q1

October 6, 2019

In [1]:
```
import pandas as pd
import numpy as np
import string
from sklearn.model_selection import KFold
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score, precision_score, recall_score
from statistics import mean
```

In [2]:
```
data = []
file = open('./all_sentiment_shuffled.txt')

for line in file:
    line = line.split(' ')
    row = []
    row.append(line[0])
    row.append(line[1])
    row.append(line[2])
    row.append(' '.join(line[3:]).translate(str.maketrans('', '', string.punctuation)))
    data.append(row)

len(data)
```

Out[2]:
```
11914
```

In [3]:
```
vocabulary = set([])
classes = set([])
for d in data:
    classes.add(d[1])
    line = d[3]
    for w in line.split(' '):
        if len(w) == 0:
            continue
        vocabulary.add(w)

print(len(vocabulary))
print(len(classes))
```

```
54090
2
```

```
In [4]: def getSubset(train_data, c):
            subset = []
            for d in train_data:
                if (d[1] == c):
                    subset.append(d)
            return subset

        print(len(getSubset(data[:5], 'neg')))
```

```
3
```

```
In [5]: def makeMegaDoc(subset_docs):
            mega_doc = {}
            for d in subset_docs:
                line = d[3].split(' ')
                for w in line:
                    if w == '':
                        continue
                    if w in mega_doc:
                        mega_doc[w] += 1
                    else:
                        mega_doc[w] = 1

            return mega_doc

        # print(makeMegaDoc(data[:5]))
```

```
In [6]: def getFrequency(mega_doc, w):
            if w in mega_doc:
                return mega_doc[w]
            else:
                return 0

        print(getFrequency(makeMegaDoc(data[:5]), 'and'))
```

```
14
```

```
In [7]: def train(train_data, classes, vocabulary, smoothing_factor=1):
            probab_class = {}
            probab_conditional_word = {}
            for c in classes:
                subset_docs = getSubset(train_data, c)
                probab_class[c] = len(subset_docs) / len(train_data)
```

```
            mega_doc = makeMegaDoc(subset_docs)
            for w in vocabulary:
                nk = getFrequency(mega_doc, w)
                probab_conditional_word[w + '|' + c] = (nk + smoothing_factor) / (len(mega_

        return probab_class, probab_conditional_word
```

In [8]:
```
def classify(test_data, classes, probab_class, probab_conditional_word):

    prediction = []

    for d in test_data:
        max_prob = 0
        max_class = ''

        for c in classes:
            curprob = probab_class[c]
            for w in d[3].split(' '):
                if w == '':
                    continue

                curprob = curprob * (probab_conditional_word[w + '|' + c])

            if curprob >= max_prob:
                max_prob = curprob
                max_class = c

        prediction.append(max_class)

    return prediction
```

In [9]:
```
def getActualLabels(act_data):
    act_labels = []
    for d in act_data:
        act_labels.append(d[1])
    return act_labels
```

In [10]:
```
def getDataInIndex(data, index):
    l = []
    for i in range(len(data)):
        if i in index:
            l.append(data[i])
    return l
```

In [11]:
```
kfold = KFold(5, True, 1)
precision = []
recall = []
```

```
        f_score = []

        for trainInd,testInd in kfold.split(data):
            train_data = getDataInIndex(data, trainInd)
            test_data = getDataInIndex(data, testInd)

            probab_class, probab_conditional_word = train(train_data, classes, vocabulary)
            prediction = classify(test_data, classes, probab_class, probab_conditional_word)

            actual = getActualLabels(test_data)
            predicted = prediction

        #     print(classification_report(actual, predicted))
            precision.append(precision_score(actual, predicted, pos_label="pos"))
            recall.append(recall_score(actual, predicted, pos_label="pos"))
            f_score.append(f1_score(actual, predicted, pos_label="pos"))

        print("Precision Score = " + str(mean(precision)))
        print("Recall Score = " + str(mean(recall)))
        print("F Score = " + str(mean(f_score)))
```

```
Precision Score = 0.891110675437585
Recall Score = 0.4624696542747241
F Score = 0.6084148331386691
```