

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
```

✓ [10] train

	rating	review
0	1.0	[4]
1	1.0	[order, pb, embroider, coat, opposite, colour,...
2	1.0	[small, think, little, price, worth, £, 5, mak...
3	1.0	[thought, would, make, lovely, different, cale...
4	1.0	[sand, rubbish, messy, stick, together, like, ...
...
496	5.0	[daughter, love, run, jump, excitement]
497	5.0	[great, model]
498	5.0	[fantastic, detail, beautiful, model, traction...
499	5.0	[easy, couple, model, great, expand, set, powe...
500	5.0	[buy, 2, year, old, grandson, love, much, also...

501 rows x 2 columns

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])
```

```

review 432 : set , arrive , 27th , perfect , condition
review 433 : model , standard , door , epoch , 1
review 434 : kit , noncorridor , produce , excellent , exlms
review 435 : really , good , service , great , product
review 436 : great , addition , auto , city , set
review 437 : model , something , br , modeller , want
review 438 : item , moon , exemplary , service , fantastic
review 439 : fantastic , lamp , fair , price , review 440 : model , standard , shop , epoch , 1
review 441 : build , kit , model , top , spin
review 442 : thank , well , make , lyr , 242
review 443 : keep , husband , happy , well , impressed
review 444 : need , controller , speed , decide , model
review 445 : good , kit , go , together , well
review 446 : f7a , quality , walthers , proto , locomotive
review 447 : christmas , follow , toot , train , brilliant
review 448 : train , german , model , company , budget
review 449 : signal , easy , plate , head , type
review 450 : yes , would , recommend , accord , recipient
review 451 : lovely , little , engine , son , love
review 452 : son , day , love , ime , pleased
review 453 : hornby , uncouple , ramptook , session , master
review 454 : quick , job , reasonable , delivery , thank
review 455 : fab , item , review 456 : cleaning , wagon , everything , require , much
review 457 : train , crane , good , set , normal
review 458 : work , take , little , time , instruction
review 459 : excellent , signal , kit , simple , build
review 460 : chassis , wheel , easy , 8 , drive
review 461 : locomotive , piko , mak , g , dcc
review 462 : look , review 463 : really , nice , quality , ho , coach
review 464 : great , review 465 : true , product , discription , review 466 : really , train , lovely , collector , sure
review 467 : happy , purchase , 4 , year , old
review 468 : excellent , engine , arrive , day , early

```

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

```

#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 all...
for i in train['review']:
    make_frequency_dict2(i)

vocab_sorted2 = {}
vocab_sorted2 = sorted(dictionary2.items(), key=lambda x:x[1],
reverse = True)

i = 0
#다섯개
for (word, freq) in vocab_sorted2:
    if i < 5:
        word2index2[word] = i

```

```

    i += 1

#나머지 몰아넣기
word2index2['OOV'] = i

encoded2 = []

for w in most5:
    tmp = []
    print(w)
    for one in w:
        tmp.append(word2index2.get(one, word2index2['OOV']))
    print(tmp)
    encoded2.append(tmp)

```

```

✓ [32] [4, 5, 2, 5, 5]
      ['first', 'class']
      [5, 5]
      ['loco', 'superb', 'model', 'hornby', 'exger']
      [5, 5, 5, 5, 5]
      ['model', 'standard', 'epoch', '1', 'two']
      [5, 5, 5, 5, 5]
      ['mum', 'buy', 'send', '2', 'month']
      [5, 0, 5, 5, 5]
      ['car', 'walters', 'trainline', 'make', 'great']
      [5, 5, 5, 5, 5]
      ['model', 'shop', 'excellent', 'would', 'pay']
      [5, 5, 5, 3, 5]
      ['first', 'class', 'model', 'usual', 'kato']
      [5, 5, 5, 5, 5]
      ['excellent', 'purpose', 'buy', 'make', 'rock']
      [5, 5, 0, 5, 5]
      ['receive', 'good', 'train', 'set', 'excellent']
      [5, 2, 5, 5, 5]
      ['signal', 'good', 'value', 'money', 'relatively']
      [5, 2, 5, 5, 5]
      ['coupling', 'railjet', 'coach', 'roco', 'internal']
      [5, 5, 5, 5, 5]
      ['model', 'old', 'e7', 'kato', 'still']
      [5, 5, 5, 5, 5]
      ['see', 'review', 'head', 'station']
      [5, 5, 5, 5]
      ['good', 'track', 'tight', 'fitting', 'perfect']
      [2, 5, 5, 5, 5]
      ['buy', 'cheap', 'alternative', 'basically', 'thing']
      [0, 5, 5, 5, 5]
      ['daughter', 'love', 'run', 'jump', 'excitement']
      [5, 5, 5, 5, 5]
      ['great', 'model']
      [5, 5]

```

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

```

#part2-v2 : word to rating dictionary

word2rating = []
five_rating_dict = [[0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    ] #0, 1, 2, 3, 4, 5(dict) / 0, 1, 2, 3, 4(ratings)

#일단 five_rating_dict[i]의 0-4(1-5)에다가 별점 쌓기

```

```

for i in range(len(encoded2)):
    for index in encoded2[i]:
        five_rating_dict[index][int(train['rating'][i]-1)] += 1

print("rating 1 2 3 4 5")
for i in range(6):
    print(i,": ", end='')
    print(five_rating_dict[i], end='')
    word2rating.append(five_rating_dict[i].index(max(five_rating_dict[i]))+1)
    print(": ", five_rating_dict[i].index(max(five_rating_dict[i]))+1)

print("\nword to rating = ", end='')
word2rating

```

```

rating 1 2 3 4 5
0 : [12, 11, 6, 6, 5]: 1
1 : [6, 6, 7, 6, 1]: 3
2 : [2, 6, 13, 9, 13]: 3
3 : [8, 1, 2, 1, 2]: 1
4 : [5, 1, 4, 6, 6]: 4
5 : [464, 474, 460, 451, 433]: 2

word to rating = [1, 3, 3, 1, 4, 2]

```

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 review 에 대해 예측 rating 을 리턴하는 함수
def getPredictedRating(review):
    #인코딩 + 각 index 의 예측 rating 저장
    enc = []
    pred = []
    for i in range(len(review)):
        enc.append(word2index2.get(review[i], word2index2['OOV']))
        pred.append(word2rating[enc[i]])
    # print(enc)
    # print(pred)

    counter = Counter(pred)
    most_common = counter.most_common(1)[0]
    return most_common[0]

```

```

correct = 0
print("actual : 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))

```

```

actual : predicted
1.0 : 2.0
1.0 : 2.0
1.0 : 2.0
1.0 : 2.0
1.0 : 2.0
1.0 : 2.0
2.0 : 2.0
2.0 : 2.0
2.0 : 2.0
2.0 : 2.0
2.0 : 2.0
3.0 : 2.0
3.0 : 2.0
3.0 : 2.0
3.0 : 2.0
3.0 : 2.0
4.0 : 2.0
4.0 : 2.0
4.0 : 2.0
4.0 : 2.0
4.0 : 2.0
5.0 : 2.0
5.0 : 2.0
5.0 : 2.0
5.0 : 2.0
5.0 : 2.0
count of correct : 5 / 26

```

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

```
count of correct : 5 / 26
```

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

Part4-3: suggest how to improve your results.

- 현재 코드상에서는 너무 적은 표본으로 word to index를 만들어, 절대다수의 단어들이 OOV에 포함되었기 때문에 예측 값이 OOV의 예상 rating인 2.0으로 편향된 상태이다. 이를 개선하기 위하여, word to index를 만들 때 인코딩을 위해 사용될 값들(OOV가 아닌 실제 단어와 매칭될 값들)을 5개보다 더 많게 하면 보다 많은 case가 고려되므로 정확도가 올라갈 것 같다.