|  |  |
| --- | --- |
| 교육제목 | 데이터 기반 인공지능 시스템 엔지니어 양성 과정 |
| 교육일시 | 2021-11-30 |
| 교육장소 | YGL 학과장 및 자택(디스코드 사용 온라인 학습) |
| **교육내용** | |
| 1. Byte Pair Encoding(BPE)   데이터 압축을 위해 생겨났던 알고리즘. 데이터에서 가장 많이 등장하는 바이트 쌍(Byte Pair)을 새로운 단어로 치환하여 압ㅊ축하는 작업을 반복하는 방식으로 작동함.  예)  aaabdaaabac # 가장 많이 등장한 바이트 쌍 "aa"를 "Z"로 치환합니다.  →  ZabdZabac # "aa" 총 두 개가 치환되어 4바이트를 2바이트로 압축하였습니다.  Z=aa # 그다음 많이 등장한 바이트 쌍 "ab"를 "Y"로 치환합니다.  →  ZYdZYac # "ab" 총 두 개가 치환되어 4바이트를 2바이트로 압축하였습니다.  Z=aa # 여기서 작업을 멈추어도 되지만, 치환된 바이트에 대해서도 진행한다면  Y=ab # 가장 많이 등장한 바이트 쌍 "ZY"를 "X"로 치환합니다.  →  XdXac  Z=aa  Y=ab  X=ZY # 압축이 완료되었습니다!  이를 토큰화에 적용하면 모든 단어들을 문자들의 집합으로 취급하여 자주 등장하는 문자 쌍을 합치면, 접두어나 접미어의 의미를 캐치할 수 있고, 처음 등장하는 단어는 문자(알파벳)들의 조합으로 나타내어 OOV문제를 완전히 해결할 수 있다.  Ex)  import re, collections  num\_merges = 10 # BPE를 몇 회 수행할 것지 정함.  dictionary = {'l o w </w>' : 5,  'l o w e r </w>' : 2,  'n e w e s t </w>' : 6,  'w i d e s t </w>' :3  }  def get\_stats(dictionary):  """  단어 사전을 불러와  단어는 공백 단위로 쪼개어 문자 list를 만들고  빈도수와 쌍을 이루게 합니다. (symbols)  """    # 유니그램의 pair들의 빈도수를 카운트  pairs = collections.defaultdict(int)  for word, freq in dictionary.items():  symbols = word.split()  for i in range(len(symbols)-1):  pairs[symbols[i], symbols[i+1]] += freq  print('현재 pair들의 빈도수 :', dict(pairs))  return pairs  def merge\_dictionary(pair, v\_in):  v\_out = {}  bigram = re.escape(' '.join(pair))  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')  for word in v\_in:  w\_out = p.sub(''.join(pair), word)  v\_out[w\_out] = v\_in[word]  return v\_out  bpe\_codes = {}  bpe\_codes\_reverse = {}  for i in range(num\_merges):  print(">> Step {0}".format(i+1))  pairs = get\_stats(dictionary)  best = max(pairs, key=pairs.get)  dictionary = merge\_dictionary(best, dictionary)  bpe\_codes[best] = i  bpe\_codes\_reverse[best[0] + best[1]] = best  print("new merge: {}".format(best))  print("dictionary: {}".format(dictionary))   1. OOV 대처하기   BPE를 사용하면 아무리 큰 데이터도 원하는 크기로 OOV 문제 없이 사전을 정의할 수 있다.  Embedding 레이어는 단어의 개수 \* Embedding 차원 수의 Weight를 생성하기 때문에 단어의 개수가 줄어드는 것은 곧 메모리의 절약으로 이어짐. 많은 데이터가 곧 정확도로 이어지기 때문에 이런 기여는 굉장히 의미가 있음.  Ex)  def get\_pairs(word):  pairs = set()  prev\_char = word[0]  for char in word[1:]:  pairs.add((prev\_char, char))  prev\_char = char  return pairs  orig = 'hi'  word = tuple(orig) + ('</w>',)  print(word)  def encode(orig):  word = tuple(orig) + ('</w>',)  print("\_\_word split into characters:\_\_ <tt>{}<tt>".format(word))  pairs = get\_pairs(word)  if not pairs:  return orig    iteration = 0  while True:  iteration += 1  print("\_\_Iteration {}:\_\_".format(iteration))  print("Bigram in the word: {}".format(pairs))  bigram = min(pairs, key = lambda pair: bpe\_codes.get(pair, float('inf')))  print("candidate for merging: {}".format(bigram))  if bigram not in bpe\_codes:  print("\_\_Candidate not in BPE merges, algorithm stops.\_\_")  break  first, second = bigram  new\_word = []  i = 0  while i < len(word):  try:  j = word.index(first, i)  new\_word.extend(word[i:j])  i = j  except:  new\_word.extend(word[i:])  break  if word[i] == first and i < len(word)-1 and word[i+1] == second:  new\_word.append(first+second)  i += 2  else:  new\_word.append(word[i])  i += 1  new\_word = tuple(new\_word)  word = new\_word  print("word after merging : {}".format(word))  if len(word) == 1:  break  else:  pairs = get\_pairs(word)    # 특별토큰인 </w>는 출력하지 않는다.  if word[-1] == '</w>':  word = word[:-1]  elif word[-1].endswith('</w>'):  word = word[:-1] + (word[-1].replace('</w>', ''), )  return word   1. Wordpiece Model(WPM)   구글에서 BPE를 변형해 제안한 알고리즘. WPM이 BPE에 대해 가지는 두 가지 차별성  -1. 공백 복원을 위해 단어의 시작 부분에 언더바\_를 추가함.  -2. 빈도수 기반이 아닌 가능도를 증가시키는 방향으로 문자 쌍을 합침.   1. IMDB리뷰 tensorflow의 subvwordTextEncoder로 토큰화   import tensorflow\_datasets as tfds  import urllib.request  import pandas as pd  urllib.request.urlretrieve("https://raw.githubusercontent.com/LawrenceDuan/IMDb-Review-Analysis/master/IMDb\_Reviews.csv", filename="IMDb\_Reviews.csv")  train\_df = pd.read\_csv('IMDb\_Reviews.csv')  # train\_df에서 'review'컬럼만 출력하시오.  train\_df['review']  # train\_df에서 'sentiment'컬럼만 출력하시오.  train\_df['sentiment']  tokenizer = tfds.deprecated.text.SubwordTextEncoder.build\_from\_corpus(train\_df['review'], target\_vocab\_size=2\*\*13) # 2의 13승  # tokenizer의 subword를 100까지 출력하세요.  print(tokenizer.subwords[:100])  # train\_df의 'review'컬럼에서 20번째 문장을 출력하시오.  print(train\_df['review'][20])  aa = train\_df['review'][20]  # tokenizer.encode를 활용하여 train\_df의 20번째 문장을 텍스트 -> 숫자로 바꾸시오.  print("토큰화된 샘플 질문 : {}".format(tokenizer.encode(aa)))   1. IMDB리뷰 sentencePiece로 토큰화   !pip install sentencepiece  import sentencepiece as spm  import pandas as pd  import urllib.request  import csv  urllib.request.urlretrieve("https://raw.githubusercontent.com/LawrenceDuan/IMDb-Review-Analysis/master/IMDb\_Reviews.csv", filename="IMDb\_Reviews.csv")  train\_df = pd.read\_csv('IMDb\_Reviews.csv')  print('리뷰 개수 : ', len(train\_df))  with open('imdb\_review.txt', 'w', encoding='utf8') as f:  f.write('\n'.join(train\_df['review']))  corpus = "imdb\_review.txt" #입력 corpus  prefix = "imdb" # 저장할 단어장 이름  vocab\_size = 5000  spm.SentencePieceTrainer.Train(  f"--input={corpus} --model\_prefix={prefix} --vocab\_size={vocab\_size}" +  " --model\_type=bpe" +  " --max\_sentence\_length=9999") # 문장 최대 길이  vocab\_list = pd.read\_csv('imdb.vocab', sep='\t', header=None, quoting=csv.QUOTE\_NONE)  vocab\_list.sample(10)  len(vocab\_list)  lines = [  "I didn't at all think of it this way.",  "I have waited a long time for someone to film"  ]  for line in lines:  print(line)  print(sp.encode\_as\_pieces(line)) # 문장을 입력하면 서브워드 시퀀스로 변환  print(sp.encode\_as\_ids(line)) # 문장을 입력하면 정수 시퀀스로 변환  print()  sp.GetPieceSize() #단어집합의 크기  sp.IdToPiece(120) #정수로부터 매핑되는 서브워드 변환  # 위에서 출력된 결과를 sp.PieceTold 안으로 입력하기  sp.PieceToId('▁not') # 서브워드로부터 매핑되는 정수로 변환  # 위에서 출력된 정수 시퀀스를 sp.DecodeIds의 정수 시퀀스에 입력하기  sp.DecodeIds([41, 141, 1364, 1120, 4, 666, 285, 92, 1078, 33, 91])  # 서브워드 시퀀스로부터 문장으로 변환  sp.DecodePieces(['▁I', '▁have', '▁wa', 'ited', '▁a', '▁long', '▁time', '▁for', '▁someone', '▁to', '▁film'])  print(sp.encode('I have waited a long time for someone to film', out\_type=str)) # subword시퀀스  print(sp.encode('I have waited a long time for someone to film', out\_type=int)) # 정수 시퀀스 | |