HW12 보고서

2013-10035
언어학과 김영민

1.0.1 파일 읽기

- 코드

```
# 1.0.1 read file
with open('./poems.txt', 'r') as fin:
    data = []
    for line in fin:
        elems = line.strip().split('\t')
        data.append((int(elems[0]), int(elems[1]), [ch for ch in elems[4]]))
```

- 출력 결과

```
>>> len(data)
160
>>> data[0]
(0, 1, ['垂', '綏', '饮', '清', '露', ' ', ' ', '流', '响', '出', '疏', '桐', '。', '居', '高', '声', '自', '远', ' ', ' ', '非', '是', '藉', '秋', '风', '。'])
```

1.0.2 순서 뒤섞기

- 코드

```
# 1.0.2 shuffle data
seed(471)
shuffle(data)
```

- 출력 결과

```
>>> data[0]
(97, 1, ['草', '树', '知', '春', '不', '久', '归', ' ', ' ', '百', '般', '红', '紫', '斗', '芳', '菲', '。', '杨', '花', '榆'
'荚', '无', '才', '思', ' ', ' ', '惟', '解', '漫', '天', '作', '雪', '飞', '。'])
```

과제 pdf 와 시드값을 동일하게 줬기 때문에 순서를 뒤섞었을 때 pdf 와 동일하게 섞인 것을 확인할 수 있다.

1.1.1 logprior

- 코드

```
# 1.1.1 logprior
Ndoc = len(train)
Nc = Counter([v[1] for v in train])
logprior = {k: log(v/Ndoc) for k, v in Nc.items()}
```

긍정(부정) 시의 개수를 전체 시의 개수로 나눈 값에 로그를 씌워 계산하였다.

- 출력결과

```
>>> Nc
Counter({0: 75, 1: 69})
>>> Ndoc
144
>>> logprior
{1: -0.7357067949787413, 0: -0.6523251860396901}
```

1.1.2 로그가능도

- 코드 (vocabulary, bigdoc)

```
# 1.1.2 log likelihood
bigdoc  = {0:[], 1:[]}
vocabulary = []
for entry in train:
    for ch in entry[2]:
        if ch not in vocabulary:
            vocabulary.append(ch)
        if entry[1] == 0:
            bigdoc[0].append(ch)
        if entry[1] == 1:
            bigdoc[1].append(ch)
```

- 출력 결과

```
>>> len(vocabulary)
1403
>>> vocabulary[:14]
['草', '树', '知', '春', '不', '久', '归', '，', '百', '般', '红', '紫', '斗', '芳']
```

```
>>> bigdoc[1][:14]
['草', '树', '知', '春', '不', '久', '归', '，', '百', '般', '红', '紫', '斗', '芳']
>>> bigdoc[0][:14]
['故', '乡', '杳', '无', '际', '，', '日', '暮', '且', '孤', '征', '。', '川', '原']
>>> len(bigdoc[1]), len(bigdoc[0])
(2941, 3606)
```

- 코드 (counts)

```
counts = {k: Counter(v) for k, v in bigdoc.items()}
```

- 출력 결과

```
>>> counts[1].most_common(5)
[('，', 211), ('。', 210), ('山', 30), ('人', 26), ('春', 24)]
>>> counts[0].most_common(5)
[('，', 260), ('。', 239), ('人', 39), ('不', 37), ('春', 32)]
>>> len(counts[1]), len(counts[0])
(964, 1011)
>>> counts[1]["想"]
1
>>> counts[0]["想"]
0
>>> counts[1]["哀"]
0
>>> counts[0]["哀"]
1
```

- 코드(평탄화)

```
for ch in vocabulary:
    for cnts in counts.values():
        cnts[ch] += 1
```

vocabulary에 있는 모든 글자에 대해 긍정, 부정에서의 빈도를 모두 1씩 올려주도록 구현했다.

- 출력 결과

```
>>> counts[1].most_common(5)
[(' ', 212), ('。', 211), ('山', 31), ('人', 27), ('春', 25)]
>>> counts[0]["想"]
1
>>> counts[1]["哀"]
1
>>> len(counts[1]), len(counts[0])
(1403, 1403)
```

- 코드 (loglikelihood)

```
loglikelihood = {ch: {1: log(counts[1][ch] / (len(bigdoc[1]) + len(counts[1]))),
                      0: log(counts[0][ch] / (len(bigdoc[0])+ len(counts[0])))}
                 for ch in vocabulary}
```

- 출력 결과

```
>>> loglikelihood["乡"]
{1: -8.37655086161377, 0: -6.321766996021397}
>>> loglikelihood["坐"]
{1: -6.179326284277552, 0: -8.518991573357617}
>>> loglikelihood["醉"]
{1: -6.990256500493881, 0: -6.909553660923517}
# 근심 수, 빼어날 수
>>> loglikelihood["愁"]
{1: -7.683403681053826, 0: -5.685778229301401}
>>> loglikelihood["秀"]
{1: -6.990256500493881, 0: -8.518991573357617}
```

'근심 수'와 '빼어날 수'의 로그가능도는 위와 같다. '근심 수'의 경우 부정적인 시에 등장할 확률이 더 높고 '빼어날 수'는 긍정적인 시에 등장할 확률이 더 높다.

1.2.1 results

- 코드

```
# 1.2.1 results
results = {}
for entry in test:
    neg_prob = logprior[0] + np.sum([loglikelihood[ch][0] for ch in entry[2] if ch in vocabulary])
    pos_prob = logprior[1] + np.sum([loglikelihood[ch][1] for ch in entry[2] if ch in vocabulary])
    c = argmax((neg_prob, pos_prob))
    results[entry[0]] = (entry[1], c)
```

출력 결과

```
>>> results[111]
(1, 1)
>>> results[144]
(0, 0)
>>> results[14]
(1, 0)
>>> results
{14: (1, 0),
 17: (1, 0),
 19: (0, 0),
 62: (0, 0),
 70: (1, 1),
 76: (1, 1),
 82: (1, 0),
 90: (1, 1),
 92: (0, 0),
 111: (1, 1),
 118: (1, 1),
 126: (0, 0),
 132: (0, 0),
 135: (0, 0),
 138: (0, 0),
 144: (0, 0)}
```

1.3.1 정확도

- 코드

```
# 1.3.1 accuracy
accuracy = len([0 for gt, pred in results.values() if gt == pred]) / len(results)
```

예측과 정답이 일치하는 횟수 / 전체 횟수 로 계산하였다.

- 출력

```
144: (0, 0)}
>>> accuracy
0.8125
>>> accuracy > 0.8
True
```

실제 정확도는 0.8125 (13/16) 이다

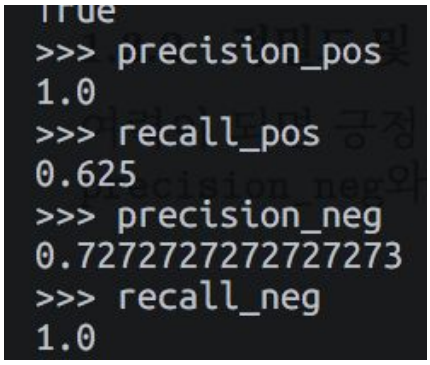
1.3.2 정밀도 및 재현율

- 코드

```
# 1.3.2 precision / recall
precision_pos = sum(gt and pred for gt, pred in results.values()) / sum(pred for _, pred in results.values())
recall_pos = sum(gt and pred for gt, pred in results.values()) / sum(gt for gt, _ in results.values())
precision_neg = sum(not gt and not pred for gt, pred in results.values()) / sum(not pred for _, pred in results.values())
recall_neg = sum(not gt and not pred for gt, pred in results.values()) / sum(not gt for gt, _ in results.values())
```

정밀도는 [긍정(부정) 으로 옳게 예측한 횟수 / 긍정(부정)으로 예측한 총 횟수] 로,
재현율은 [긍정(부정) 으로 옳게 예측한 횟수 / 실제 긍정(부정)인 가짓수] 로 계산했다.

- 출력 결과



위와 같이 계산한 결과 구한 재현율 및 정밀도는 이렇게 나온다.
각 경우에 대해 실제 가짓수를 세 보면 각각 5/5, 5/8, 8/11, 8/8 인데, 이와 일치하는 결과다.