IAIP 2023-1 HW7

22100579 이진주

Part0: clean the texts

|  |
| --- |
| def remove\_html(text\_data):    soup = BeautifulSoup(text\_data, 'lxml')    return soup.get\_text();  def remove\_punctuation(text):    sent = []    for t in text.split(' '):      no\_punc = "".join([c for c in t if c not in string.punctuation])      sent.append(no\_punc)    sentence = " ".join(s for s in sent)    return sentence  def tolower(text):    return text.lower()  def lemmatization(text):    nlp = spacy.load('en\_core\_web\_sm')    doc = nlp(text.strip())    tok\_lem\_sentence = [token.lemma\_ for token in doc]    sentence = " ".join(s for s in tok\_lem\_sentence)    return sentence  def removeStopword(text):    stop\_words = stopwords.words('english')    # print("stop words: ", stop\_words)    # print(text, '\n')    rmv\_sw\_sentence = [w for w in text.split() if not w in stop\_words]    # print(rmv\_sw\_sentence)    removed\_word = [w for w in text.split() if not w in rmv\_sw\_sentence]    # print("\nRemoved word: ", set(removed\_word))    sentence = " ".join(s for s in rmv\_sw\_sentence)    return sentence  def clean(text):    text = remove\_html(text)    text = remove\_punctuation(text)    text = lemmatization(text)    text = tolower(text)    text = removeStopword(text)    return text |
|  |

Part1: print most 5 frequent words for each review data.

|  |
| --- |
| from collections import Counter  most5 = []  for i in range(len(train)):    tokens = train['review'][i]    freq = Counter(tokens)    top5 = freq.most\_common(5)    most5.append(top5)    print("review ", i, ": ", end="")    for j in range(len(top5)):      if(j!= 4):        print(top5[j][0], ", ", end="")      else:        print(top5[j][0]) |
|  |

Part2 -v1: Make a word-to-index dictionary from the train data set

|  |
| --- |
| #part2-v1: word to index dictionary  #case2 통합  import numpy as np  dictionary2 = {}  def make\_frequency\_dict2(text):    for word in text:      if word not in dictionary2:        dictionary2[word] = 0      dictionary2[word] += 1  for i in train['review']:    make\_frequency\_dict2(i)  vocab\_sorted2 = {}  vocab\_sorted2 = sorted(dictionary2.items(), key=lambda x:x[1], reverse = True)  word2index2 = {}  index = 0  for (word, freq) in vocab\_sorted2:    if freq > 1:      word2index2[word] = index      index += 1  word2index2['OOV'] = index  encoded2 = []  for w in most5:    tmp = []    print(w)    for one in w:      tmp.append(word2index2.get(one, word2index2['OOV']))    print(tmp)    encoded2.append(tmp) |
|  |

Part2 -v2: Make a word-to-rating dictionary.

|  |
| --- |
| #part2-v2 : word to rating dictionary  word2rating = []  five\_rating\_dict = [] #(dict indexs) /  0, 1, 2, 3, 4(ratings)  for i in range(index+1):    tmp = [0,0,0,0,0]    five\_rating\_dict.append(tmp)  #일단 five\_rating\_dict[i]의 0-4(1-5)에다가 별점 쌓기  for i in range(len(encoded2)):    for j in encoded2[i]:      five\_rating\_dict[j][int(train['rating'][i]-1)] += 1  print("rating 1  2  3  4  5")  for i in range(index+1):    print(i,": ", end='')    print(five\_rating\_dict[i], end='')    pred = five\_rating\_dict[i].index(max(five\_rating\_dict[i]))+1    word2rating.append(pred)    print(": ", pred)  print("\nword to rating = ", end='')  for i in range(index+1):    print(word2rating[i], ",", end='') |
|  |

Part4-1: encode test data and predict the rating of test review,

|  |
| --- |
| # Part4-1: encode test data and predict the rating of test review,  from collections import Counter  #하나의 string review에 대해 예측 rating을 리턴하는 함수  def getPredictedRating(review):    enc = []    pred = []    for i in range(len(review)):      enc.append(word2index2.get(review[i], word2index2['OOV']))      pred.append(word2rating[enc[i]])    counter = Counter(pred)    most\_common = counter.most\_common(1)[0]  return most\_common[0]  correct = 0  print("actial : predicted")  for i in range(len(test)):    print(test['rating'][i], end='')    p = getPredictedRating(test['review'][i])    print('    : ', p, '.0')    if(int(test['rating'][i] == p)):      correct += 1  print("count of correct : ", correct, "/", len(test)) |
|  |

Part4-2: suggest how to evaluate your predicted result.

|  |
| --- |
| count of correct : 8 / 26 |

* 전체 test data들의 예측된 rating들과 실제 rating의 값을 비교하여 일치하는 data의 비율을 계산한다.

Part4-3: suggest how to improve your results.

* 현재 코드상에서는 word to index로 매핑되는 단어의 기준을 1개보다 많이 등장했을 때로 설정했는데, 이로 인해 인덱스의 스펙트럼이 너무 넓어져, 상위 5개 모음에 한번도 등장하지 않은 단어 index는 매칭되는 rating도 없어 별점 1점으로 고정되고, 이로 인해 예층 rating에서 1의 비율이 증가했다. 여러 시도를 통해 word to index로 매핑되는 단어 기준을 적절하게 조절하여 정확도를 올릴 수 있을 것 같다.