BERT as Service

사전 지식

- Sentence Encoding/Embedding은 감정 분석, 텍스트 분류와 같은 작업을 할 때 필요한 것이다.
    - 가변 길이 문장을 고정 크기의 벡터로 표현하는 것
    - 벡터의 각 요소는 원래 문장의 일부 의미를 담고 있다.
- BERT as Service는 BERT를 문장 인코더로 사용하고 ZeroMQ를 통해 단, 두 줄의 코드로 문장을 고정 길이 표현으로 매핑할 수 있다.
    - ZeroMQ는 비동기 메시징 라이브러리이다.
    - ZeroMQ는 메시지 큐를 제공하지만 전용 메시지 브로커 없이 동작이 가능
- BERT as Serivce는 12/24 Layer의 BERT 모델을 기반으로 한다.

시작 방법

- python >=3.5, tensorflow >= 1.10 에서 사용 가능
- 서버 설치 후, 서버 실행
    - bert-serving-start -model_dir <model의 위치> -max_seq_len <최대 문장 길이> -num_worker <동시 요청 수>
- 클라이언트에서 문장 인코딩 가져오기

```
from bert_serving.client import BertClient
bc = BertClient()
vec = bc.encode(['First do it', 'then do it right', 'then do it better'])
```

세부 설명

- 문장 벡터의 크기
    - 각 문장은 768차원의 벡터로 변환된다.
    - 단, fine-tuning된 BERT의 pooling층에 따라 다를 수 있다.
- 고정 크기 벡터
    - 고정 크기 벡터를 얻을 때 기본적으로 마지막에서 두 번째 Layer의 output을 평균 Pooling하여 얻는다.
    - 기본적으로 마지막에서 두 번째 Layer를 이용하는 이유
        - 마지막 Layer는 사전 훈련 중 학습 방법(마스킹된 단어 예측 등)에 따라 값이 편향될 수 있다.
        - 마지막 Layer를 사용하려면 pooling_layer값을 -1로 두면 된다. (마지막 레이어는 파인튜닝에 따라 사용 가능할 수 있다.)
        - 첫번째 Layer를 사용하면 원래 단어 정보를 보존할 수 있다.(셀프 어텐션을 사용하지 않은 값)
        - 따라서 첫번째 ~ 마지막 레이어의 선택은 트레이드 오프이다.
- BEER as Service가 동시에 처리할 수 있는 요청 수
    - 최대 동시 요청 수는 num_worker의 값에 따라 달라진다.
    - 요청 수가 num_worker보다 많으면 대기 큐에서 기다렸다가 작업을 수행한다.

    - 요청 수는 문장의 수가 아닌 클라이언트에서 보낸 문장 목록을 의미한다.
    - num_worker의 수는 보유한 GPU/CPU의 보다 적거나 같아야한다.
        - 그렇지 않으면 여러 작업이 하나의 GPU/CPU에 할당되어 메모리 부족 현상이 발생할 수 있다.
- 여러개의 Layer 사용
    - 서버를 실행할 때 pooling_layer에 사용할 레이어를 알려주면 된다.
        - -pooling_layer -4 -3 -2 -1
- 사용가능한 pooling 방법
    - NONE
        - pooling이 전혀 없으며 문장 임베딩 대신 단어 임베딩을 사용하는 경우 사용
    - REDUCE_MEAN
        - 인코딩 layer의 output에 평균 pooling을 취한다.
    - REDUCE_MAX
        - 인코딩 layer의 output에 최대값 pooling을 취한다.
    - REDUCE_MEAN_MAX
        - 평균 pooling과 Max pooling을 개별적으로 수행한 후 두개의 벡터를 연결하여 1536 차원의 문장 임베딩을 생성한다.
    - CLS_TOKEN or FIRST_TOKEN
        - [CLS] 토큰을 얻는다.
    - SEP_TOKEN or LAST_TOKEN
        - [SEP] 토큰을 얻는다.
- [CLS] 토큰을 디폴트로 사용하지 않는 이유
    - 사전 학습된 모델은 fine-tuning되지 않아 [CLS] 토큰은 좋은 문장 임베딩을 가지지 않는다.
    - 나중에 모델을 fine-tuning하면 사용할 수 있다.
- fine-tuning된 BERT 모델 사용 방법
    - model 디렉토리 하위에 bert_model.ckpt(사전 훈련된 가중치가 포함된 tensorflow 체크 포인트), vocab.txt (WordPiece를 단어 ID에 매핑하기 위한 vocab 파일). bert_config.json(모델의 하이퍼파라미터를 지정하는 구성 파일)
- 코사인 유사도를 구하는 방법
    - 코사인 유사도는 모든 차원에 동일한 가중치가 적용되는 선형 공간
    - 코사인 유사도를 구하려면 절대 값이 아닌 순위에 포커스를 두어야한다.

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