

March 6, 2017

HOMEWORK FOUR

Problem 1. Solution

The number of unique bigrams: 182246

Table 1: The 5 most-frequently-occurring bigrams:

Bigrams	The number of occurrences
'with a'	4587
'in the'	2595
'of the'	2245
'is a'	2056
'on the'	2033

Listing 1: Key code for Prob.1

```
1 wordCount = defaultdict(int)
2 punctuation = set(string.punctuation)
3 for d in data:
4     r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
5     w = r.split()
6     for i in range(len(w)-1):
7         wordCount[w[i]+' '+w[i+1]] += 1
8 len(wordCount)
9
10 counts = [(wordCount[w] , w) for w in wordCount]
11 counts.sort()
12 counts.reverse()
13 counts
```

Problem 2. Solution

MSE using bigrams: 0.34315301406136378

Listing 2: Key code for Prob.2

```
1 words = [x[1] for x in counts[:1000]]
2 wordId = dict(zip(words, range(len(words))))
3 wordSet = set(words)
```

```

4
5 def feature(datum):
6     feat = [0]*len(words)
7     r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
8     w = r.split()
9     for i in range(len(w)-1):
10         bitemp=w[i]+' '+w[i+1]
11         if bitemp in words:
12             feat[wordId[bitemp]] += 1
13     feat.append(1) #offset
14     return feat
15
16 X = [feature(d) for d in data]
17 y = [d['review/overall'] for d in data]
18
19 clf = linear_model.Ridge(1.0, fit_intercept=False)
20 clf.fit(X, y)
21 theta = clf.coef_
22 predictions = clf.predict(X)
23 diff = predictions-y
24 mse = sum([t**2 for t in diff])/len(diff)
25 mse

```

Problem 3. Solution

MSE using unigrams and bigrams: 0.28904733303355806

Listing 3: Key code for Prob.3

```

1 wordCount = defaultdict(int)
2 punctuation = set(string.punctuation)
3 for d in data:
4     r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
5     w = r.split()
6     for i in range(len(w)-1):
7         wordCount[w[i]+' '+w[i+1]] += 1
8     for tmp in w:
9         wordCount[tmp] += 1
10 len(wordCount)
11
12 counts = [(wordCount[w] , w) for w in wordCount]
13 counts.sort()
14 counts.reverse()
15 # counts
16 words = [x[1] for x in counts[:1000]]
17 wordId = dict(zip(words, range(len(words))))
18 wordSet = set(words)
19
20 def feature(datum):
21     feat = [0]*len(words)
22     r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
23     w = r.split()
24     for i in range(len(w)-1):
25         bitemp=w[i]+' '+w[i+1]
26         if bitemp in words:
27             feat[wordId[bitemp]] += 1

```

```

28     for tmp in w:
29         if tmp in words:
30             feat[wordId[tmp]] += 1
31     feat.append(1) #offset
32     return feat
33
34 X = [feature(d) for d in data]
35 y = [d['review/overall'] for d in data]
36
37 clf = linear_model.Ridge(1.0, fit_intercept=False)
38 clf.fit(X, y)
39 theta = clf.coef_
40 predictions = clf.predict(X)
41
42 diff = predictions-y
43 mse = sum([t**2 for t in diff])/len(diff)

```

Problem 4. Solution

Table 2: The 5 unigrams/bigrams with the most positive associated weights:

Unigrams/Bigrams	Weights
'sort'	0.51982780120456751
'a bad'	0.22881971426910591
'of these'	0.22283470424121538
'not bad'	0.21687721630732146
'the best'	0.20639109567227393

Table 3: The 5 unigrams/bigrams with the most negative associated weights:

Unigrams/Bigrams	Weights
'sort of'	-0.63976214971855316
'water'	-0.27048649882966247
'corn'	-0.23703101460442585
'the background'	-0.21624829959516467
'straw'	-0.19593772177944399

Listing 4: Key code for Prob.4

```

1 weight = [(theta[i] , words[i]) for i in range(len(words))]
2 weight.sort()
3 #weight
4
5 weight.reverse()
6 #weight

```

Problem 5. Solution

Table 4: Inverse document frequency:

Words	Idf
'foam'	1.1378686206869628
'smell'	0.5379016188648442
'banana'	1.6777807052660807
'lactic'	2.9208187539523753
'tart'	1.8068754016455384

Table 5: Tf-idf Scores in the first review:

Words	Tf-idf
'foam'	2.2757372413739256
'smell'	0.5379016188648442
'banana'	3.3555614105321614
'lactic'	5.841637507904751
'tart'	1.8068754016455384

Listing 5: Key code for Prob.5

```
1 wordList = ['foam','smell','banana','lactic','tart']
2 wordCount = defaultdict(int)
3 punctuation = set(string.punctuation)
4
5 for d in data:
6     r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
7     w = r.split()
8     for word in wordList:
9         if word in w:
10             wordCount[word] += 1
11
12 freq = [math.log10(len(data) * 1.0 / wordCount[word]) for word in wordList]
13 #freq
14
15 d=data[0]
16 wordCount2 = defaultdict(int)
17 r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
18 w = r.split()
19 for word in w:
20     if word in wordList:
21         wordCount2[word] += 1
22
23 tfidf=[freq[i]*wordCount2[wordList[i]] for i in range(len(wordList))]
24 #tfidf
```

Problem 6. Solution

The cosine similarity between the rst and the second review in terms of their tf-idf representations:
0.106130241679

Listing 6: Key code for Prob.6

```
1 wordCount1 = defaultdict(int)
2 punctuation = set(string.punctuation)
3
4 for d in data:
5     r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
6     w = r.split()
7     for word in wordList:
8         if word in w:
9             wordCount1[word] += 1
10
11 freq = [math.log10(len(data) * 1.0 / wordCount1[word]) for word in wordList]
12 d=data[0]
13 wordCount2 = defaultdict(int)
14 r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
15 w = r.split()
16 for word in w:
17     if word in wordList:
18         wordCount2[word] += 1
19 tfidf1=[freq[i]*wordCount2[wordList[i]] for i in range(len(wordList))]
20
21 d=data[1]
22 wordCount2 = defaultdict(int)
23 r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
24 w = r.split()
25 for word in w:
26     if word in wordList:
27         wordCount2[word] += 1
28 tfidf2=[freq[i]*wordCount2[wordList[i]] for i in range(len(wordList))]
29 1-sciipy.spatial.distance.cosine(tfidf1,tfidf2)
```

Problem 7. Solution

Which other review has the highest cosine similarity compared to the rst review (provide the beerId and proleName, or the text of the review):

BeerId: 52211

User/profileName: 'Heatwave33'

Max cosine similarity: 0.31732766002633128

Number in the dataset: 4003

Listing 7: Key code for Prob.7

```

1 cosine_m=0
2 num_max=-1
3
4 for t in range(1,len(data)):
5     d=data[t]
6     wordCount2 = defaultdict(int)
7     r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
8     w = r.split()
9     for word in w:
10         if word in wordList:
11             wordCount2[word] += 1
12     tfidf2=[freq[i]*wordCount2[wordList[i]] for i in range(len(wordList))]
13     tmp=1-sciPy.spatial.distance.cosine(tfidf1,tfidf2)
14     if(tmp>cosine_m):
15         cosine_m=tmp
16         num_max=t
17
18 print data[num_max]

```

Problem 8. Solution

MSE: 0.27875956007772162

Listing 8: Key code for Prob.8

```

1 wordCount = defaultdict(int)
2 punctuation = set(string.punctuation)
3 for d in data:
4     r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
5     for w in r.split():
6         wordCount[w] += 1
7
8 counts = [(wordCount[w], w) for w in wordCount]
9 counts.sort()
10 counts.reverse()
11
12 wordList = [x[1] for x in counts[:1000]]
13
14 wordCount1 = defaultdict(int)
15 punctuation = set(string.punctuation)
16
17 for d in data:
18     r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
19     w = r.split()
20     for word in wordList:
21         if word in w:
22             wordCount1[word] += 1
23
24 freq = [math.log10(len(data) * 1.0 /wordCount1[word]) for word in wordList]
25
26 wordId = dict(zip(words, range(len(words))))
27 wordSet = set(words)
28
29 def feature(datum):

```

```

30     feat = [0]*len(words)
31     wordCount2 = defaultdict(int)
32     r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
33     w = r.split()
34     for word in w:
35         if word in wordList:
36             wordCount2[word] += 1
37     for i in range(len(wordList)):
38         feat[i]=freq[i]*wordCount2[wordList[i]]
39     feat.append(1) #offset
40     return feat
41
42 X = [feature(d) for d in data]
43 y = [d['review/overall'] for d in data]
44
45 clf = linear_model.Ridge(1.0, fit_intercept=False)
46 clf.fit(X, y)
47 theta = clf.coef_
48 predictions = clf.predict(X)
49
50 diff = predictions-y
51 mse = sum([t**2 for t in diff])/len(diff)

```

Submitted by Zhang, Jinhan PID A53211930 on March 6, 2017.