * (9-i)
sentence = ' '.join(sentence)

### 이전 학습 모델 불러오기
with open('./tfidf.pickle', 'wb') as f:
    pickle.dump(Tfidf, f)
mmwrite('./tfidf.mtx', Tfidf_matrix)
sentence_vec = Tfidf.transform([sentence])
### 코사인 유사도 계산
cosine_sim = linear_kernel(sentence_vec,
                           Tfidf_matrix)
### 함수 적용
recommendation = getRecommendation(cosine_sim)
```

3. 문장과 Cosine유사도가 영화 제목 출력

```
['결박', '절하', '미안함', '레닌', '프랑스', '폭압', '넘치면서도', '이난', '똑같지는', '고초']
==========================================================================================
스포츠 스포츠 스포츠 스포츠 스포츠 스포츠 스포츠 스포츠 스포츠 스포츠 결박 결박 결박 결박 결박 결
레닌 레닌 레닌 프랑스 프랑스 프랑스 프랑스 프랑스 폭압 폭압 폭압 폭압 넘치면서도 넘치면서도 넘치면
```

Modeling – 유사도 도출(2)

1. TFIDF모델 구성

```
### tfidf 모델생성
Tfidf = TfidfVectorizer(sublinear_tf=True,ngram_range=(1,2))
Tfidf_matrix = Tfidf.fit_transform(
    df_review_one_sentence['review'])
### pickle 형태 저장
with open('./tfidf.pickle', 'wb') as f:
    pickle.dump(Tfidf, f)
### mtx 형태 저장
mmwrite('./tfidf.mtx', Tfidf_matrix)
```

2. 키워드 유사도 도출

```
### 유사도 추출
if key_word in embedding_model.wv.index_to_key:
    sim_word = embedding_model.wv.most_similar(key_word, topn=10)
    labels = []
    for label, _ in sim_word:
        labels.append(label)
    #### 토큰 추출 및 유사도 생성
    for i, word in enumerate(labels):
        sentence += [word] * (9-i)
sentence = ' '.join(sentence)
Tfidf = TfidfVectorizer(sublinear_tf=True)
Tfidf_matrix = Tfidf.fit_transform(
    df_review_one_sentence['review'])
### 이전 학습 모델 불러오기
with open('./tfidf.pickle', 'wb') as f:
    pickle.dump(Tfidf, f)
mmwrite('./tfidf.mtx', Tfidf_matrix)
sentence_vec = Tfidf.transform([sentence])
### 코사인 유사도 계산
cosine_sim = linear_kernel(sentence_vec,
                           Tfidf_matrix)
### 함수 적용
recommendation = getRecommendation(cosine_sim)
```

3. 문장과 Cosine유사도가 영화 제목 출력

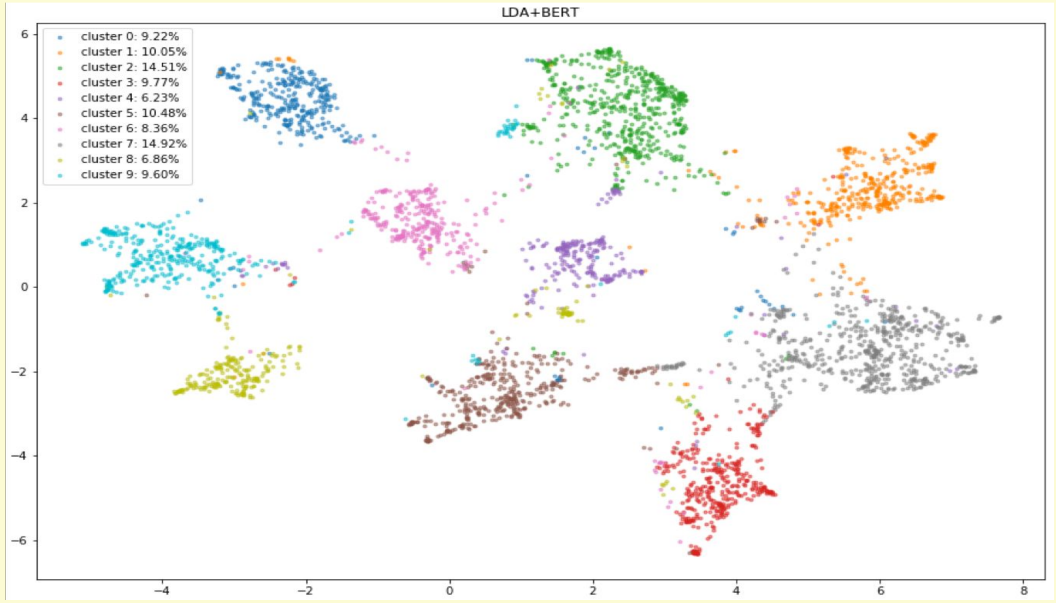
	title
0	이태원
1	스톤
2	바웬사, 희망의 인간
3	쿠크하트 : 시계심장을 가진 소년
4	에덴: 로스트 인 뮤직
5	조조 래빗
6	넥스트 제네레이션 패트레이버
7	스케치
8	바다로 가자
9	데싸우 댄서스

Modeling – Topic Model(1)

토픽 모델링(Topic Modeling)이란?

토픽 모델 (Topic model) 이란, 문서 집합의 추상적인 주제를 발견하기 위한 통계적 모델

중요한 특징으로 레이블이 된 데이터가 필요 없고, 스스로 패턴을 식별하는 비지도학습 방식



잠재 디리클레 할당
(Latent Dirichlet Allocation)

- 확률 기반의 토픽 모델링
- 단어가 특정 토픽에 존재할 확률과 문서에 특정 토픽이 존재할 확률을 추정하여 토픽 추출

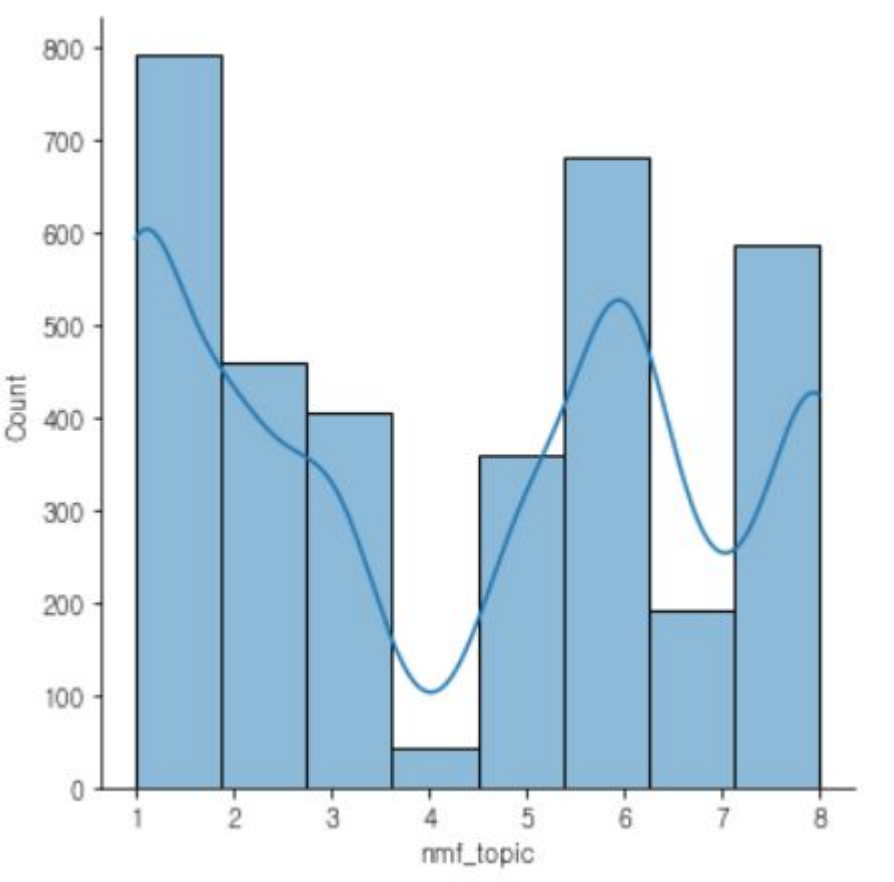
비음수 행렬 분해
(Non-negative Matrix Factorization)

- 행렬 분해 기반의 토픽 모델링
- 행렬 V를 두 개의 행렬 W와 H로 분해되는 선형대수와 다변량 분석으로 토픽 추출
- 섞여 있는 데이터에서 원본/성분을 구분하는데 사용
  ex) 시각 처리, 문서 분류, 음파 분석

Modeling – Topic Model(2)

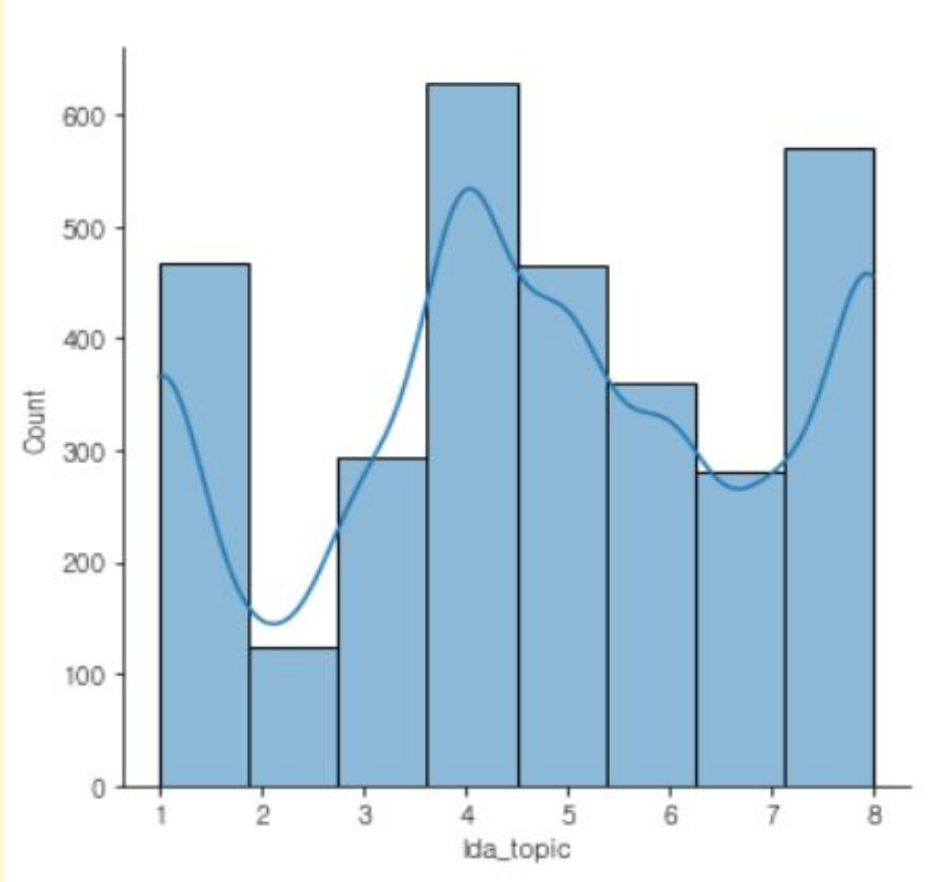
NMF

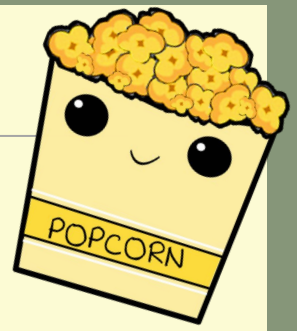
특정 토픽에 치우침



LDA

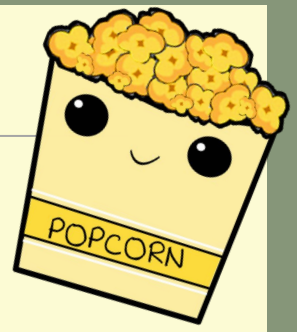
토픽에 치우치지 않고 균등하게 분류





Application

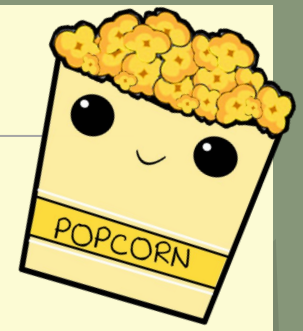
Application – Django(1)



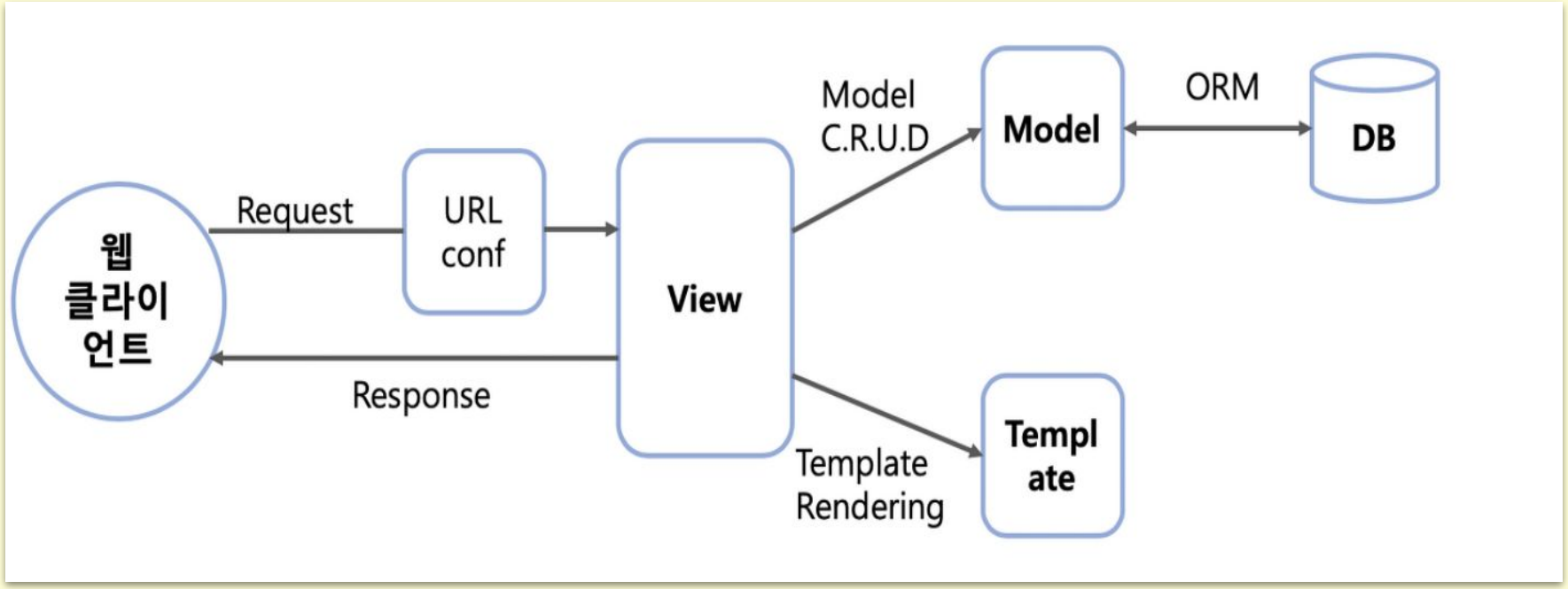
Django

– 웹 프로그램을 위해 만들어야 할 기능들을 프로그램에 내장하고 있는 파이썬 기반의 웹 프레임 워크

– 새로 개발할 필요 없이 내장된 기능만을 이용해 빠른 개발을 할 수 있다는 장점이 있다.



Django 실행 과정



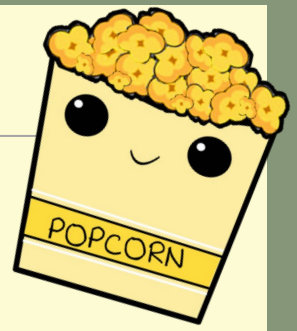
프로젝트 생성 → model 개발 → URL conf 생성 →
Template 생성 및 수정 → views 수정 → 웹 서버 실행

Application – PyQt(1)

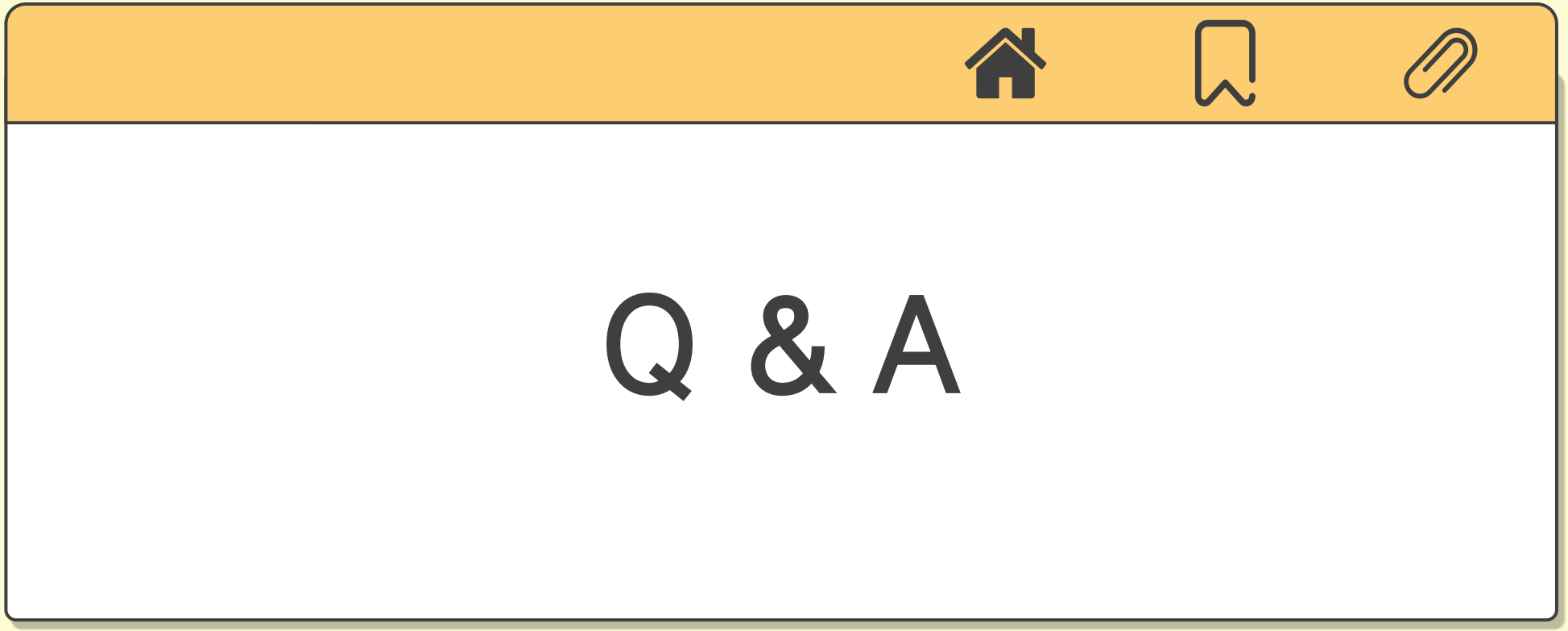
PyQt

- 파이썬에서 GUI(Graphical User Interface) 프로그래밍을 할 때 사용하는 대표적인 패키지

- 그래픽 사용자 인터페이스, 네트워킹, 스레드, 정규표현식, SQL 데이터베이스, SVG, OpenGL, XML, 사용자 및 응용 프로그램 설정, 위치서비스, 단거리통신 (예:NFC 및 블루투스) 및 클라우드 액세스 등 1000개가 넘는 클래스를 Python 모듈 세트로 구현하고 제공.



Application 시연


Q & A