ASSIGNMENT 9

CS 432 SPRING 2017

APRIL 29, 2017

Michelle Graham

Q1:

I chose the following blog about a user's top 100 movie reviews:

http://afistop100moviesreviewed.blogspot.com/2012/04/1-citizen-kane-1941_02.html

This is the rss feed:

http://afistop100moviesreviewed.blogspot.com/feeds/posts/default?max-results=120

I made the following categories to classify the movies by genre: Action, Comedy, Crime, Drama, Fantasy, Romance, and Thriller.

Ground Truth Table Snippet:

Item	Title	Classification
#1	CITIZEN KANE	Drama
#2	THE GODFATHER	Crime
#3	CASABLANCA	Romance
#4	RAGING BULL	Drama
#5	SINGIN' IN THE RAIN	Romance
#6	GONE WITH THE WIND	Romance
#7	LAWRENCE OF ARABIA	Drama
#8	SCHINDLER'S LIST	Drama
#9	VERTIGO	Thriller
#10	THE WIZARD OF OZ	Fantasy
#11	City Lights	Romance
#12	The Searchers	Drama
#13	Star Wars Episode IV: A New Hope	Fantasy
#14	Psycho	Thriller
#15	2001: A Space Odyssey	Fantasy
#16	Sunset Blvd	Drama
#17	The Graduate	Comedy
#18	The General	Action
#19	On the Waterfront	Crime
#20	It's a Wonderful Life	Fantasy
#21	Chinatown	Drama
#22	Some Like It Hot	Romance
#23	The Grapes of Wrath	Drama
#24	E.T. the Extra-Terrestrial	Fantasy
#25	To Kill a Mockingbird	Drama

Q2:

Parsing the xml was easy for grabbing titles, however, the content was a bit more complicated. I used a series of regular expressions to get rid of a lot of the excess symbols and information that was not needed. After I successfully parsed the blogs for their titles and content, I made a combination of them for the training data. As a note, I had to shorten some of the text found in content because using all of it was causing an error with my classification. This was done by splitting the first seven sentences from each combination of content; see shortCombo.py. Most of the sentences were a consistent length, however, some were comparably shorter than others. I tried to keep it as even as possible for consistency.

Once I got all of the desired data, I made use of the given docclass.py, shortCombo.py, and a file containing the manual categorizations of movie titles, ActualCats.py, to begin training the data. This was stored in the train.py file. I collaborated with a classmate for this portion of the assignment. I made a minor edit to the docclass.py to define a simple train to classify the input data by title and category. This information is used to retrieve the classifications.

All of the classifications are stored in the Excel spreadsheets. In order to determine the true positives, I looked at which categories were successfully and accurately classified. Next, for the false positives, I looked at which categories were successful in classification, but inaccurate. The false negatives are the actual categories that are used to compare the classified data that are not accurate. For example, “Pulp Fiction” is originally classified as being a “Crime” film. Crime would be a false negative for that movie. If the trainer classified it as being “Fantasy,” this would be inaccurate. Therefore, it would have “Fantasy” as its false positive. Lastly, since the trainer did not accurately classify the movie title, it would not have a value for true positive. The precision, recall, and f-measure are found by use of the formulas provided on slide 28.

Formulas:

Precision = TP / (TP+FP)
Recall = TP / (TP+FN)
F-Measure = 2 * P*R / (P+R)

shortCombo.py:

```
import re

parsedCombined2 = open('combined.txt','r')
outfile = open('shortenedCombined.txt','w+')

for line in parsedCombined2:
    s = ''.join(sentence + '.' for sentence in re.split('\.(?=\s*(?:[A-Z]|$))', line, maxsplit=7)[:-1])
    print s
    outfile.write(s+"\n")
```

docclass.py edit:

```
def choochoo(cl,title,category):
    cl.train(title,category)
```

train.py:

```
#n*10 indicates what cross validation values are in each file. example: n=3 has values 21-30
cVal = "cValidation" + str(n*10)+".txt"
crossVal = open(cVal,"wb")
cl = docclass.fisherclassifier(docclass.getwords)

#clear database file in project after every run to ensure consistency
cl.setdb('cross.db')

#create cross 10 sublists
classified =  combined[((n-1)*10):(n*10)]
sublist2 =  combined[:(n-1)*10]
sublist3 =  combined[(n)*10:]
trainingdata_Entries = sublist2+sublist3

#Categories to be classified
#c =  categories[((n-1)*10):(n*10)]
sublist5 =  categories[:(n-1)*10]
sublist6 =  categories[(n)*10:]
trainingdata_Categories= sublist5+sublist6

while count < 90:
       docclass.choochoo(cl, trainingdata_Entries[count], trainingdata_Categories[count])
       count += 1
count =0

while count2 < 10:
   print count2
   print classified[(count2)]
   prediction = cl.classify(classified[(count2)])
   crossVal.write(prediction)
   count2 += 1
```

Results for 50:

Categories	True Positives	False Positives	False Negatives	Precision	Recall	F-Measure
Action	0	0	4	0.00	0.00	0.00
Comedy	0	0	13	0.00	0.00	0.00
Crime	1	0	6	1.00	0.14	0.25
Drama	6	26	1	0.19	0.86	0.31
Fantasy	0	3	3	0.00	0.00	0.00
Romance	0	4	5	0.00	0.00	0.00
Thriller	2	8	8	0.20	0.20	0.20
			Average	0.20	0.17	0.11

Q3:

Results for 10:

This data was found exactly like the data in question 2. The only difference is that I used 90 terms for my training data and classified 10.

Categories	True Positives	False Positives	False Negatives	Precision	Recall	F-Measure
Action	0	0	1	0.00	0.00	0.00
Comedy	0	2	1	0.00	0.00	0.00
Crime	0	0	2	0.00	0.00	0.00
Drama	0	4	2	0.00	0.00	0.00
Fantasy	0	1	2	0.00	0.00	0.00
Romance	0	0	1	0.00	0.00	0.00
Thriller	1	2	0	0.33	1.00	0.50
			Average	0.05	0.14	0.07

Note: The results for Q2 and Q3 have a low accuracy. I believe this was due to the high number of categories. This could probably yield better results if I condensed some of the categories into an “other” category and ran the code again.

Q4:

The cross.py file is used to find the cross validation of each set of 10 classifications. This is done by taking each sub list of classified terms and evaluating them as per questions 2 and 3. In other words, find their precision, recall, and f-measure. Each round of cross validation was evaluated to yield a Macro Average per round. Then, all ten averages were averaged to determine the Final Accuracy. The accuracy seems to decrease when using more terms in the training data. I feel as this is not a good result, because ideally, the accuracy should increase along with the amount of training data. The Final Accuracy, found but using the cross validation, seemed to stay within the same realm of the first trial (50 terms).

cross.py:

```
#change this variable every one to get the values 1 = 0-9; 10 = 90-99
n = 9
#n*10 indicates what cross validation values are in each file. example: n=3 has values 21-30
cVal = "cValidation" + str(n*10)+".txt"
crossVal = open(cVal,"wb")
cl = docclass.fisherclassifier(docclass.getwords)

#clear database file in project after every run to ensure consistency
cl.setdb('cross.db')

#create cross 10 sublists
classified =  combined[((n-1)*10):(n*10)]
sublist2 =  combined[:(n-1)*10]
sublist3 =  combined[(n)*10:]
trainingdata_Entries = sublist2+sublist3

#Categories to be classified
#c =  categories[((n-1)*10):(n*10)]
sublist5 =  categories[:(n-1)*10]
sublist6 =  categories[(n)*10:]
trainingdata_Categories= sublist5+sublist6

while count < 90:
      docclass.choochoo(cl, trainingdata_Entries[count], trainingdata_Categories[count])
      count += 1
count =0

while count2 < 10:
   print count2
   print classified[(count2)]
   prediction = cl.classify(classified[(count2)])
   crossVal.write(prediction)
   count2 += 1
```

Example Round- Round 1:

Round 1 (n = 10)						
Categories	True Positives	False Positives	False Negatives	Precision	Recall	F-Measure
Action	0	0	0	0.00	0.00	0.00
Comedy	0	0	0	0.00	0.00	0.00
Crime	0	0	0	0.00	0.00	0.00
Drama	2	2	4	0.50	0.33	0.40
Fantasy	1	0	0	1.00	1.00	1.00
Romance	0	3	0	0.00	0.00	0.00
Thriller	1	0	1	1.00	0.50	0.67
			Average	0.36	0.26	0.30

Final Accuracy:

10-Crossfold Validation			
Categories	Precision	Recall	F-Measure
Action	0.00	0.00	0.00
Comedy	0.05	0.03	0.04
Crime	0.05	0.10	0.07
Drama	0.44	0.24	0.31
Fantasy	0.13	0.20	0.16
Romance	0.05	0.03	0.04
Thriller	0.25	0.18	0.21
Average	0.14	0.11	0.12

Comparison:

	Precision	Recall	F-Measure
50 terms	0.20	0.17	0.11
10 terms	0.05	0.14	0.07
Cross Validation	0.14	0.11	0.12

Note: The changes in precision and f-measure seem to be inconclusive. The recall seems to decrease as the amount of training data increases.

Resources:

https://github.com/uolter/PCI/tree/master/chapter6

https://docs.python.org/2/library/re.html