Introduction to Web Science: Assignment #10

Dr. Nelson

Alexander Nwala

Thursday, December 11, 2014

Contents

Problem 1

Choose a blog or a newsfeed (or something similar as long as it has an Atom or RSS feed). It should be on a topic or topics of which you are qualified to provide classification training data. In other words, choose something that you enjoy and are knowledgable of. Find a feed with at least 100 entries.

Create between four and eight different categories for the entries in the feed:

examples:

```
- work, class, family, news, deals
- liberal, conservative, moderate, libertarian
- sports, local, financial, national, international, entertainment
- metal, electronic, ambient, folk, hip-hop, pop
```

Download and process the pages of the feed as per the week 12 class slides.

SOLUTION 1

The solution for this problem is outlined by the following steps:

1. **Select a blog:** My goal was to select a blog with entries over 100 (from Blogspot) with as much heterogenous content as possible. Given this criteria, I selected http://stevegilliard.blogspot.com/
2. **Extract entries from blog:** The Atom feed of the blog was extracted and saved into an XML file due to Listing 1. 300 entries were collected in order to have enough data in the event of errors.

Listing 1: Save Blog Entries to XML file

```
#get blog entries
'''
     input:
          blogUrl:
               'http://blogNameHere.blogspot.com/'
          outputXMLFileName:
               'blogEntries.xml'
'''
def getXMLEntriesFromBlog(blogUrl, outputXMLFileName, maximumEntries=10):

     if( len(blogUrl) > 0 and len(outputXMLFileName) > 0 and maximumEntries > 0 ):
          maximumEntries = str(maximumEntries)
          blogUrl = blogUrl.strip()
          try:
               co = 'curl -s ' + blogUrl + 'feeds/posts/default?max-results='+
                   maximumEntries
               output = commands.getoutput(co)

               outputFile = open(outputXMLFileName, 'w')
               outputFile.write(output)
               outputFile.close()
          except:
```

Table 1: Distribution of Classes

ITEM	Class	First 50	Second 50
1	Politics	26	13
2	Story	15	22
3	Law	3	6
4	Misc	6	9

```
            print 'Failed to parse feed %s' % blogUrl
            exc_type, exc_obj, exc_tb = sys.exc_info()
            fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
25          print(fname, exc_tb.tb_lineno, sys.exc_info() )
```

3. **Manually classify all entries:** After inspecting the blog entries, I chose 4 classes:

```
Politics - For Politics related entries
Story - For News and narratives on different topics
Law - For Crime and Law enforcement related entries
Misc - For topics which clearly do not belong to the previous classes
```

4. **Partition data into 2 sets:** The 100 entries from the blog were split into 2 partitions (first 50 and last 50 entries). Table 1. is a summary of the class distribution for the 100 blog entries of the 2 sets.

```
The file 50nBlogTrainingDataSource.txt contains the first set of:
    <BlogEntryTitle, BlogEntrySummary, UserAssignedClassLabel>
The file 50nBlogTestDataSource.txt contains the second set of:
    <BlogEntryTitle, BlogEntrySummary, UserAssignedClassLabel>
```

Problem 2

Manually classify the first 50 entries, and then classify (using the fisher classifier) the remaining 50 entries. Report the cprob() values for the 50 titles as well. From the title or entry itself, specify the 1-, 2-, or 3-gram that you used for the string to classify. Do not repeat strings; you will have 50 unique strings. For example, in these titles the string used is marked with *s:

Rachel Goswell - “Waves Are Universal” (LP Review)
The *Naked and Famous* - “Passive Me, Aggressive You” (LP Review)
Negativland* - “Live at Lewis’s, Norfolk VA, November 21, 1992” (concert)
Negativland - “*U2*” (LP Review)

Note how “Negativland” is not repeated as a classification string.

Create a table with the title, the string used for classification, cprob(), predicted category, and actual category.

SOLUTION 2

The solution for this problem is outlined by the following steps:

1. **Train model:** Listing 2. (myFisherModelInTrainingAndTesting) utilizes a modified version of the PCI book's [2] feedfilter.py read method (nonInteractiveRead). The modification facilitates automated testing and training (not user-interactive) by operating on an input file which contains the user supplied class labels.

   Given that the same function is used for training and testing, a call to myFisherModelInTrainingAndTesting is made supplying the "train" argument. The result of the training is written into a database file (stevegilliard.db)

Listing 2: Fisher Model

```
#Fisher model training and testing
'''
     input: trainingInputFileName.txt, entriesXMLFileName.xml, ['test'|'train'],
         stopAtCount, 'getWord'|'getEntry'
'''
def myFisherModelInTrainingAndTesting(trainingInputFileName, entriesXMLFileName,
    dbFileName, mode, maxItems, getWordGetEntryMethod='getWord'):

     if( (mode == 'train' or mode == 'test') and (getWordGetEntryMethod == '
         getWord' or getWordGetEntryMethod == 'getEntry') ):

          if( len(trainingInputFileName) > 0 and len(entriesXMLFileName) > 0 and
              len(dbFileName) > 0 and maxItems > 0 ):

               if( getWordGetEntryMethod == 'getWord' ):
                    cl=docclass.fisherclassifier(docclass.getwords)
               else:
                    cl=docclass.fisherclassifier(feedfilter.entryfeatures)

               cl.setdb(dbFileName)
               feedfilter.nonInteractiveRead(entriesXMLFileName, cl,
                   trainingInputFileName, mode, maxItems, getWordGetEntryMethod)

'''
  input: feed.xml, fisherclassifier, trainingAndTestDataFileName.txt, ['train'|'
      test'], stopAtVCount, 'getWord'|'getEntry'
'''
def nonInteractiveRead(feed, classifier, trainingAndTestDataFileName, mode,
    maxItems, getWordGetEntryMethod='getWord'):

  if( len(feed) > 0 and len(trainingAndTestDataFileName) > 0 and (mode == 'train'
      or mode == 'test') and maxItems > 0 and (getWordGetEntryMethod == 'getWord'
      or getWordGetEntryMethod == 'getEntry')):
...
    if( title.lower() == entry['title'].strip().lower() ):

    #print 'Title:     '+entry['title'].encode('utf-8')
    fulltext='%s\n%s' % (entry['title'],entry['summary'])

```

```
    if( mode == 'train' ):
      #training get the correct category and train on that

      if( getWordGetEntryMethod == 'getWord' ):
        classifier.train(fulltext, actualClassLabel)
      else:
       classifier.train(entry, actualClassLabel)

      print '...training model', count
    else:
      #testing: guess the best guess at the current category
      try:
        if( getWordGetEntryMethod == 'getWord' ):
          prediction = str(classifier.classify(fulltext))
        else:
          prediction = str(classifier.classify(entry))
      except:
        print '...skipping', count
        continue

      cProbValue = classifier.getGlobalCProbValue()
...
```

2. **Test model:** Utilizing the model learned during training, the same function (myFisherModelInTrainingAndTesting) was called to test the model by supplying the "test" argument. Table 2. summarizes the result of the test. The string used for testing was the blog contents (summaries) and is omitted from the table due to its length. Also Table 3. shows the first 50 entries used to train the model. Table 2. is three entries short due to absent weighted probability values associated with three entries.

Problem 3

Assess the performance of your classifier in each of your categories by computing precision, recall, and F1. Note that the definitions of precisions and recall are slightly different in the context of classification; see:
http://en.wikipedia.org/wiki/Precision_and_recall#Definition_.28classification_context.29
and
http://en.wikipedia.org/wiki/F1_score

SOLUTION 3

The solution for this problem is outlined by the following steps:

1. **Generate confusion matrix and calculate precision/recall/F1 measures:** This was achieved due to Listing 3. which utilized the scikit-learn Machine Learning in Python library [1]

Listing 3: Calculate Precision

```
#calculate precision, recall, f1 score, and plot confusion matrix
predictionFileName = '50nBlogTestDataSourcePredictions.txt'
```

Table 2: Fisher Method Test Result For getword Function

ITEM	ENTRY-TITLE	CPROB	PRED-LABEL	ACTUAL-LABEL
1	Mr. Sulu Phasers Tim Hardaway	0.2500	story	story
2	Too close for comfort	0.3693	politics	politics
3	Middle School?	0.2500	politics	politics
4	No kids for gays	0.3147	politics	politics
5	HDTV	0.3693	story	story
6	Change sometimes isn't so fast	0.2500	story	story
7	More idiocy	0.2500	politics	politics
8	Yawn, Hillary's In	0.2685	politics	politics
9	Scumbags in action	0.2500	politics	politics
10	The battle of the exploding pigs	0.3693	politics	politics
11	Racial Row on UK show	0.3693	politics	story
12	No spanking	0.3147	politics	story
13	Provoking Sadr	0.6018	politics	politics
14	Jenna writes a book	0.2500	story	misc
15	Only in Washington	0.5058	politics	politics
16	Oooops	0.2500	politics	story
17	Another 9/11 tragedy	0.3693	story	story
18	Time to serve	0.2500	politics	misc
19	About going to Walter Reed	0.3693	story	story
20	Voir Dire from hell	0.8965	politics	law
21	Simple point	0.3693	misc	misc
22	Fading away	0.8227	politics	politics
23	We're playing you	0.8333	politics	story
24	Libby on trial	0.3147	politics	law
25	The Car Wreck of American TV	0.3693	politics	misc
26	No more actors	0.2500	story	story
27	Good luck, Jane	0.2689	politics	misc
28	More dead in Iraq	0.6132	politics	law
29	The good old days my ass	0.1575	politics	politics
30	A simple question	0.2500	politics	politics
31	What bubble?	0.3147	politics	story
32	Idiocy in action	0.2500	story	story
33	Yeah, attacking the Mahdi Army will happen	0.5855	politics	politics
34	What can I do?	0.5855	politics	story
35	Marine murdered for insurance?	0.8965	story	law
36	Hey, it just happened	0.3147	politics	story
37	Why the fuck is he making military policy	0.2500	politics	politics
38	You have to be kidding	0.2500	politics	story
39	Bush will lose the country	0.3693	politics	politics
40	Pointless	0.2500	politics	story
41	Better to drop FAE's instead	0.2500	politics	story
42	How stupid can you be	0.2500	story	story
43	Madness in action	0.3147	politics	politics
44	They don't want us	0.2500	politics	politics
45	Sucker	0.2500	politics	story
46	Silicone	0.3693	politics	misc
47	Idiocy in action	0.2012	politics	story

Problem 3 continued on next page...

Table 3: Fisher Method Training Dataset

ITEM	ENTRY-TITLE	ACTUAL-LABEL
1	I told you this would happen	politics
2	He has affliliations	politics
3	Reality Check	politics
4	Bush's plan falling apart	politics
5	A history lesson	politics
6	This makes me laugh	misc
7	Talk or no talk	misc
8	A nervous WH	politics
9	www.thenewsblog.net	misc
10	Thank you, Jesus	story
11	Shut yer festerin' gob	politics
12	Give me back my airmen	politics
13	Max responds	politics
14	Here we go again	law
15	Favorite Egg recipe	misc
16	WH lies	politics
17	I don't know	story
18	She wanted unemployment?	story
19	Racist pig speaks	story
20	Bush's fantasyland	politics
21	Weird weather cures	story
22	Idiot	story
23	Are you kidding?	politics
24	Bwaaaah	politics
25	The war was wrong	politics
26	Fucking Moron	politics
27	When the law doesn't work, try blackmail	politics
28	We oppose the war	politics
29	Black life means nothing to Lieberman	politics
30	The Hamilton Rule	politics
31	Where is Hillary?	politics
32	So who did this?	politics
33	Dear Max	misc
34	Nothing like Vietnam	story
35	A real GI Bill	politics
36	Jesus	story
37	Be greatful, Iraqis	politics
38	This isn't in the plans for the Kurds	politics
39	Duh	story
40	Beckham in America	misc
41	What are people afraid of?	story
42	Gunpoint at Democracy	law
43	No kidding	politics
44	You bet your life	story
45	Leave Iran Alone	story
46	Failure	story
47	Spocko v Disney	story
48	Yes, you need to apologize, you goddamn redneck bastard	story
49	Mercenary murder in Iraq?	law
50	Attacking Iran	politics

Problem 3 continued on next page...

Table 4: Precision/Recall/F1 Measures 50/50 Split With getword Function

ITEM	Class	Precision	Recall	F1
1	Politics	0.3333	1.0000	0.5000
2	Story	0.5000	0.2500	0.3333
3	Law	0.0000	0.0000	0.0000
4	Misc	1.0000	0.1111	0.2000

```
listOfPredictedLabels, listOfActualLabels = getListOfPredictedAndActualLabels(
    predictionFileName)

labels = ['politics', 'story', 'law', 'misc']
confusionMatrix = confusion_matrix(listOfActualLabels, listOfPredictedLabels,
    labels=labels)
precisionScore = precision_score(listOfActualLabels, listOfPredictedLabels,
    average=None)
recallScore = recall_score(listOfActualLabels, listOfPredictedLabels, average=None
    )
f1Score = f1_score(listOfActualLabels, listOfPredictedLabels, average=None)

print confusionMatrix
print precisionScore
print recallScore
print f1Score

fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(confusionMatrix)
plt.title('Confusion Matrix of the Fischer Classifier')
fig.colorbar(cax)
ax.set_xticklabels(['']  + labels)
ax.set_yticklabels(['']  + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

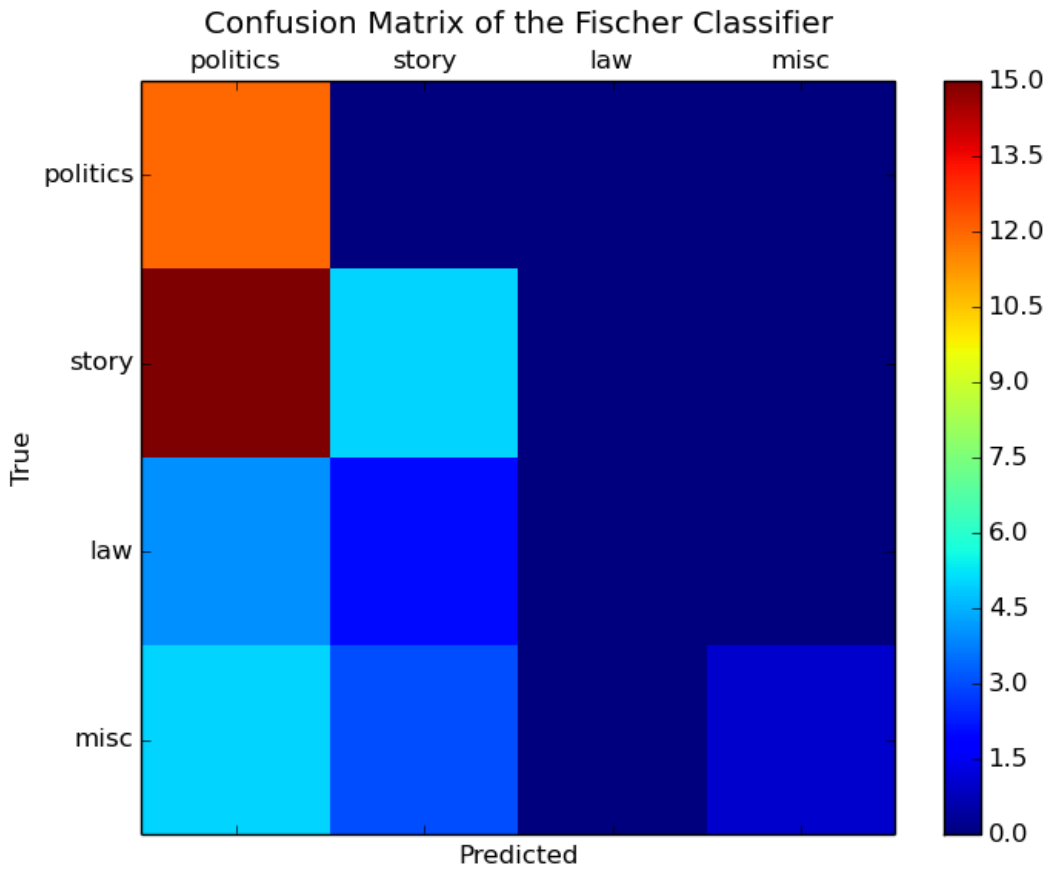
Figure 1. outlines the confusion matrix due to Listing 3.

The following represent the numeric values corresponding to the confusion matrix. Table 4. outlines the precision of the various classes. The sum of true positives and false positives are equal to zero for some labels such as Law, this is reason why the precision/recall/F1 is ill defined for the Law class.

```
   p   s  l  m
p [12  0  0  0]
s [15  5  0  0]
l [ 4  2  0  0]
m [ 5  3  0  1]

p = Politics, s = Story, l = Law, m = Misc
```

Figure 1: Confusion Matrix For 50/50 Split



Problem 4

Redo questions 2 & 3, but with manually train 90 entries and then classify the remaining 10.

Then redo questions 2 & 3, but with the extensions on slide 26 and pp. 136–138. Fully discuss the changes you've made.

Which method (more training vs. better features) gave better improvement over your baseline? Why do you think that is?

SOLUTION 4

The solution for this problem is outlined by the following steps:

1. **Train and test with 90/10 split:** This was achieved in the same manner as the 50/50 case (50 training dataset, 50 dataset), but this time with 90 entries for training and 10 entries for test. In

Table 5: Precision/Recall/F1 Measures For 90/10 Split

ITEM	Class	Precision	Recall	F1
1	Politics	0.1667	1.0000	0.2857
2	Story	0.3333	0.2500	0.2857
3	Law	0.0000	0.0000	0.0000
4	Misc	1.0000	0.1111	0.3333

Table 6: Fisher Method Test Result For 90/10 Split

ITEM	ENTRY-TITLE	CPROB	PRED-LABEL	ACTUAL-LABEL
1	Mr. Sulu Phasers Tim Hardaway	0.2500	story	misc
2	Despite total ignorance, I support this idea	0.2695	politics	story
3	Time to serve	0.2500	politics	misc
4	Simple point	0.1062	misc	misc
5	We're playing you	0.4857	story	story
6	Good luck, Jane	0.2659	politics	misc
7	A simple question	0.2500	politics	politics
8	Pointless	0.3345	politics	story
9	Sucker	0.2500	politics	story
10	Silicone	0.1945	story	misc

addition, Table 5. outlines the precision/recall/F1 scores for each blog entry while Table 6. outlines the cprob/predition/actual label for each entry. And below is the confusion matrix.

```
   p s l m
p [1 0 0 0]
s [3 1 0 0]
l [0 0 0 0]
m [2 2 0 1]

p = Politics, s = Story, l = Law, m = Misc
```

The sum of true positives and false positives are equal to zero for some labels (Law), thus precision/recall/F1 are ill defined.

2. **Train and test with 50/50 split (improved feature):** The first 50/50 split utilized the getword function for creating the list of features (nonalphanumeric split to break up the words). The changes carried out in this train/test operation included in the getfeature function includes the extraction of title words, count of uppercase words and extraction of word pairs. These changes are included in myFisherModelInTrainingAndTesting and nonInteraciveRead (Listing 2.) through the passing of the "getEntry" argument.

   Table 8. outlines the cprop and the prediction/actual labels. Table 7. outlines the precision/recall/F1 measure, Figure 3. outlines the confusion matrix due to the foregoing operation. And below is the corresponding confusion matrix.

```
    p   s  l  m
p [12  0  0  0]
```

Figure 2: Confusion Matrix For 90/10 Split

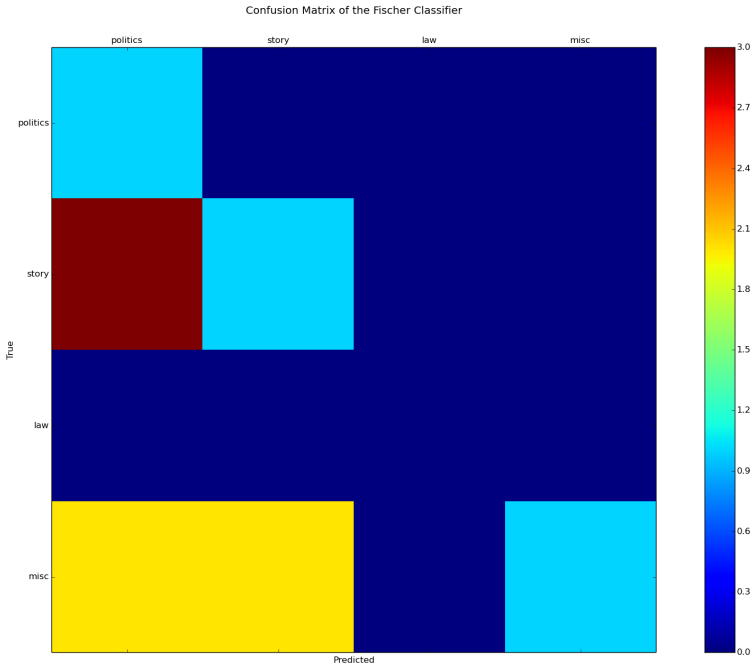


Table 7: Precision/Recall/F1 Measures For 50/50 Split With getfeature Function

ITEM	Class	Precision	Recall	F1
1	Politics	0.3243	1.0000	0.4897
2	Story	0.5000	0.2500	0.3333
3	Law	0.0000	0.0000	0.0000
4	Misc	0.0000	0.0000	0.0000

```
s [15  5  0  0]
l [ 4  2  0  0]
m [ 6  3  0  0]

 p = Politics, s = Story, l = Law, m = Misc
```

The sum of true positives and false positives are equal to zero for some labels (Law and Misc), thus the measures are ill defined.

3. **Comparison; which is better getword or getfeature?** Given that some labels have 0 value, only the labels which are present in both models were compared:

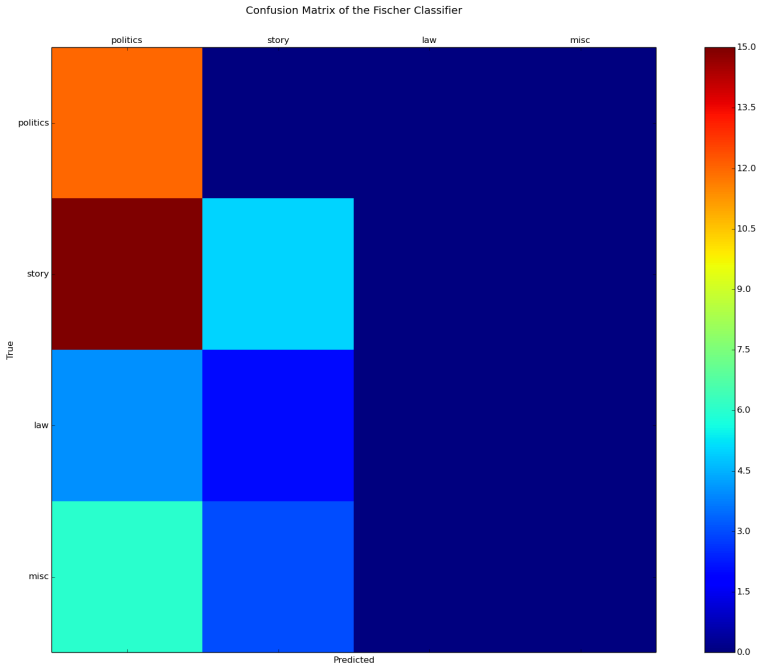
```
<Precision (getword), Precision (getfeature)>
<Politics: 0.3333, Politics: 0.3243>
<Story: 0.5000, Story: 0.5000>
```

Table 8: Fisher Method Test Result For getfeature Function

ITEM	ENTRY-TITLE	CPROB	PRED-LABEL	ACTUAL-LABEL
1	Mr. Sulu Phasers Tim Hardaway	0.2500	story	misc
2	Too close for comfort	0.3693	politics	story
3	Middle School?	0.2500	politics	story
4	No kids for gays	0.3147	politics	story
5	HDTV	0.3693	story	misc
6	Change sometimes isn't so fast	0.2500	story	law
7	More idiocy	0.2500	politics	politics
8	Yawn, Hillary's In	0.2838	politics	politics
9	Scumbags in action	0.2500	politics	law
10	The battle of the exploding pigs	0.3693	politics	misc
11	Racial Row on UK show	0.3693	politics	story
12	No spanking	0.3147	politics	story
13	Provoking Sadr	0.6018	politics	politics
14	Jenna writes a book	0.2500	story	misc
15	Only in Washington	0.5058	politics	politics
16	Oooops	0.2500	politics	story
17	Another 9/11 tragedy	0.3693	story	story
18	Time to serve	0.2500	politics	misc
19	About going to Walter Reed	0.3693	story	story
20	Voir Dire from hell	0.8965	politics	law
21	Simple point	0.3693	misc	misc
22	Fading away	0.8227	politics	politics
23	We're playing you	0.8333	politics	story
24	Libby on trial	0.3147	politics	law
25	The Car Wreck of American TV	0.3693	politics	misc
26	No more actors	0.2500	story	story
27	Good luck, Jane	0.2689	politics	misc
28	More dead in Iraq	0.6132	politics	law
29	The good old days my ass	0.1575	politics	politics
30	A simple question	0.8965	politics	law
31	What bubble?	0.3693	politics	misc
32	Idiocy in action	0.8227	politics	politics
33	Yeah, attacking the Mahdi Army will happen	0.8333	politics	story
34	What can I do?	0.3147	politics	law
35	Marine murdered for insurance?	0.3693	politics	misc
36	Hey, it just happened	0.2500	story	story
37	Why the fuck is he making military policy	0.2689	politics	misc
38	You have to be kidding	0.6132	politics	law
39	Bush will lose the country	0.1658	politics	politics
40	Pointless	0.2500	politics	story
41	Better to drop FAE's instead	0.2500	politics	story
42	How stupid can you be	0.2500	story	story
43	Madness in action	0.3147	politics	politics
44	They don't want us	0.2500	politics	politics
45	Sucker	0.2500	politics	story
46	Silicone	0.3693	politics	misc
47	Idiocy in action	0.2012	politics	story

Problem 4 continued on next page...

Figure 3: Confusion Matrix For 50/50 Split With getfeature Function




```
<Recall (getword), Recall (getfeature)>
<Politics: 1.0000, Politics: 1.0000>
<Story: 0.2500, Story: 0.2500>

<F1 (getword), F1 (getfeature)>
<Politics: 0.5000, Politics: 0.4800>
<Story: 0.3333, Story: 0.5000>
```

Given that the measures marginally differ, this dataset does not favor any model. However, getfeature is a more sensible way of document classification since is considers more features, as opposed to the monolithic view of getwords.

References

[1] sci-kit learn. http://scikit-learn.org/stable/. Accessed: 2014-12-11.

[2] Toby Segaran. Programming Collective Intelligence, 2007.