**tweepy.py**

텍스트이(가) 표시된 사진

자동 생성된 설명

twitter API로 발급받은 consumer key, access token을 입력하여 사용

Auth handler, Access request, api instance creation을 진행한 뒤

coin\_list\_string에 검색할 키워드를 OR을 활용하여 이어 하나의 문자열로 만든다.

위 코드의 경우 bitcoin 또는 btc를 포함하는 트윗중 리트윗을 제외한 트윗을 검색

tweepy.Cursor로 설정한 개수만큼의 조건에 맞는 트윗을 검색해, 본문에 설정한 키워드가 포함되어있는지 확인한 뒤(검색시 본문에 해당 내용이 없어도 사용자 이름에 해당 키워드가 포함되거나, 기타 이유 등으로 섞여들어올 가능성이 있음)

해당 트윗이 포함하는 각종 정보를 저장해 tweet\_list에 추가한다.

이렇게 수집한 트윗 내용을 dataframe으로 저장한다.

**dataset\_process.py**

광고 학습 데이터셋 생성에 사용

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

트윗 내용을 분석하기 쉽게 전처리해줌

광고 데이터 학습을 위한 데이터셋 생성의 경우 url, emoji 두가지 제거 후 multi spacing을 제거하여 진행했음

텍스트이(가) 표시된 사진

자동 생성된 설명

광고 학습 데이터셋 생성중 지나치게 반복적으로 포함되는 특정 형식 트윗이 있어 학습시 편향이 우려되어 데이터셋 생성 과정에서 지나치게 많이 반복되는 내용, 혹은 너무 짧은 데이터들을 필터링

텍스트이(가) 표시된 사진

자동 생성된 설명

위 tweepy 코드와 거의 동일한 동작을 하나, 저장된 .csv 파일을 엑셀로 열어 라벨링해야 하는데 이전 코드의 경우 엑셀에서 형식이 깨져서 라벨링하는데 어려움이 있어 extended 모드로 하나의 string으로 저장하는 방식으로 활용함.

**데이터셋**

텍스트이(가) 표시된 사진

자동 생성된 설명

트위터 내용, 라벨 두가지로 구성됨

학습시 가장 위 Text, Label 행만 삭제 후 진행

**distilbert.py**

텍스트이(가) 표시된 사진

자동 생성된 설명

저장된 데이터셋을 읽어, message 열과 label 열을 각각 x,y에 list로 저장하는데, y는 true/false를 1/0으로 변환해서 저장

텍스트이(가) 표시된 사진

자동 생성된 설명

Distilbert model을 사용

각 트윗 내용을 input id, attention mask를 포함하는 tuple로 변환해야함. 이 과정을 DistilbertTokenizer를 활용하여 진행

텍스트이(가) 표시된 사진

자동 생성된 설명

이렇게 각 트윗 내용에 tokenizer를 mapping하는 함수를 정의하고, 각 내용을 mapping하여 encodings에 저장

텍스트이(가) 표시된 사진

자동 생성된 설명

이렇게 input id, attention mask를 포함하는 encodings와 이에 대응하는 y를 tensorflow dataset object로 변환해야함. 이를 tfdataset에 저장

텍스트이(가) 표시된 사진

자동 생성된 설명

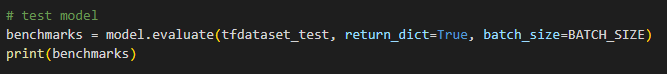
생성한 tfdataset의 20%를 test용으로 분리, 나머지를 학습용으로 활용

주석을 해제하면 데이터셋을 shuffle할 수 있음. 사용해보니 학습결과가 떨어지는 경우가 잦아 굳이 shuffle하지 않음

텍스트이(가) 표시된 사진

자동 생성된 설명

위에서 설정한 tfdataset\_train을 활용하여 model에 fit하여 fine-tuning을 진행함



fine-tuning한 model을 위에서 설정한 tfdataset\_test를 활용하여 testing함

텍스트이(가) 표시된 사진

자동 생성된 설명

새로운 sample data에 대해 predict를 진행할 수 있는 함수 create\_predictor를 정의, 이를 활용하여 sample data에 대해 model을 적용하여 분석할 수 있음

텍스트이(가) 표시된 사진

자동 생성된 설명

학습시킨 model과 MODEL\_NAME, MAX\_LEN값을 지정한 위치에 저장

**ad\_test.py**



지정한 위치에서 학습된 model을 load

이후 이전에 설명한 함수를 그대로 활용하여 predict를 진행

왜 distilbert를 사용했는가? 왜 tensorflow를 사용했는가?

자연어처리 분야에서 transformer 기반의 pre-trained language model인 BERT가 등장, 주역이 됨

그러나 BERT 기반 모델의 크기는 부담스러울정도로 증가



~2020 recent pretraining model’s parameter quantity

모델 크기를 줄이기 위한 경량화 방법이 활발히 연구되고, 그 중 하나가 DistilBERT

DistilBERT는 지식 증류, Knowledge Distillation)을 이용하여 BERT를 경량화한 모델

크기가 작은 모델(student network)이 학습이 잘 된 크기가 큰 모델(teacher network)의 분포를 따르도록 함으로써 작은 모델의 성능이 큰 모델의 성능을 따라잡도록 하는게 목표.

학습시 큰 모델의 출력(벡터)을 지식을 증류할 대상이 되는 자식 모델의 정답 라벨로 사용

사전학습(pre-training)과정에서 3가지 Loss를 사용 : CrossEntropy Loss, Masked Language Modeling Loss, Cosine Embedding Loss

Lce = ∑ ti ∗ log(si)

ti 는 teacher model인 BERT의 출력, si 는 student model인 DistilBERT의 출력을 의미함.

텍스트, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명

student layer의 구조는 기존 BERT와 동일하나, token type embedding과 pooler layer는 없어지고 transformer block을 두배로 줄였다. 또한, initialization은 teacher의 레이어 2개당 1개를 취했다.

그 결과, BERT보다 모델의 크기는 40% 줄이고, 연산 속도는 60% 증가했으며 성능은 BERT의 97% 수준을 유지할 수 있었음.

이처럼, 훨씬 저렴한 컴퓨팅 파워로 BERT와 거의 동등한 수준의 연산을 수행할 수 있다는 장점이 있음. 광고 분석의 경우 단순 이진 분류 작업이므로 다른 고성능의 모델보다는 최대한 가벼운 모델을 사용하자는 생각으로 Distilbert를 사용하기로 결정함.

Reference : https://arxiv.org/abs/1910.01108