2023.03.17 미팅 자료

bert-as-service, cosine-similarity, Kochat, es-gpt

팀원 : 배한성, 김은서, 조현아, 조유진

CONTENTS

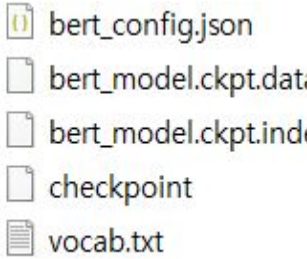
bert-as-service

bert-as-service

- Bert-as-Service에서 필요한 모델 파일은 .ckpt.index, .ckpt.meta, .ckpt.data, config.json, vocab.txt
  - 즉, 5개의 파일이 필요하다.

- .ckpt 파일은 모델 학습 과정에서 체크포인트를 저장하면 저장할 수 있었다.
  - 하지만 .ckpt.meta는 tensorflow 2.x 버전에서는 저장할 수 없었다. (관련 메소드가 존재하지 않음)
  - .ckpt.meta는 그래프의 노드가 저장된 파일임을 확인하였다.
  - tensorflow 2.x 버전에서 그래프의 노드를 저장하는 함수로는 .pbtxt확장자만 저장할 수 있었음

- 따라서 tensorflow 1.x에서 학습을 시켜야한다는 것을 알았다.
  - 하지만 pre-training된 BERT 모델을 tensorflow의 모델로 가져올려면 transformers의 모듈인 TFBertForSequenceClassification 를 사용해야함
  - 위 모듈은 tensorflow 2.x의 모델을 로드하기 때문에 tensorflow 1.x로 학습할 수 없었다.



- config.json과 vocab.txt는 tensorflow2.x의 메소드로 얻을 수 있었다.

bert-as-service

- Bert-as-Service가 제공하는 embedding Vector로 Cosine Similarity를 구하면 특정 값 이상일 때 A와 B는 비슷한 것을 알 수 없다.
  - Bert-as-Service가 제공한 embedding Vector로 Cosine Similarity를 구하면 항상 0.8 이상의 값이 계산된다.

- 따라서 튜토리얼과 GitHub에서는 문장 A와 문장 B, 문장 C의 embedding Vector를 추출하여 A와 B의 유사도, A와 C의 유사도를 계산한 후 비교하여 어느 문장 쌍이 더 유사한지 구하는 방법을 추천하고 있다.

The cosine similarity of two sentence vectors is unreasonably high (e.g. always > 0.8), what's wrong?

A decent representation for a downstream task doesn't mean that it will be meaningful in terms of cosine distance. Since cosine distance is a linear space where all dimensions are weighted equally. if you want to use cosine distance anyway, then please focus on the rank not the absolute value. Namely, do not use:

```
if cosine(A, B) > 0.9, then A and B are similar
```

Please consider the following instead:

```
if cosine(A, B) > cosine(A, C), then A is more similar to B than C.
```

bert-as-service

- 공식 문서에서 제공하는 multi_cased_L-12_H-768_A-12 모델을 사용해 Cosine Similarity를 구한 예
- Cosine Similarity는 scikit-learns의 cosine_similarity함수를 이용
- 특정 수치 이상이면, 챗봇의 경로를 정해야하는 우리 프로젝트에서는 적합하지 않음

```
bc = BertClient()
vec=bc.encode(['상품을 추천해주세요','노트북 리뷰를 요약해줘','컴퓨터 정보 알려줘'])
```

```
(keybert) C:₩keybert>python serviceTest.py
sklearn
0번째와 1번째가 유사도가 더 높습니다.
0번째 1번째 유사도 :  [[0.98375994]]
0번째 2번째 유사도 :  [[0.9837599]]
```

cosine similarity

cosine similarity

- Classification을 위한 BERT 모델(BertForSequenceClassification)을 통해 Cosine Similarity 계산

```
model=BertForSequenceClassification.from_pretrained('bert-base-multilingual-cased',output_hidden_states = True)
model_file = "/content/drive/MyDrive/model/model_final.pt"  # 테스트 진행할 모델.pt
model.load_state_dict(torch.load(model_file))

model.eval()
```

```
token_vecs1 = hidden_states1[-3][0] #classification하는 layer로 인해 -3번째 layer를 사용
token_vecs2= hidden_states2[-3][0]
token_vecs3 = hidden_states3[-3][0]

sentence_embedding1 = torch.mean(token_vecs1, dim=0).numpy()
sentence_embedding2 = torch.mean(token_vecs2, dim=0).numpy()
sentence_embedding3 = torch.mean(token_vecs3, dim=0).numpy()

print(sentence_embedding1.shape)
```

cosine similarity

- Classification을 위한 BERT 모델(BertForSequenceClassification)을 통해 Cosine Similarity 계산

```
token_vecs1 = hidden_states1[-3][0] #classification하는 layer로 인해 -3번째 layer를 사용
token_vecs2= hidden_states2[-3][0]
token_vecs3 = hidden_states3[-3][0]

sentence_embedding1 = torch.mean(token_vecs1, dim=0).numpy()
sentence_embedding2 = torch.mean(token_vecs2, dim=0).numpy()
sentence_embedding3 = torch.mean(token_vecs3, dim=0).numpy()

print(sentence_embedding1.shape)
```

cosine similarity

- 가장 유사한 1, 3번의 문장의 코사인 유사도값이 이상했음

```
from sklearn.metrics.pairwise import cosine_similarity

# text1 = "성능 괜찮은 노트북 추천해줘"
# text2 = "음료수 뭐마실까"
# text3 = "노트북 하나 추천해줘"
cos1 = cosine_similarity([sentence_embedding1], [sentence_embedding2])
cos2 = cosine_similarity([sentence_embedding1], [sentence_embedding3])
cos3 = cosine_similarity([sentence_embedding2], [sentence_embedding3])

print('Vector similarity 1 to 2 for :  %.2f' % cos1)
print('Vector similarity 1 to 3 for :  %.2f' % cos2)
print('Vector similarity 2 to 3 for :  %.2f' % cos3)
```

```
Vector similarity 1 to 2 for :  0.08
Vector similarity 1 to 3 for :  -0.06
Vector similarity 2 to 3 for :  0.83
```

cosine similarity

- keyBERT에서 사용할 SBERT를 이용해 Cosine Similarity 계산

```
from sentence_transformers import SentenceTransformer, util

model = SentenceTransformer('jhgan/ko-sbert-multitask') #jhgan/ko-sbert-multitask
```

```
sentences=["성능 괜찮은 노트북 추천해줘","노트북좀 요약해봐", "음료수 뭐마실까", "노트북 하나 추천해줘"]
sentence_embeddings = model.encode(sentences)
```

cosine similarity

- 앞서 사용한 Classification 모델에 비해 값이 잘 나온 것을 보여준다.
- 특정 수치 이상이면 챗봇의 경로를 설정할 수 있을 것으로 예상

```
from sklearn.metrics.pairwise import cosine_similarity
# "성능 괜찮은 노트북 추천해줘","노트북좀 요약해봐", "음료수 뭐마실까", "노트북 하나 추천해줘"
# 0번째 문장과 1, 2, 3번째 문장의 유사도를 계산
cosine_similarity([sentence_embeddings[0]], sentence_embeddings[1:])
```

```
array([[0.7875458 , 0.18199414, 0.87817734]], dtype=float32)
```

cosine similarity

- 4개의 문장을 모두 비교해본 결과

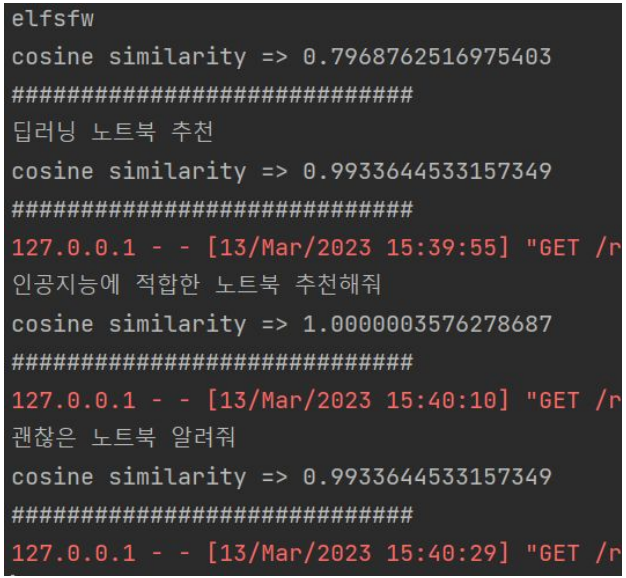
```
similarities = util.cos_sim(sentence_embeddings, sentence_embeddings)
# 유사도의 값을 행렬로 표시    문장 번호  0   1   2   3
print(similarities) #                  0
                    #                  1
                    #                  2
                    #                  3
```

```
tensor([[1.0000, 0.7875, 0.1820, 0.8782],
        [0.7875, 1.0000, 0.1950, 0.8147],
        [0.1820, 0.1950, 1.0000, 0.2187],
        [0.8782, 0.8147, 0.2187, 1.0000]])
```

Kochat

Kochat

- “딥러닝에 적합한 노트북 추천”과 Kochat 데모 페이지에서 사용자가 입력한 문장 비교
  - **bert-as-service**를 통해 문장 임베딩을 구한 후 cosine similarity 구하기



Kochat

- “딥러닝에 적합한 노트북 추천”과 Kochat 데모 페이지에서 사용자가 입력한 문장 비교
  - SBERT를 통해 문장 임베딩을 구한 후 cosine similarity 구하기

```
sentences = [sequence, "딥러닝에 적합한 노트북 추천"]
sentence_embeddings = model.encode(sentences)
cosine_sim = cosine_similarity([sentence_embeddings[0]], sentence_embeddings[1:])
```

```
인공지능에 적합한 노트북 추천해줘
tensor([[0.8031]])
127.0.0.1 - - [16/Mar/2023 17:58:28] "
127.0.0.1 - - [16/Mar/2023 17:58:40] "
인공지능할만한 노트북 추천해줘
tensor([[0.7469]])
127.0.0.1 - - [16/Mar/2023 17:58:49] "
프로그래밍 할만한 노트북 추천해줘
tensor([[0.7160]])
```

```
딥러닝할만한 노트북 추천해줘
tensor([[0.9261]])
127.0.0.1 - - [16/Mar/2023 17:59:03] "
127.0.0.1 - - [16/Mar/2023 17:59:07] "
딥러닝할만한 노트북 알려줘
tensor([[0.8482]])
127.0.0.1 - - [16/Mar/2023 17:59:18] "
영상편집할만한 노트북 추천해줘
tensor([[0.6289]])
```

es-gpt

es-gpt란

- Elasticsearch + GPT3 Answer
- Elasticsearch 결과를 가로채 GPT3로 전송하여 사용자의 쿼리에 대한 정확하고 적절한 답변을 제공하는 프로그램

- Elasticsearch란?
  - Apache Lucene에 구축되어 배포된 검색 및 분석 엔진
  - JSON 문서 형식의 데이터를 Logstash와 같은 수집 도구나 API를 사용해 Elasticsearch로 전송
    Elasticsearch는 자동으로 원래 문서를 저장하고 클러스터의 인덱스 문서에 대한 검색 가능한 참조를 추가

  -> 문서를 검색하고 조회 가능

- 문제점
  - 사용자가 입력한 요청 문장에 대한 의도와 필요한 키워드를 추출하고
    추출한 키워드를 통해 db에서 검색 및 가공해야하는 본 프로젝트의 기능과는 다른 API로 생각됨.