**hw4 Machine Learning Fall 2018 | Part II - Programming | amc1354 & ads798¶**

## **Problem 5**[**¶**](#gjdgxs)

### **(a) List the 5 tokens that occur most frequently in the training set.**[**¶**](#30j0zll)

Obviously, the most frequent tokens are English stopwords and punctuation. We decided to create a more useful dataset by removing punctuation, English stopwords, and stemming words so we don't have attributes with repeated meaning. E.g., "love", "loved", "loving" would form different attributes. After stemming, they will belong to the same attribute "lov".

In [3]:

**import** **nltk**  
**from** **nltk.stem.porter** **import** PorterStemmer  
**import** **re**  
**from** **nltk.corpus** **import** stopwords  
nltk.download('stopwords')  
  
trainingcorpus = []  
**for** i **in** range(0, len(dataset)):  
 review = re.sub('[^a-zA-Z]', ' ', dataset['review'][i])  
 review = review.split()  
 ps = PorterStemmer()  
 review = [ps.stem(word) **for** word **in** review **if** **not** word **in** set(stopwords.words('english'))]  
 review = ' '.join(review)  
 trainingcorpus.append(review)  
  
token\_count = Counter(' '.join(trainingcorpus).split()).most\_common()  
  
print(token\_count[:5])

[nltk\_data] Downloading package stopwords to /Users/amc/nltk\_data...  
[nltk\_data] Package stopwords is already up-to-date!

Out[3]:

[('movi', 258), ('film', 228), ('one', 130), ('like', 85), ('n', 85)]

**(b) Calculate IG for every attribute and list the 5 attributes with the highest IG**[**¶**](#1fob9te)

In [13]:

**from** **sklearn.feature\_extraction.text** **import** CountVectorizer  
**import** **math**  
**import** **numpy** **as** **np**  
**def** entropy(var):  
 *#var is a variable having only 0,1 values*  
 N\_1 = sum(var)  
 N = len(var)  
 N\_0 = N - N\_1  
 **return** 0 **if** N\_0==0 **or** N\_1==0 **else** - N\_1/N \* math.log(N\_1/N, 2) - N\_0/N \* math.log(N\_0/N, 2)  
  
vectorizer = CountVectorizer()  
DTM = vectorizer.fit\_transform(trainingcorpus)  
*#Now, to transorm it in an occurrence matrix, where the term is not 0, we put a 1.*  
npDTM = np.array(DTM.toarray())  
npDTM[npDTM != 0] = 1  
  
labels = np.array(dataset['label'])  
  
entropy\_label = entropy(labels)  
  
features\_IG = pandas.DataFrame(data = {'feature': [], 'IG': []})  
  
**for** column **in** npDTM.T:  
 N\_1 = len(labels[column==1])  
 N\_0 = len(labels[column==0])  
 N = N\_1 + N\_0  
 IG = entropy\_label - N\_1/N \* entropy(labels[column==1]) - N\_0/N \* entropy(labels[column==0])  
 features\_IG = features\_IG.append(pandas.DataFrame([["", IG]], columns=['feature','IG']), ignore\_index=**True**)  
  
features\_IG['feature'] = vectorizer.get\_feature\_names()  
  
print(features\_IG.sort\_values('IG', ascending=**False**)[:5])

feature IG  
259 bad 0.025357  
324 best 0.020668  
2436 move 0.012183  
2716 perform 0.012123  
3886 touch 0.011337

**(c) Train Neural Net using only the 50 attributes with highest information gain**[**¶**](#3znysh7)

In [ ]:

*#select top 50 IG*