



여행지 추천 및 검색 알고리즘 개발

Multi-search: 민소연 박지원 장윤서

목차

#01 프로젝트 개요

- 주제 선정 동기
- 프로젝트 진행 계획

#02 프로젝트 진행 과정

- 데이터셋 구축 과정
- 모델 개발 과정

#03 프로젝트 결과

- 프로토 타입 시연 결과
- 보완점 및 기대 효과



프로젝트 개요



#01 프로젝트 주제 선정 동기

원하는 분위기의 여행지를
텍스트만으로 검색하기 어려웠던 경험



마을 같은 여행지?



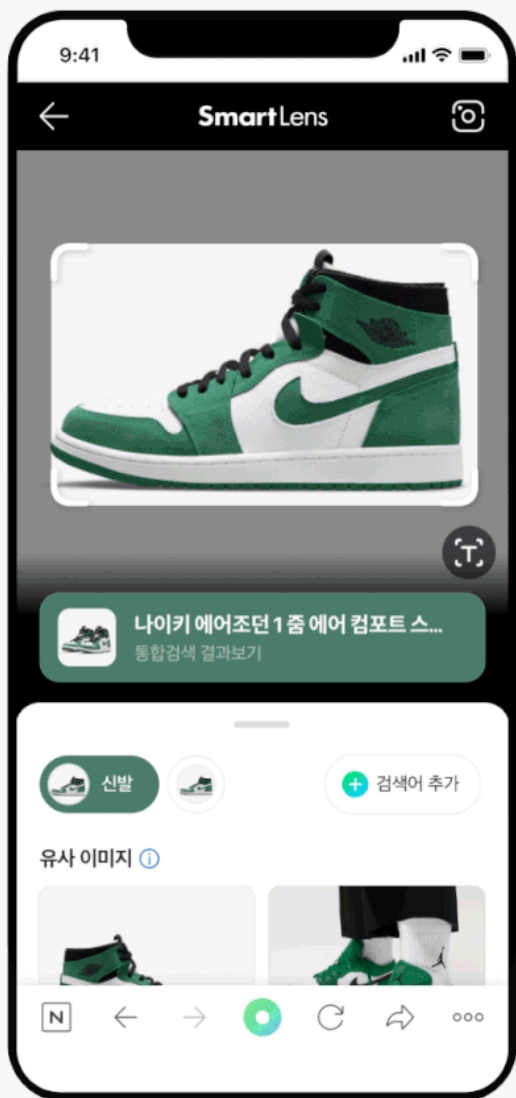
호수가 있는 여행지?



도시 풍경의 여행지?

#01 프로젝트 주제 선정 동기

네이버의 Omni-Search 기능



아이디어

착안

텍스트

이미지

텍스트와 이미지를 모두 활용해서
원하는 여행지의 정보를 추천해주자!

#02 프로젝트 진행 계획

< 3주 간의 계획 및 여정 >

2월 1일 ~ 7일	주제 선정	자료 조사	역할 분담
2월 8일 ~ 15일	데이터 수집	데이터 전처리	모델 개발
2월 16일 ~ 20일	성능 비교	프로토 타입	발표 자료

프로젝트 진행 과정



#01 데이터셋 구축 과정



















1) 이미지 데이터셋 구축

관광지 75개 선정 후

	A
1	Grand Canyon National Park
2	Giant's Causeway, United Kingdom
3	Bryce Canyon, USA
4	Taj Mahal India
5	Serengeti National Park
6	Machu Picchu Peru
7	Great Wall of China
8	Seongsan Ilchulbong Korea
9	Iguazu Falls Brazil
10	Amazon Rainforest Brazil
11	Ngorongoro Conservation Area
12	Angkor Wat Cambodia
13	The Palace Museum China
14	Karnak Egypt
15	Bora Bora French Polynesia
16	Acropolis Greece
17	Potala Palace China
18	Church of the Holy Sepulchre
19	Qin Shi Huang Tomb Terracotta Warrior China
20	El Castillo 97751 Tinum, Yucatan, Mexico
21	Petra Jordan
22	Easter Island Chile



각 관광지마다 이미지 3장씩 수집
data/db_imageset 폴더에 저장
-> 확장자 전부 png로 변경

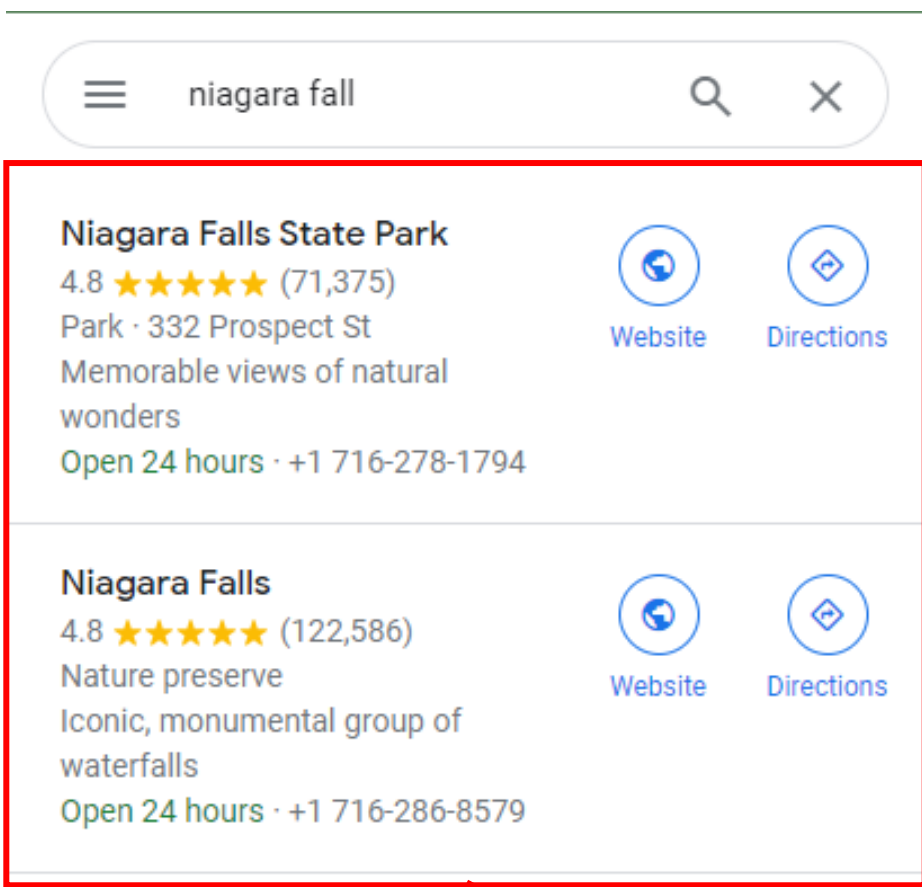
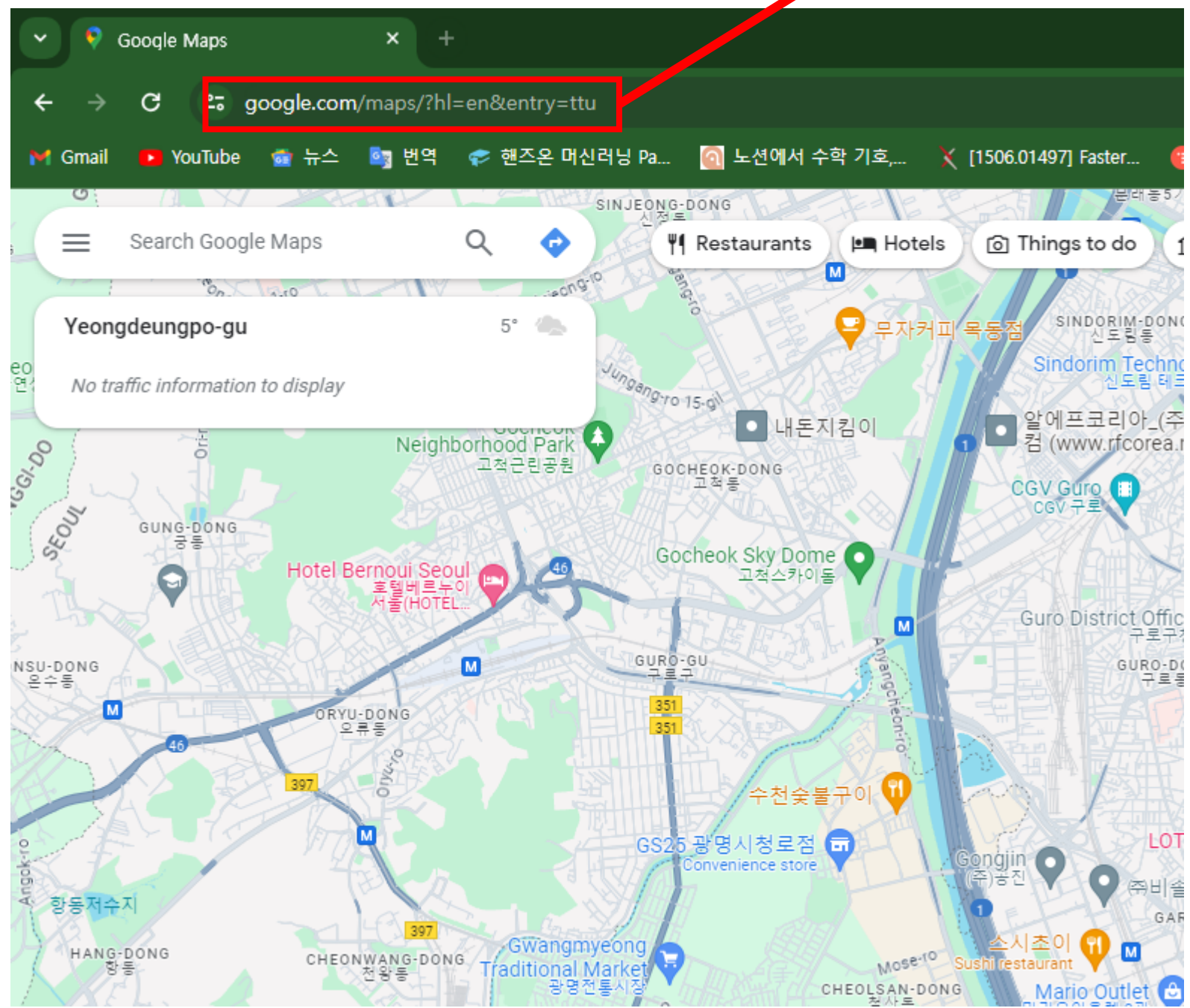
	Acropolis1.png	
	Acropolis2.png	
	Acropolis3.png	
	AlhambraGranada1.png	
	AlhambraGranada2.png	
	AlhambraGranada3.png	
	AmazonRainforest1.png	
	AmazonRainforest2.png	
	AmazonRainforest3.png	

#01 데이터셋 구축 과정

2) 텍스트 리뷰 데이터셋 구축

셀레니움 사용 -> 동적 크롤링

영문 review 위주로 수집하기 위해 Eng - Google map 도메인 네임 사용



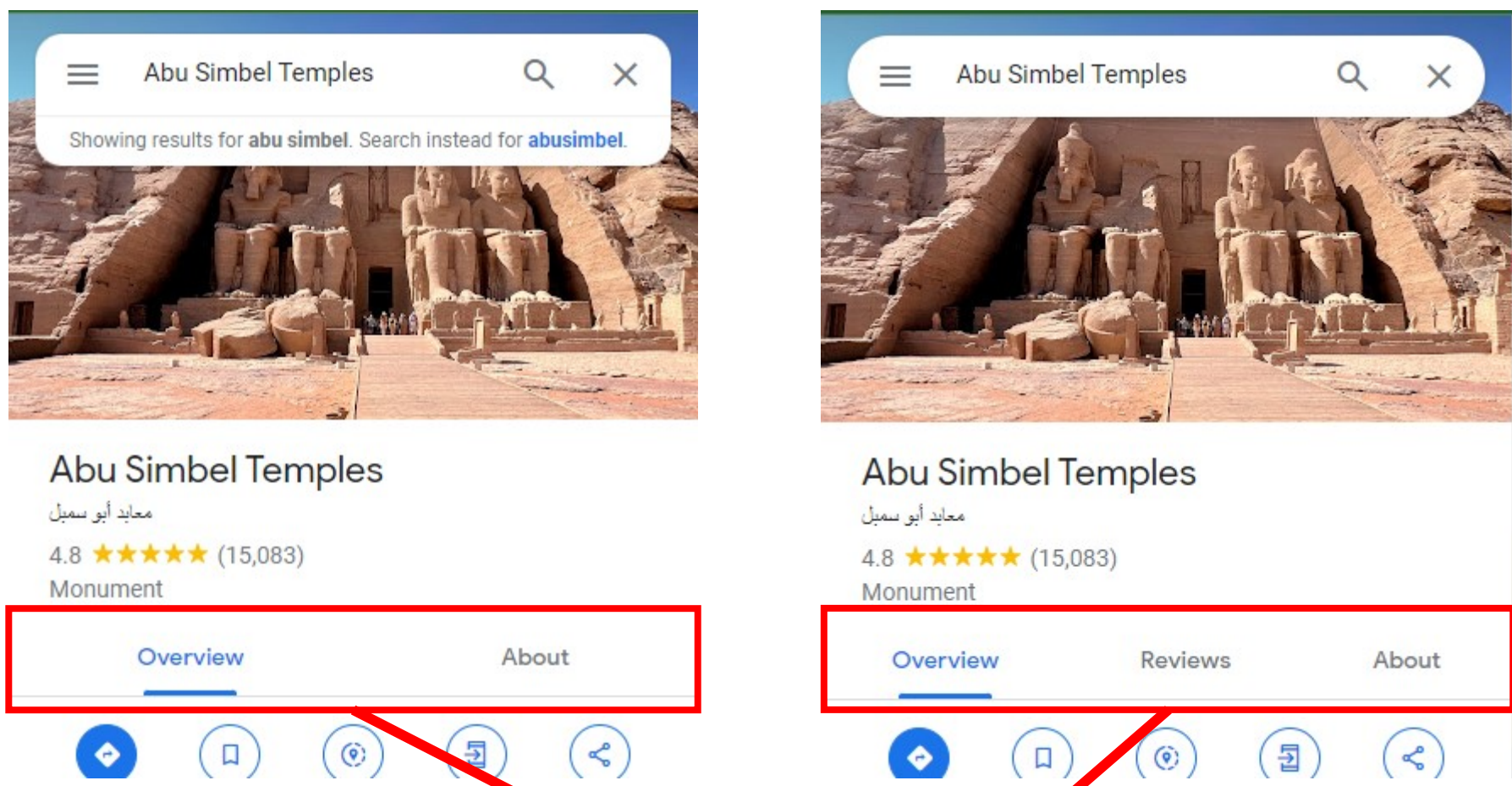
You've reached the end of the list.

2개 이상의 검색 결과가 나오지 않도록 하는 검색어 탐색

#01 데이터셋 구축 과정

2) 텍스트 리뷰 데이터셋 구축

셀레니움 사용 -> 동적 크롤링



Review 탭이 존재하는 검색 결과가 나오도록 하는 검색어 탐색



- X data_Abbaye du Mont-Saint-Michel.xlsx
- X data_Acropolis Greece.xlsx
- X data_Alcatraz Island San Francisco, CA 94133, United States.xlsx
- X data_Alhambra C. Real de la Alhambra, sn, Centro, 18009 Granada, ...
- X data_Amazon Rainforest Brazil.xlsx
- X data_Angel Falls Venezuela.xlsx
- X data_Angkor Wat Cambodia.xlsx
- X data_Baalbek Roman Ruins 2644 PCJ, Baalbek, Lebanon.xlsx
- X data_Banff National Park Improvement District No. 9, AB T0L, Cana...
- X data_Bora Bora French Polynesia.xlsx
- X data_Borobudur Temple.xlsx
- X data_British Museum England.xlsx
- X data_Bryce Canyon USA.xlsx

총 75개 여행지에 대해서
각각 500개의 리뷰 텍스트 스크래핑

#01 데이터셋 구축 과정

2) 텍스트 리뷰 데이터셋 전처리

 processed_txt

3

문장 기호 제거 + 소문자 처리 + 불용어 제거 + Nan을 'None' 으로 변환

 removed_emoji

2

한글 리뷰 및 이모티콘 제거

 test_imageset

 tour_review

1

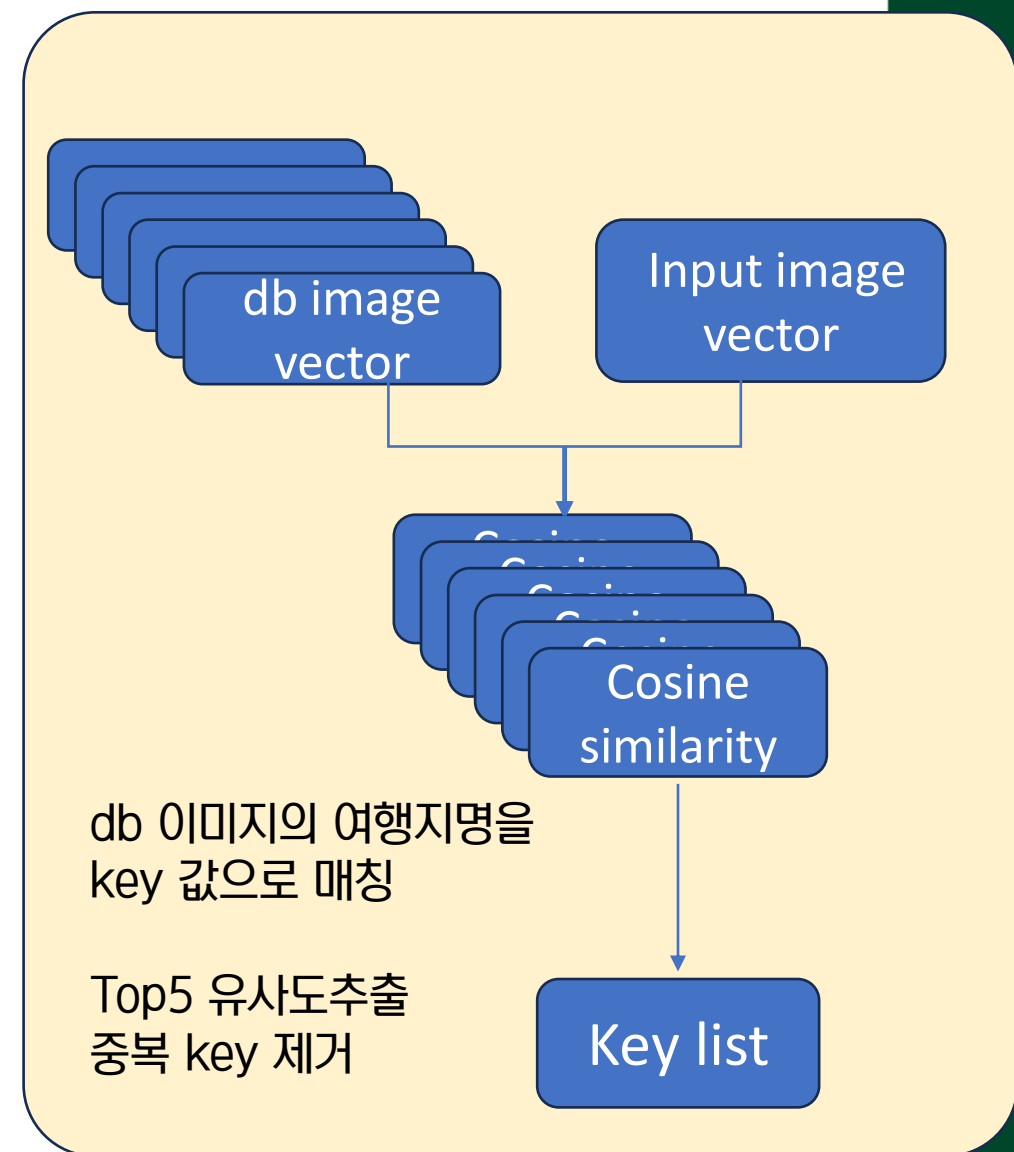
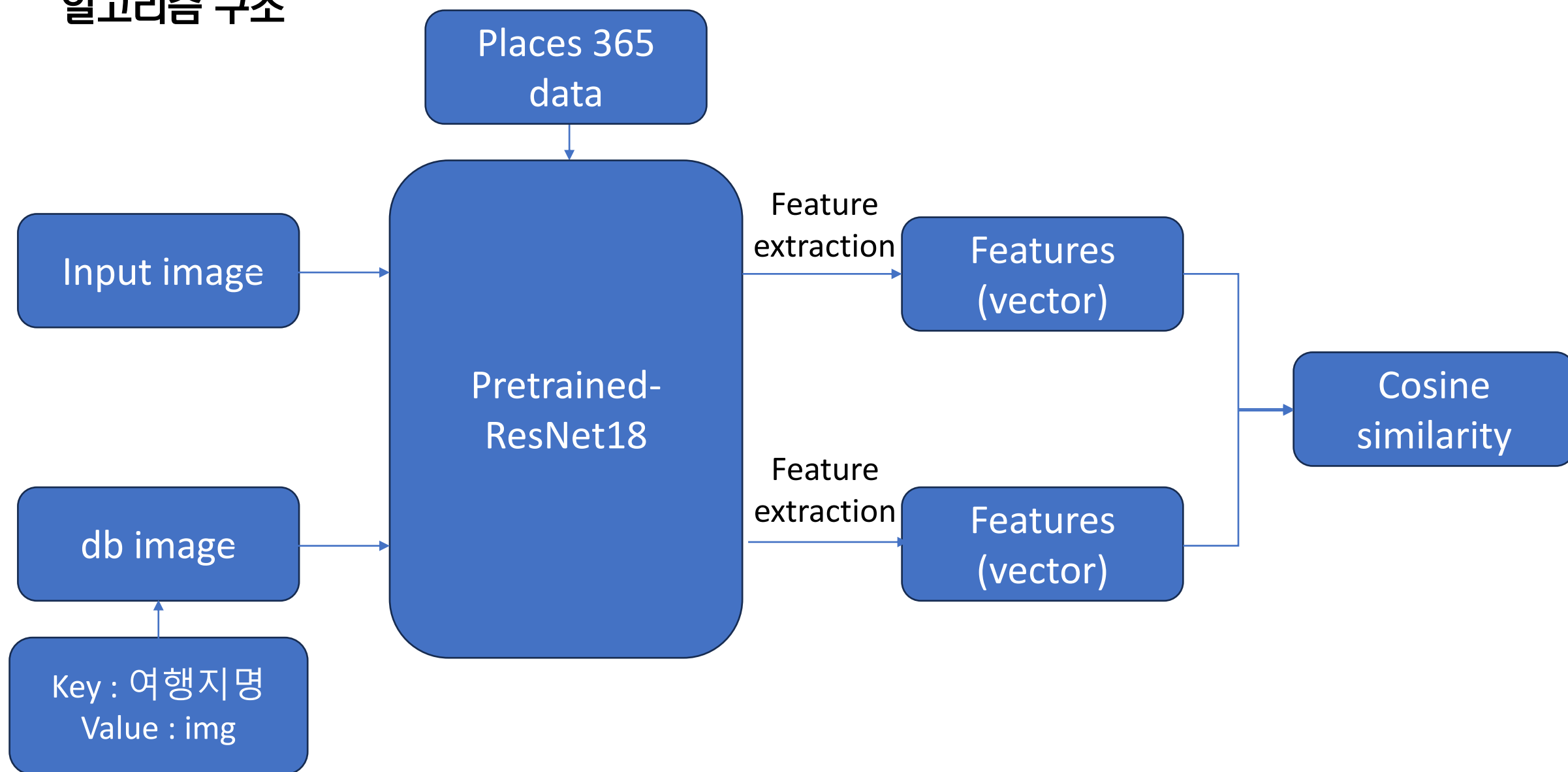
원본 데이터 파일 확장자 수정

 tour_review_gsheet

#02 모델 개발 과정

1) 이미지 유사도 검색 알고리즘 개발

알고리즘 구조



- db 이미지와 사용자 input image의 코사인 유사도 각각 연산 -> top5 유사한 이미지 추출
- Top5 이미지 key값 중 중복을 제거한 key값을 추천 여행지명 리스트로 출력

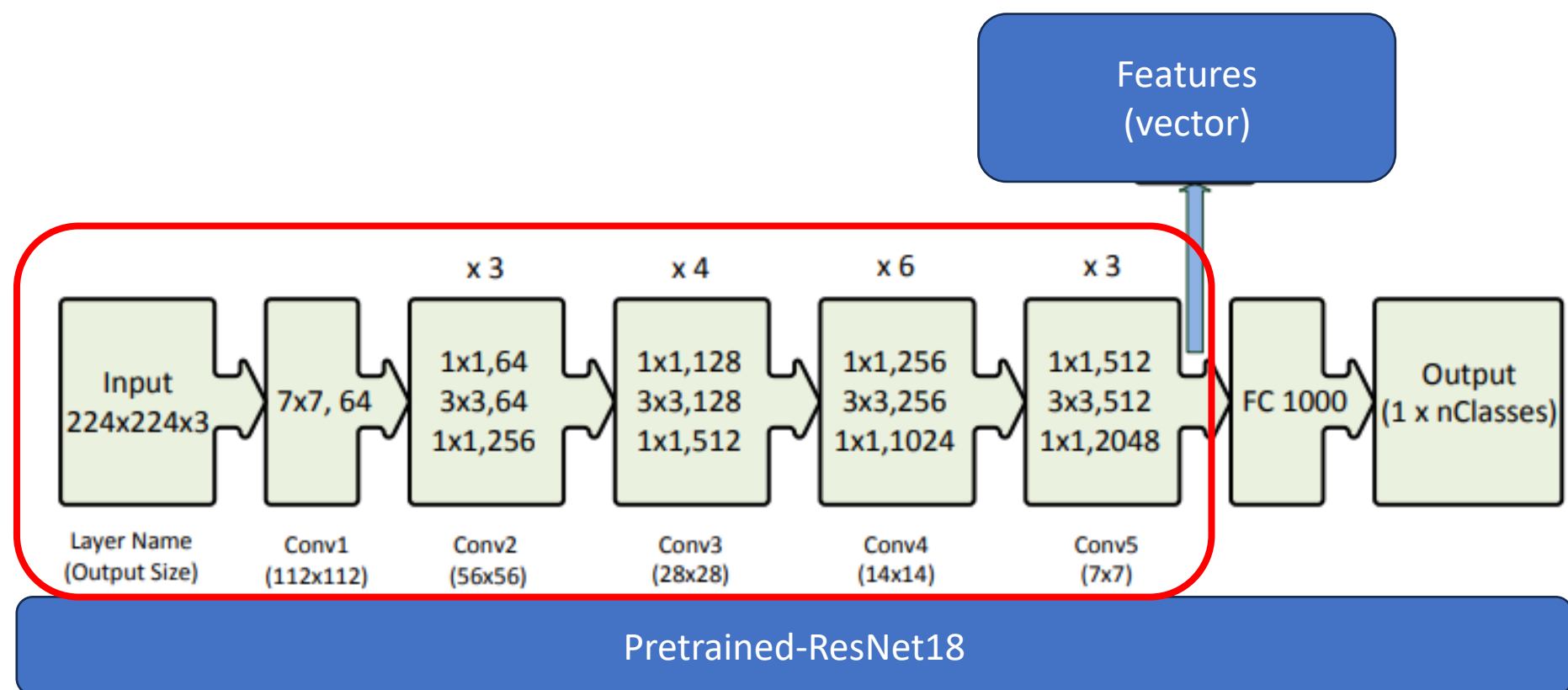
#02 모델 개발 과정

1) 이미지 유사도 검색 알고리즘 개발

Feature extraction

CNN 기반 모델의 FC레이어를 통과하기 전 feature vector을 사용해 코사인 유사도를 계산하는 방식

Resnet model의 FC레이어를 제거한 부분을
feature extractor 모델로 재정의



```
feature_extractor =  
torch.nn.Sequential(*list(model_resnet18.children())[:-1])
```

#02 모델 개발 과정 – 성능 엔지니어링

같은 Test image에 대해 각 base 모델 적용한 알고리즘 수행 후 general하게 가장 좋은 성능을 보이는 base 모델을 최종 모델로 선정

실험한 Base model

- Resnet50 + places 365 data
- Resnet18 + places 365 data
- Vit (Vision Transformer)
- Swin V2 Large

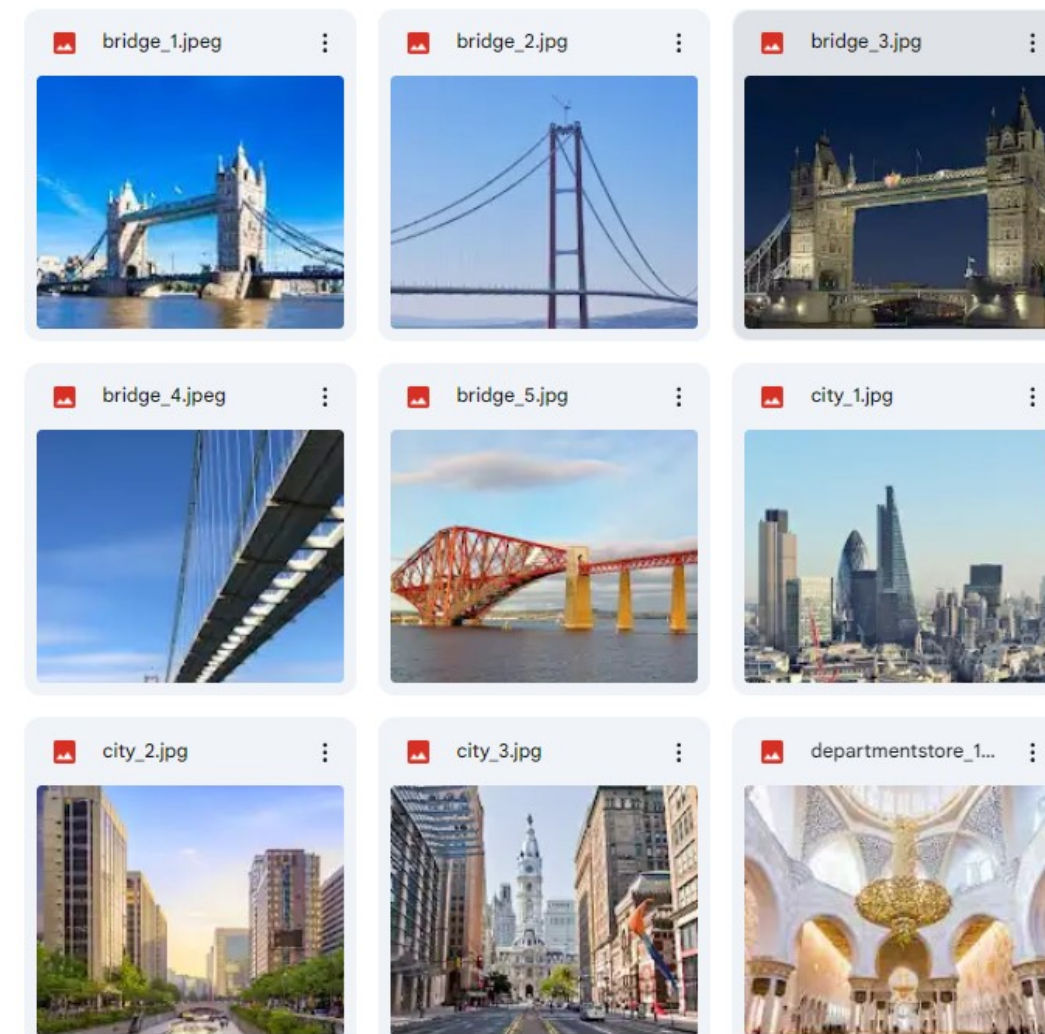
Places 365 data pretrained CNNs

Mit.edu에서 제공하는 places 365 dataset

- 총 365 class의 장소 이미지 제공
- CNN 모델 scene classification task 학습

Test imageset

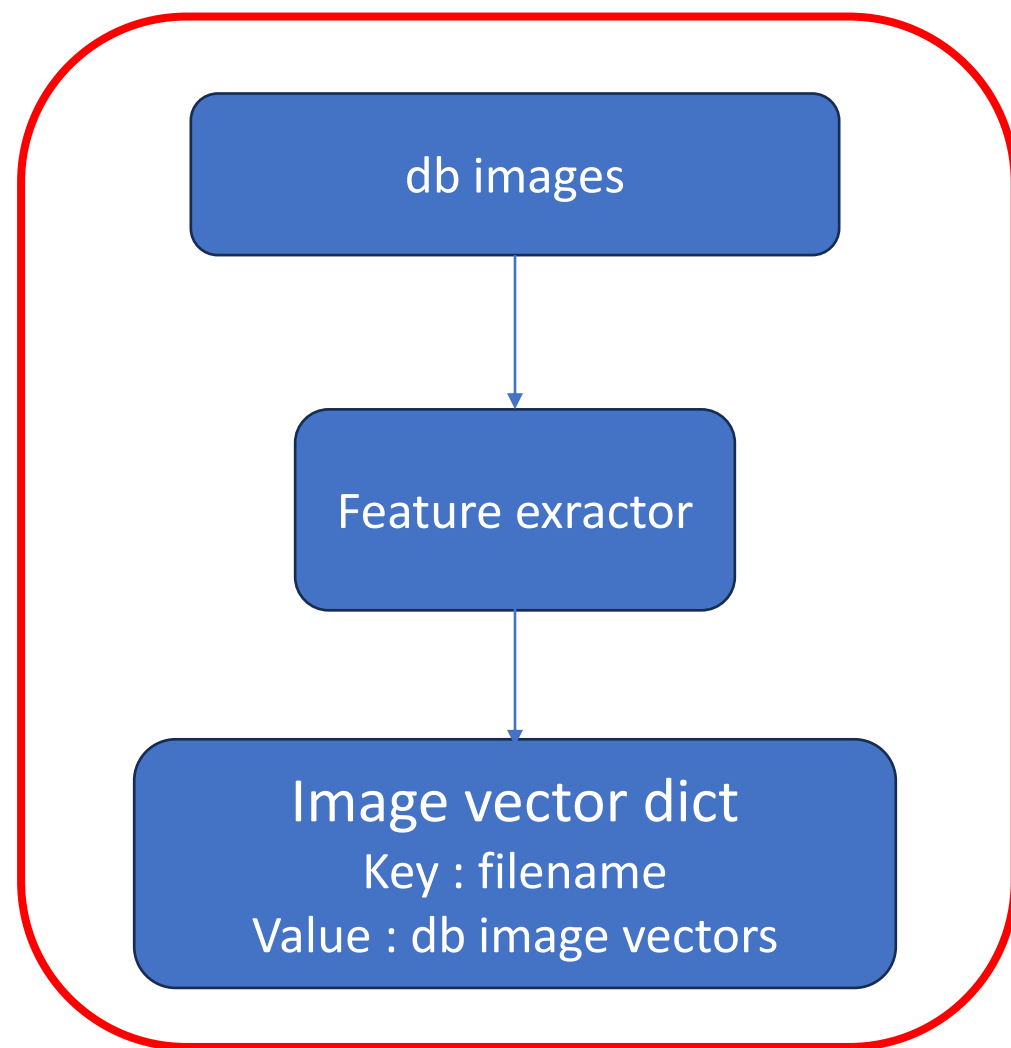
실제 사용자 서비스 시뮬레이션을 가정한 랜덤한 여행지 이미지셋
(구글 검색 사용, 15 class로 분류)



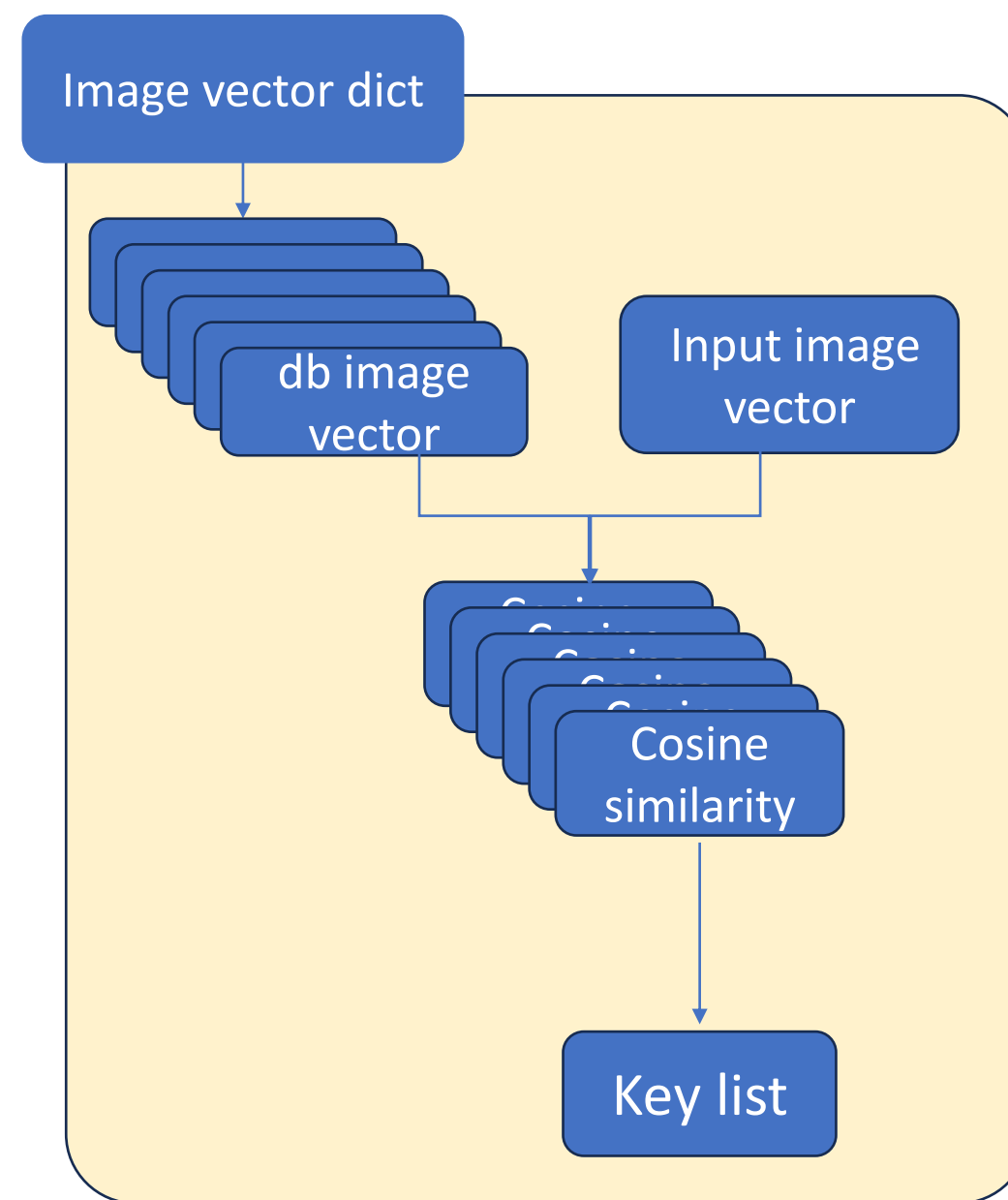
#02 모델 개발 과정 – 모델 최적화

- 각 db 이미지에 대한 feature vector를 미리 계산해둔 후 딕셔너리 형식으로 저장
- 사용자가 image input 시 미리 저장된 딕셔너리를 불러와 db 이미지와의 코사인 유사도를 계산 -> 서비스 시 연산시간 31.42초 절약

모든 db 이미지에 대해 미리 수행 :



사용자 서비스 시 수행 :



#02 모델 개발 과정 – 코드

1) ViT

```
!git clone https://github.com/lukemelas/ViT-PyTorch
```

```
# load the pre-trained weights
model = ViT('L_32_imagenet1k', pretrained=True)
model.eval()
```

2) Swin V2 Large

```
! pip install timm
```

```
# load the pre-trained weights
model = timm.create_model('swinv2_large_window12to24_192to384.ms_in22k_ft_in1k', pretrained=True)
model = model.to("cuda")
model.eval()
```


#02 모델 개발 과정 – 코드

3) ResNet 50 & 4) ResNet 18

```
def load_model(arch):  
    # device : gpu를 사용할 경우에는 'cuda', 그렇지 않을 경우에는  
    # th architecture to use  
    # load the pre-trained weights  
    model_file = '%s_places365.pth.tar' % arch  
    if not os.access(model_file, os.W_OK):  
        weight_url = 'http://places2.csail.mit.edu/models_places365/' + model_file  
        os.system('wget ' + weight_url)  
  
    model = models.__dict__[arch](num_classes=365)  
    checkpoint = torch.load(model_file, map_location=lambda storage, loc: storage)  
    state_dict = {str.replace(k, 'module.', ''): v for k, v in checkpoint['state_dict'].items()}  
    model.load_state_dict(state_dict)  
    model.eval()  
    return model
```

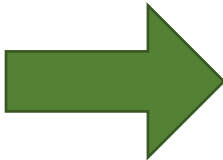
```
model_resnet18=load_model('resnet18')  
model_resnet50=load_model('resnet50')
```

Mit.edu의 places 365 pretrained 된 모델 가중치를 불러와 적용

#02 모델 개발 과정 – 성능 비교 결과



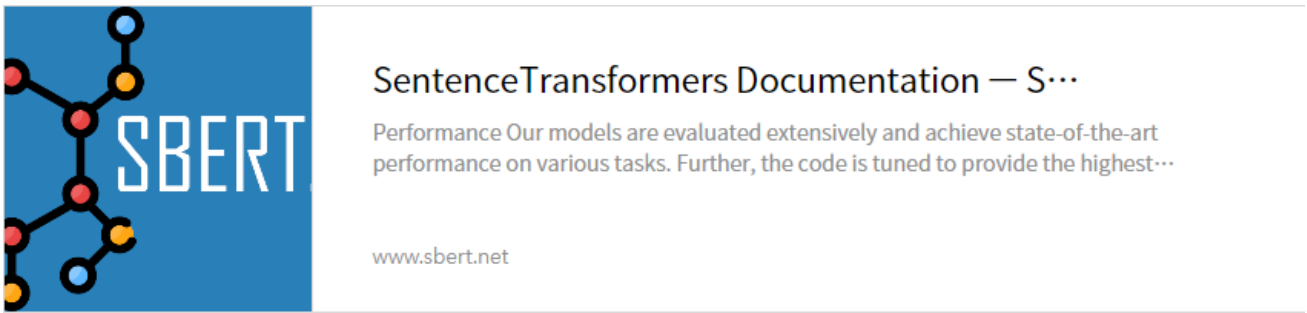
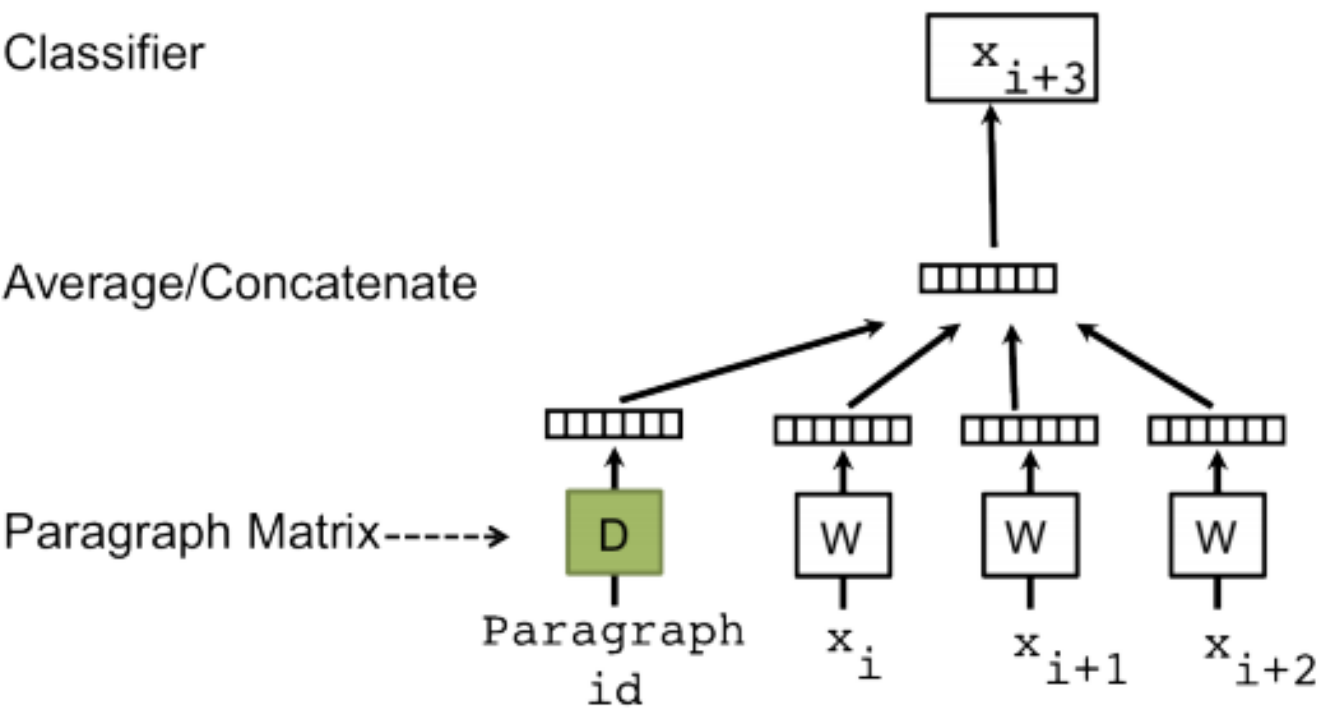
City image



Cosine Similarity	ViT	Swin V2 Large	Resnet 50	Resnet 18
1				
2				
3				
4				
5				

#02 모델 개발 과정

2) 텍스트 유사도 검색 알고리즘 개발



SBERT를 사용하는 것으로 개발 방향성 수정

Doc2Vec 사용 -> 무의미한 검색 결과

#02 모델 개발 과정

2) 텍스트 유사도 검색 알고리즘 개발



SentenceTransformers Documentation — S...

Performance Our models are evaluated extensively and achieve state-of-the-art performance on various tasks. Further, the code is tuned to provide the highest...

www.sbert.net

Cosine similarity 연산을 위한 pre-trained model 불러오기

Semantic Search 결과에 대한 정성적 평가 및 비교

```
print("Semantic Search Results")

for query, query_embedding in zip(queries, query_embeddings):
    distances = scipy.spatial.distance.cdist([query_embedding], sample_embeddings, "cosine")[0]

    results = zip(range(len(distances)), distances)
    results = sorted(results, key=lambda x: x[1])
    print("Query:", query)
    print("\nTop 5 most similar sentences in corpus:")

    for idx, distance in results[0:5]:
        print(sample_list[idx].strip(), "(Cosine Score: %.4f)" % (1-distance))
```

```
Semantic Search Results
Query: panoramic view

Top 5 most similar sentences in corpus:
['spectacular', 'views', 'worth', 'visit'] (Cosine Score: 0.7551)
['beautiful', 'view', '', ''] (Cosine Score: 0.7543)
['amazing', 'place', 'breathtaking', 'views'] (Cosine Score: 0.7490)
['beautiful', 'views'] (Cosine Score: 0.7487)
['stunning', 'views', 'city'] (Cosine Score: 0.7432)
['really', 'amazing', 'views', 'great', 'go', 'sunset', 'see', 'city', 'light', ''] (Cosine Score: 0.7398)
['amazing', 'place', 'beautiful', 'stunning', 'views'] (Cosine Score: 0.7394)
['incredible', 'view', 'nice', 'place'] (Cosine Score: 0.7362)
['amazing', 'place', 'breathtaking', 'view', 'must', 'see'] (Cosine Score: 0.7331)
['amazing', 'views'] (Cosine Score: 0.7265)
['great', 'views', 'cool', 'place', 'watch', 'sunset'] (Cosine Score: 0.7194)
['incredible', 'views', 'well', 'maintained', 'place'] (Cosine Score: 0.7074)
['views', 'unparallel', 'breathtaking'] (Cosine Score: 0.7055)
['enjoyable', 'view'] (Cosine Score: 0.7041)
['nice', 'place', 'nice', 'views'] (Cosine Score: 0.6985)
['amazing', 'experience', 'view', 'landscape', 'rio', 'amazing'] (Cosine Score: 0.6949)
['awesome', 'views', 'city', 'rio', 'top', 'must', 'see', 'tourists', 'area'] (Cosine Score: 0.6911)
['amazing', 'views', 'totally', 'worth', 'visiting'] (Cosine Score: 0.6881)
['amazing', 'experience', 'breathtaking', 'view', 'rio'] (Cosine Score: 0.6877)
['awesome', 'experience', 'worth', 'beautiful', 'view', 'beautiful', 'city'] (Cosine Score: 0.6854)
```

#02 모델 개발 과정

2) 텍스트 유사도 검색 알고리즘 개발

Performance 상위 5개 모델의 낮은 성능 결과

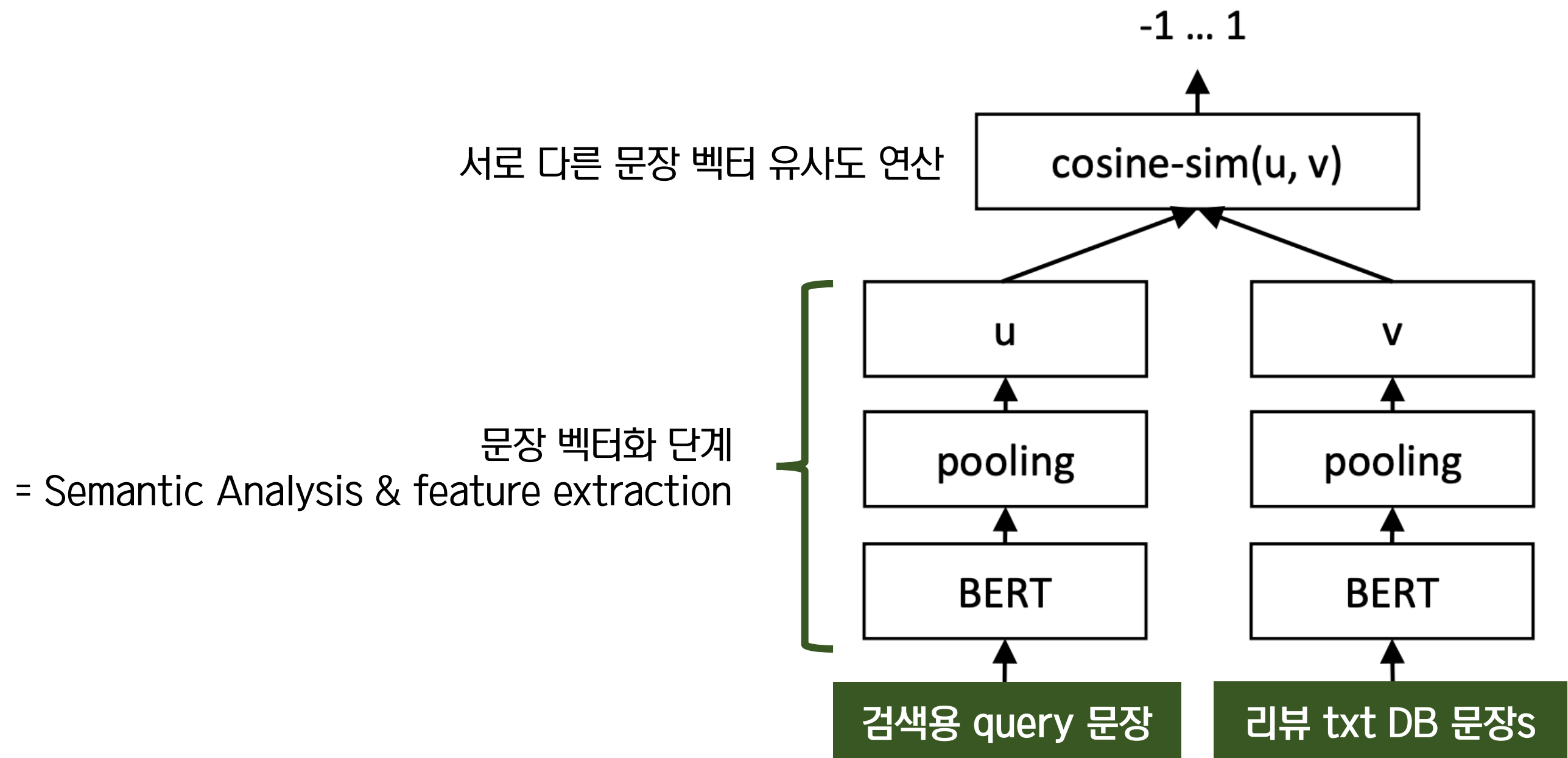
-> 상대적으로 짧은 리뷰 텍스트 대응에 있어서 오버 스펙인 것으로 추정

Model Name	Performance Sentence Embeddings (14 Datasets) ⓘ	Performance Semantic Search (6 Datasets) ⓘ	⚡ Avg. Performance ⓘ	Speed ⓘ	Model Size ⓘ
all-mpnet-base-v2 ⓘ	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1 ⓘ	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1 ⓘ	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2 ⓘ	68.70	50.82	59.76	7500	120 MB
multi-qa-distilbert-cos-v1 ⓘ	65.98	52.83	59.41	4000	250 MB
all-MiniLM-L6-v2 ⓘ	68.06	49.54	58.80	14200	80 MB
multi-qa-MiniLM-L6-cos-v1 ⓘ	64.33	51.83	58.08	14200	80 MB
paraphrase-multilingual-mpnet-base-v2 ⓘ	65.83	41.68	53.75	2500	970 MB
paraphrase-albert-small-v2 ⓘ	64.46	40.04	52.25	5000	43 MB
paraphrase-multilingual-MiniLM-L12-v2 ⓘ	64.25	39.19	51.72	7500	420 MB
paraphrase-MiniLM-L3-v2 ⓘ	62.29	39.19	50.74	19000	61 MB
distiluse-base-multilingual-cased-v1 ⓘ	61.30	29.87	45.59	4000	480 MB
distiluse-base-multilingual-cased-v2 ⓘ	60.18	27.35	43.77	4000	480 MB

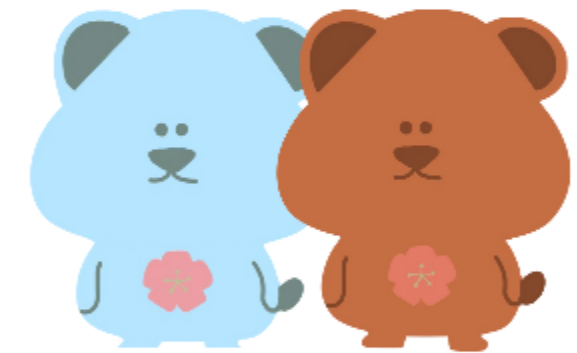
=> Roberta base의
비교적 단순한 문장 유사도 비교 모델 사용 결정

#02 모델 개발 과정

2) 텍스트 유사도 검색 알고리즘 개발



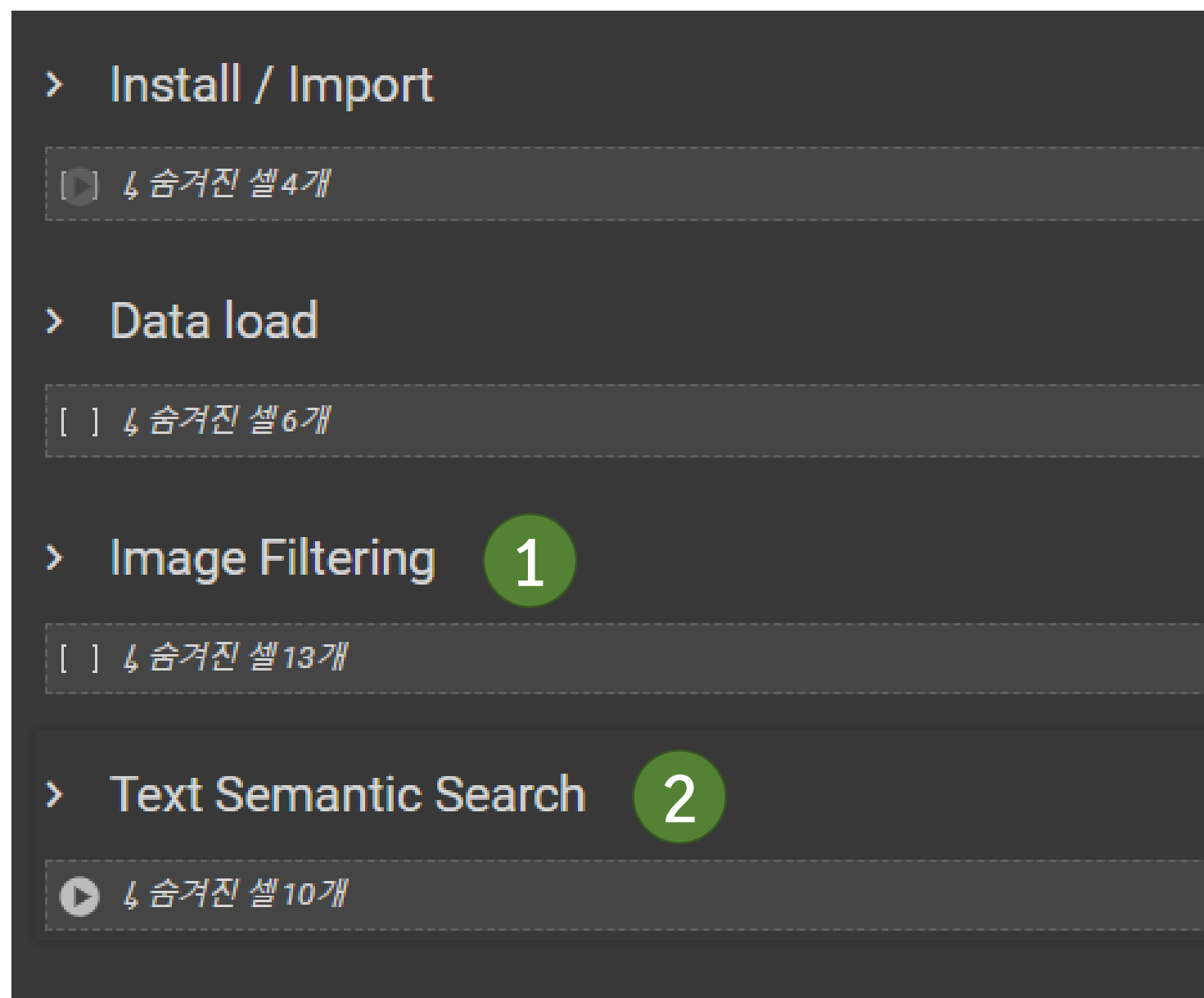
프로젝트 결과



#01 프로토타입 시연 결과

- 프로토 타입 Colab 노트북 구성

Github를 통해 검색 알고리즘 실습용 오픈소스로 배포 예정



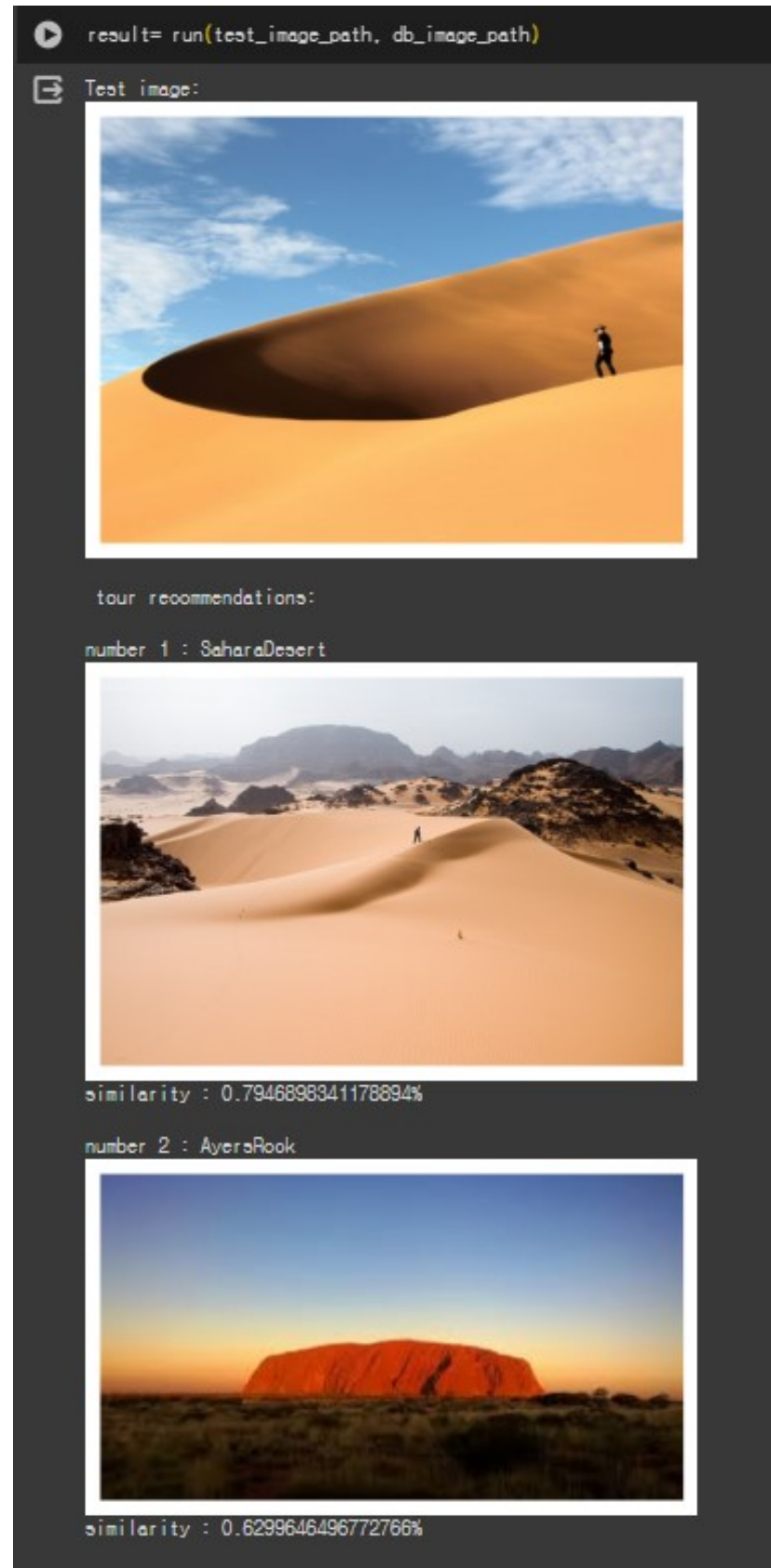
필요한 library 등 설치
및
Data 불러오기

이미지 검색 기반
1차 여행지 정보 filtering

Filtering 된 여행지 리뷰 DB내 query 기반 검색
→ 원하는 리뷰 감상 존재 여부 확인

#01 프로토타입 시연 결과

1 이미지 검색 기반 1차 여행지 정보 filtering



2 Filtering 된 여행지 리뷰 DB내 query 기반 검색 → 원하는 리뷰 감상 존재 여부 확인

```
Semantic Search Results
Query: recommend to go

Top 5 most similar sentences in corpus:
['havent', 'came', 'wanna', 'go', 'sometime'] (Cosine Score: 0.8176)
['saw', 'sand', '1010', 'would', 'recommend'] (Cosine Score: 0.7384)
['dream', 'visit', ''] (Cosine Score: 0.7367)
['hot', 'dont', 'recommend', 'going', 'place'] (Cosine Score: 0.7283)
['nice', 'hope', 'starts', 'expanding', 'everyone', 'enjoy'] (Cosine Score: 0.7282)
Query: recommend to go

Top 5 most similar sentences in corpus:
['experience', 'like', '100', 'recommended'] (Cosine Score: 0.7783)
['great', 'experience', 'highly', 'recommended', 'see', ''] (Cosine Score: 0.7781)
['fabulous', 'recommend', 'guided', 'tours', 'better', 'experience'] (Cosine Score: 0.7610)
['awesome', 'place', 'must', 'go', 'havent', ''] (Cosine Score: 0.7489)
['amazing', 'absolutely', 'worth', 'coming', 'visit'] (Cosine Score: 0.7488)
```

#03 보완점 및 기대 효과

- 데이터 다양성 및 품질 향상
 - 각 관광지를 더 잘 표현할 수 있는 DB 이미지 데이터 수집
 - 텍스트 유사도 검색을 위한 더 의미있는 텍스트 데이터 수집 필요
- 관광지 추천 알고리즘 성능 개선
 - 이미지 유사도 검색 이후 텍스트 유사도 검색 결과도 추가로 반영해 관광지 추천 리스트 최종 결정
 - 관광지 리스트를 75개보다 더 확장시켜 다양한 장소 추천

#03 기대효과

- 개인화된 추천 기능

이미지와 텍스트를 모두 사용함으로써 사용자의 구체적인 요구와 선호를 더 정확히 파악하여 더욱 개인화된 관광지 추천 가능

- 새로운 관광지 발견

다양하고 풍부한 데이터와 고도화된 추천 알고리즘을 통해 사용자가 알지 못했던 새로운 관광지를 발견

THANK YOU

