sad-python3

March 22, 2017

In [1]:
```
import logging
import sys

log = logging.getLogger("P/R/ROC")
log.setLevel(logging.INFO)
ch = logging.StreamHandler()
ch.setLevel(logging.INFO)
formatter = logging.Formatter("%(asctime)s %(levelname)s:%(name)s %(message)s")
ch.setFormatter(formatter)

log.addHandler(ch)

log.info("Now logging")
```

```
2017-03-21 20:01:57,470 INFO:P/R/ROC Now logging
```

In [2]:
```
# MAX_TWEETS = 1_578_628
MAX_TWEETS = 100_001

# http://www.cs.ou.edu/~cgrant/teaching/cs5970sp16/sad/sad.html

import io
feature_list = []
with io.open("sad.thorn", "r", encoding='utf-8') as source:
    labels = []
    for i,line in enumerate(source.readlines()):
        if i == MAX_TWEETS: break
        elif i == 0:
            labels = line.split('þ')
        else:
            data = line.split('þ')
            feature_list.append({'id': int(data[0]),
                                 'label': '+' if (data[1] == '1') else '-',
                                 'text' : data[2]})
#print(feature_list[0:10])
```

```
        pos_tweets = [x for x in feature_list if x['label'] == '+']
        neg_tweets = [x for x in feature_list if x['label'] == '-']

        log.info("pos: {}, neg: {}".format(len(pos_tweets), len(neg_tweets)))
```

```
2017-03-21 20:01:58,524 INFO:P/R/ROC pos: 56462, neg: 43538
```

In [3]:
```
        # Create big [({feature}, label)]
        import nltk
        from nltk import word_tokenize

        import collections
        import nltk.util
        from nltk.util import ngrams

        GRAM_SIZE = 1

        # postive feature list
        pos_features = []
        for item in pos_tweets:
            for words in ngrams(word_tokenize(item['text']), GRAM_SIZE):
                word = ' '.join(words)
                pos_features.append({'feature': word})

        neg_features = []
        for item in neg_tweets:
            for words in ngrams(word_tokenize(item['text']), GRAM_SIZE):
                word = ' '.join(words)
                neg_features.append({'feature': word})

        # Remove stop words
        from nltk.corpus import stopwords
        stopwords = stopwords.words('english')
        #print(stopwords)

        # TODO filter tweets by stopwords
        pos_features = [x for x in pos_features if x["feature"] not in stopwords]
        neg_features = [x for x in neg_features if x["feature"] not in stopwords]
```

In [4]:
```
        import random
        from random import shuffle

        SPLIT = 0.75

        shuffle(pos_features)
        shuffle(neg_features)
```

```
        POS_TRAIN_SIZE = int(len(pos_features) * SPLIT)
        NEG_TRAIN_SIZE = int(len(neg_features) * SPLIT)

        train_set = [(feat, '+') for feat in pos_features[:POS_TRAIN_SIZE]]
        train_set += [(feat, '-') for feat in neg_features[:NEG_TRAIN_SIZE]]

        test_set = [(feat, '+') for feat in pos_features[POS_TRAIN_SIZE:]]
        test_set += [(feat, '-') for feat in neg_features[NEG_TRAIN_SIZE:]]

        shuffle(test_set)
        shuffle(train_set)
```

0.0.1 Naive Bayes Classifier

```
In [5]: from nltk.classify import NaiveBayesClassifier

        log.info("Training NB")
        nb_classifier = NaiveBayesClassifier.train(train_set)
        log.info("Finished Training NB. classes: {}".format(nb_classifier.labels()))
```

```
2017-03-21 20:02:31,957 INFO:P/R/ROC Training NB
2017-03-21 20:02:34,154 INFO:P/R/ROC Finished Training NB. classes: ['+', '-']
```

0.0.2 Decision Tree Classifier

```
In [6]: from nltk.classify import DecisionTreeClassifier

        log.info("Training DT")
        dt_classifier = DecisionTreeClassifier.train(train_set)
        log.info("Finished Training DT. classes: {}".format(dt_classifier.labels()))
```

```
2017-03-21 20:02:34,161 INFO:P/R/ROC Training DT
```

```
        ---------------------------------------------------------------------------

        KeyboardInterrupt                         Traceback (most recent call last)

        <ipython-input-6-2ed8c7eef75b> in <module>()
          2
          3 log.info("Training DT")
    ----> 4 dt_classifier = DecisionTreeClassifier.train(train_set)
          5 log.info("Finished Training DT. classes: {}".format(dt_classifier.labels()))


        /usr/local/lib/python3.6/site-packages/nltk/classify/decisiontree.py in train(labeled_fe
        159         # Refine the stump.
```

```
    160         tree.refine(labeled_featuresets, entropy_cutoff, depth_cutoff-1,
--> 161                     support_cutoff, binary, feature_values, verbose)
    162
    163         # Return it


/usr/local/lib/python3.6/site-packages/nltk/classify/decisiontree.py in refine(self, lab
    193         for fval in self._decisions:
    194             fval_featuresets = [(featureset, label) for (featureset, label)
--> 195                                 in labeled_featuresets
    196                                 if featureset.get(self._fname) == fval]
    197


/usr/local/lib/python3.6/site-packages/nltk/classify/decisiontree.py in <listcomp>(.0)
    192         if depth_cutoff <= 0: return
    193         for fval in self._decisions:
--> 194             fval_featuresets = [(featureset, label) for (featureset, label)
    195                                 in labeled_featuresets
    196                                 if featureset.get(self._fname) == fval]


KeyboardInterrupt:
```

0.0.3 MaxentClassifier

In [7]:
```
from nltk.classify import MaxentClassifier

        log.info("Training M")
        m_classifier = MaxentClassifier.train(train_set)
        log.info("Finished Training M. classes: {}".format(m_classifier.labels()))
```

```
2017-03-22 09:37:08,368 INFO:P/R/ROC Training M


  ==> Training (100 iterations)

      Iteration    Log Likelihood    Accuracy
      ---------------------------------------
             1          -0.69315        0.438
             2          -0.59092        0.654
             3          -0.57451        0.654
             4          -0.56479        0.654
             5          -0.55836        0.654
             6          -0.55380        0.654
             7          -0.55039        0.654
             8          -0.54774        0.654
```

9	-0.54563	0.654
10	-0.54391	0.654
11	-0.54248	0.654
12	-0.54127	0.654
13	-0.54023	0.654
14	-0.53933	0.654
15	-0.53854	0.654
16	-0.53785	0.654
17	-0.53724	0.654
18	-0.53669	0.654
19	-0.53619	0.654
20	-0.53574	0.654
21	-0.53534	0.654
22	-0.53497	0.654
23	-0.53463	0.654
24	-0.53431	0.654
25	-0.53403	0.654
26	-0.53376	0.654
27	-0.53351	0.654
28	-0.53328	0.654
29	-0.53306	0.654
30	-0.53286	0.654
31	-0.53267	0.654
32	-0.53250	0.654
33	-0.53233	0.654
34	-0.53217	0.654
35	-0.53202	0.654
36	-0.53188	0.654
37	-0.53175	0.654
38	-0.53162	0.654
39	-0.53150	0.654
40	-0.53139	0.654
41	-0.53128	0.654
42	-0.53118	0.654
43	-0.53108	0.654
44	-0.53098	0.654
45	-0.53089	0.654
46	-0.53081	0.654
47	-0.53072	0.654
48	-0.53065	0.654
49	-0.53057	0.654
50	-0.53050	0.654
51	-0.53043	0.654
52	-0.53036	0.654
53	-0.53029	0.654
54	-0.53023	0.654
55	-0.53017	0.654
56	-0.53011	0.654

```
        57          -0.53005        0.654
        58          -0.53000        0.654
        59          -0.52995        0.654
        60          -0.52990        0.654
        61          -0.52985        0.654
        62          -0.52980        0.654
        63          -0.52975        0.654
        64          -0.52971        0.654
        65          -0.52966        0.654
        66          -0.52962        0.654
        67          -0.52958        0.654
        68          -0.52954        0.654
        69          -0.52950        0.654
        70          -0.52946        0.654
        71          -0.52943        0.654
        72          -0.52939        0.654
        73          -0.52936        0.654
        74          -0.52932        0.654
        75          -0.52929        0.654
        76          -0.52926        0.654
        77          -0.52923        0.654
        78          -0.52920        0.654
        79          -0.52917        0.654
        80          -0.52914        0.654
        81          -0.52911        0.654
        82          -0.52908        0.654
        83          -0.52906        0.654
        84          -0.52903        0.654
        85          -0.52901        0.654
        86          -0.52898        0.654
        87          -0.52896        0.654
        88          -0.52893        0.654
        89          -0.52891        0.654
        90          -0.52889        0.654
        91          -0.52886        0.654
        92          -0.52884        0.654
        93          -0.52882        0.654
        94          -0.52880        0.654
        95          -0.52878        0.654
        96          -0.52876        0.654
        97          -0.52874        0.654
        98          -0.52872        0.654
        99          -0.52870        0.654

2017-03-22 11:18:43,912 INFO:P/R/ROC Finished Training M. classes: ['-', '+']

     Final          -0.52868        0.654
```

0.0.4 Get precision recall f1 measure of each (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html)

```
In [8]: import sklearn
        import sklearn.metrics
        from sklearn.metrics import precision_recall_fscore_support

        import numpy as np
        from operator import itemgetter

        def prfs(classify, test_set):
            truth = np.array([b for (a,b) in test_set])
            prediction = np.array([classify(a) for (a,b) in test_set])
            return precision_recall_fscore_support(truth, prediction)
```

0.0.5 Run the Naive Bayes Classier

```
In [9]: p,r,f1,_ = prfs(nb_classifier.classify, test_set)
        log.info("nb precision: {}".format(p))
        log.info("nb recall: {}".format(r))
        log.info("nb f1: {}".format(f1))
```

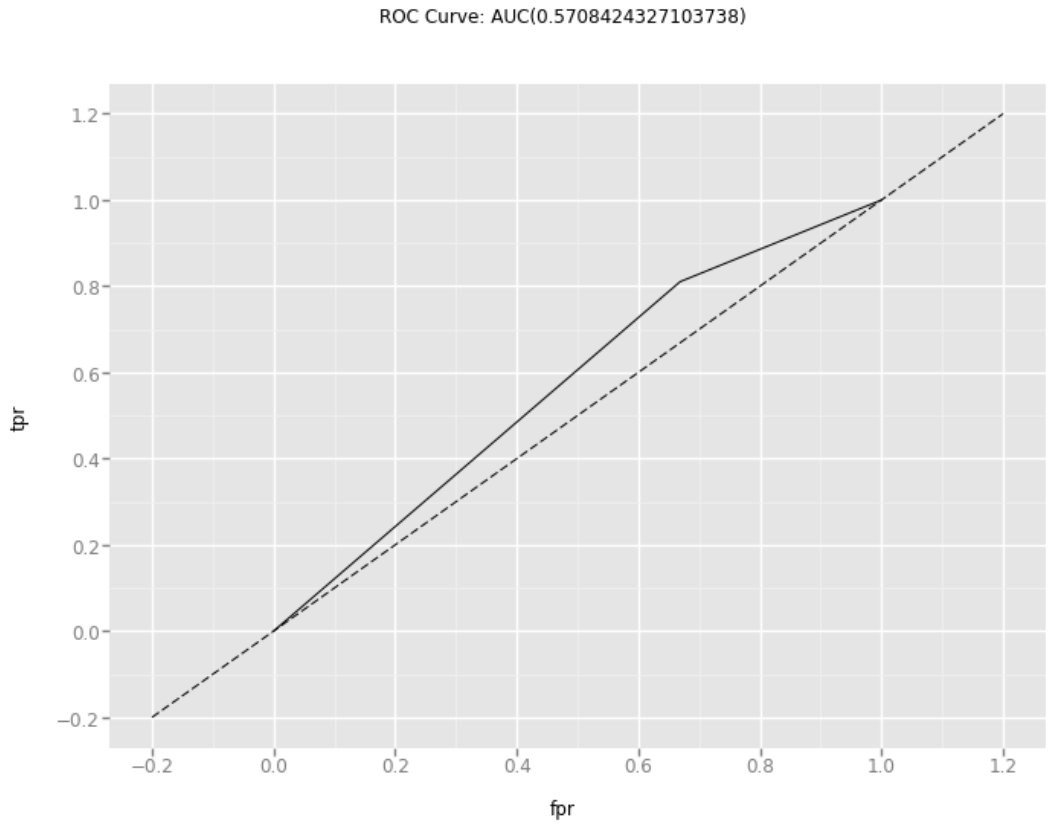
```
2017-03-22 11:24:29,577 INFO:P/R/ROC nb precision: [ 0.60902948  0.57620809]
2017-03-22 11:24:29,578 INFO:P/R/ROC nb recall: [ 0.81051475  0.33117012]
2017-03-22 11:24:29,581 INFO:P/R/ROC nb f1: [ 0.69547305  0.42060278]
```

0.0.6 Plotting the ROC curve (http://blog.yhat.com/posts/roc-curves.html)

```
In [10]: from sklearn.metrics import roc_curve
         from sklearn.metrics import auc
         import pandas as pd
         import ggplot
         from ggplot import aes
         from ggplot import geom_abline
         from ggplot import geom_line
         from ggplot import ggplot
         from ggplot import ggtitle
         import matplotlib.pyplot as plt

         # An iPython directive for inline graphs
         %matplotlib inline

         def plus2one(x): return 1 if x == '+' else 0

         def plot(classify, test_set):
             truth = [plus2one(b) for (a,b) in test_set]
```

```
        data = [plus2one(classify(a)) for (a,b) in test_set]
        fpr, tpr, thresholds = roc_curve(truth, data, drop_intermediate=False)
        roc_auc = auc(fpr, tpr)

        df = pd.DataFrame(dict(fpr=fpr, tpr=tpr))
        g = ggplot(df, aes(x='fpr', y='tpr' )) + geom_line() + geom_abline(linetype='dashed
        g += ggtitle("ROC Curve: AUC({})".format(roc_auc))
        return g

    def matplot(classify, test_set):
        truth = [plus2one(b) for (a,b) in test_set]
        data = [plus2one(classify(a)) for (a,b) in test_set]
        fpr, tpr, thresholds = roc_curve(truth, data, drop_intermediate=False)

        roc_auc = auc(fpr, tpr)
        plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.legend(loc="lower right")
        return plt
```

In [13]:
```
plot(nb_classifier.classify, test_set)
```



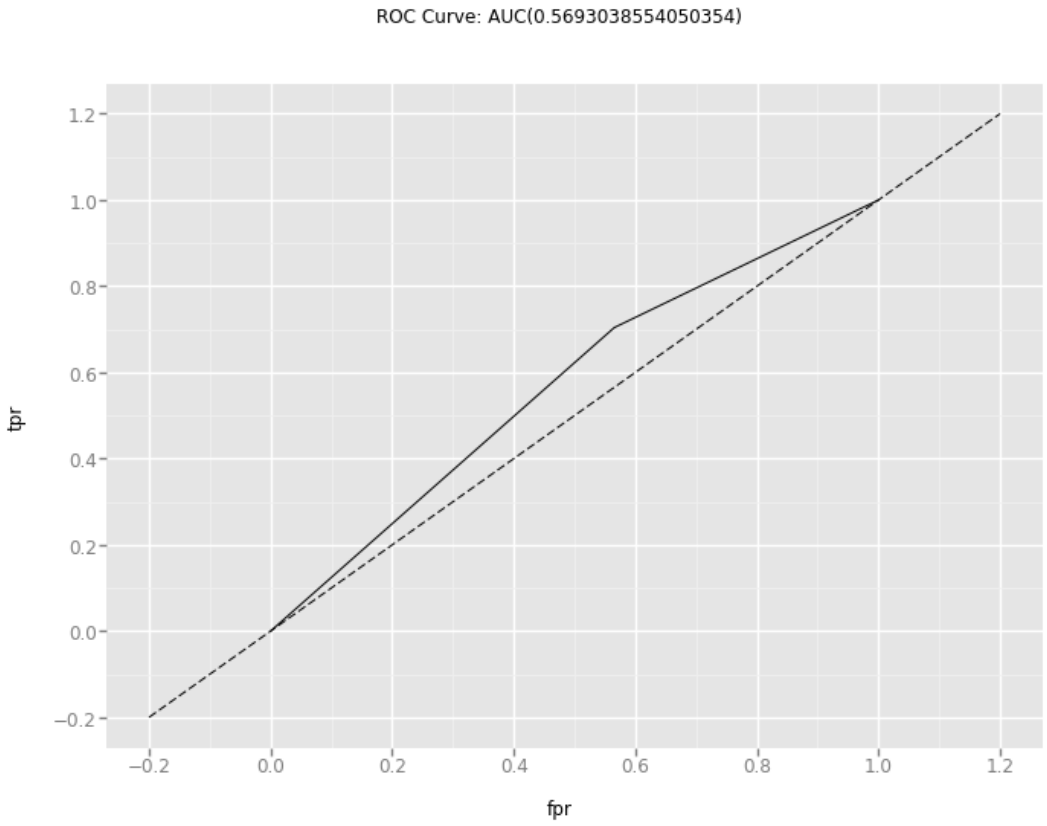
```
Out[13]: <ggplot: (320277708)>
```

0.0.7 Run the Decision Tree Classifier

```
In [ ]: p,r,f1,_ = prfs(dt_classifier.classify, test_set)
        log.info("dt precision: {}".format(p))
        log.info("dt recall: {}".format(r))
        log.info("dt f1: {}".format(f1))

In [ ]: plot(dt_classifier.classify, test_set)
```

0.0.8 Run the Maxent Classifier

```
In [11]: p,r,f1,_ = prfs(m_classifier.classify, test_set)
         log.info("m precision: {}".format(p))
         log.info("m recall: {}".format(r))
         log.info("m f1: {}".format(f1))

2017-03-22 11:24:40,572 INFO:P/R/ROC m precision: [ 0.61546108  0.53321895]
2017-03-22 11:24:40,573 INFO:P/R/ROC m recall: [ 0.70407625  0.43453146]
2017-03-22 11:24:40,574 INFO:P/R/ROC m f1: [ 0.65679313  0.47884332]


In [12]: plot(m_classifier.classify, test_set)
```



```
Out[12]: <ggplot: (-9223372036538259452)>
```

```
In [ ]:
```