In [86]:
```
import numpy
```

In [87]:
```
train = numpy.loadtxt("pa3train.txt")
test = numpy.loadtxt("pa3test.txt")
feature_len = len(train[0])-1
```

In [88]:
```
def sign(number):
    return 1 if number >= 0 else -1
```

In [89]:
```
def train_percepie_boy(train, passes, label_one):
    w = numpy.zeros(feature_len)

    for p in range(passes):
        for point in train:
            label = 1 if label_one == point[feature_len] else -1
            if  label * numpy.dot(w, point[:feature_len]) <= 0:
                w = w + (label * point[:feature_len])

    return w
```

In [90]:
```
def perceptron_error(test, w, label):
    wrong = 0
    for point in test:
        sign = numpy.dot(w, point[:feature_len])
        if  sign < 0 and point[feature_len] == label:
            wrong += 1
        elif sign >= 0 and point[feature_len] != label:
            wrong += 1

    return wrong/len(test)
```

In [91]:
```
def train_voted_perceptron(train, passes, label_one):
    w = numpy.zeros(feature_len)
    c = 1
    classifiers = []
    for p in range(passes):
        for point in train:
            label = 1 if label_one == point[feature_len] else -1
            if  label * numpy.dot(w, point[:feature_len]) <= 0:
                classifiers.append((w, c))
                w = w + (label * point[:feature_len])
                c = 1
            else:
                c += 1

    classifiers.append((w, c))
    return classifiers
```

In [92]:
```
def voted_perceptron_error(test, classifiers, label):
    wrong  = 0

    for point in test:
        test_feat = point[:feature_len]
        pred = 0
        for c in classifiers:
            pred += c[1] * sign(numpy.dot(c[0], test_feat))
        if sign(pred) < 0 and point[feature_len] == label:
            wrong += 1
        elif sign(pred) >= 0 and point[feature_len] != label:
            wrong += 1
    return wrong/len(test)
```

In [93]:
```
def average_perceptron_error(test, classifers, label):
    wrong = 0
    w = sum([classi[0] * classi[1] for classi in classifers])
    for point in test:
        sign = numpy.dot(w, point[:feature_len])
        if  sign < 0 and point[feature_len] == label:
            wrong += 1
        elif sign >= 0 and point[feature_len] != label:
            wrong += 1
    return wrong/len(test)
```

In [111]:
```
one_two_subset = [point for point in train if point[len(point)-1] == 1 or point[le
one_two_subset_t = [point for point in test if point[len(point)-1] == 1 or point[l
```

In [117]:
```
for passes in range(2,5):
    w = train_percepie_boy(one_two_subset, passes, 1)
    print(perceptron_error(one_two_subset, w, 1))
    print(perceptron_error(one_two_subset_t, w, 1))
```

```
0.03577981651376147
0.0610079575596817
0.01834862385321101
0.04509283819628647
0.01651376146788991
0.04509283819628647
```

In [119]:
```
for passes in range(2,5):
    classis = train_voted_perceptron(one_two_subset, passes, 1)
    print(voted_perceptron_error(one_two_subset, classis, 1))
    print(voted_perceptron_error(one_two_subset_t, classis, 1))
```

```
0.03853211009174312
0.0610079575596817
0.026605504587155965
0.042440318302387266
0.022018348623853212
0.04509283819628647
```

In [120]:
```
for passes in range(2,5):
    classis = train_voted_perceptron(one_two_subset, passes, 1)
    print(average_perceptron_error(one_two_subset, classis, 1))
    print(average_perceptron_error(one_two_subset_t, classis, 1))
```

```
0.05137614678899083
0.08222811671087533
0.03486238532110092
0.0610079575596817
0.031192660550458717
0.050397877984084884
```

In [104]:
```
from heapq import nlargest, nsmallest

classis = train_voted_perceptron(one_two_subset, 3, 1)
w = sum([classi[0] * classi[1] for classi in classis])

top = nlargest(3, enumerate(w), key=lambda x:x[1])  # returns list of (index, valu
bot = nsmallest(3, enumerate(w), key=lambda x:x[1])  # returns list of (index, val

diction = []

for line in open("pa3dictionary.txt", 'r'):
    diction.append(line.strip().split('/n'))

print("top 3:")
print(diction[top[0][0]])
print(diction[top[1][0]])
print(diction[top[2][0]])
print("bot 3:")
print(diction[bot[0][0]])
print(diction[bot[1][0]])
print(diction[bot[2][0]])
```

```
top 3:
['file']
['program']
['line']
bot 3:
['he']
['team']
['game']
```

In [105]:
```
def train_one_vs_all(train):
    classis = []
    for label in range(1,7):
        classis.append(train_percepie_boy(train, 1, label))
    return classis
```

In [106]:
```
def one_vs_all(test, classifiers):
    preds = []
    for point in test:
        pred = [sign(numpy.dot(classi, point[:feature_len])) for classi in classif
        if sum(pred) == -4:
            preds.append(pred.index(1)+1)
        else:
            preds.append(0)
    return preds
```

In [107]:
```
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import normalize

classis = train_one_vs_all(train)
preds = one_vs_all(test, classis)
actual = [point[feature_len] for point in test]
con_mat = confusion_matrix(actual, preds)
con_mat = con_mat.astype(numpy.float64)
con_mat = normalize(con_mat, norm='l1')
con_mat = con_mat[1:]
con_mat = con_mat.transpose()
```

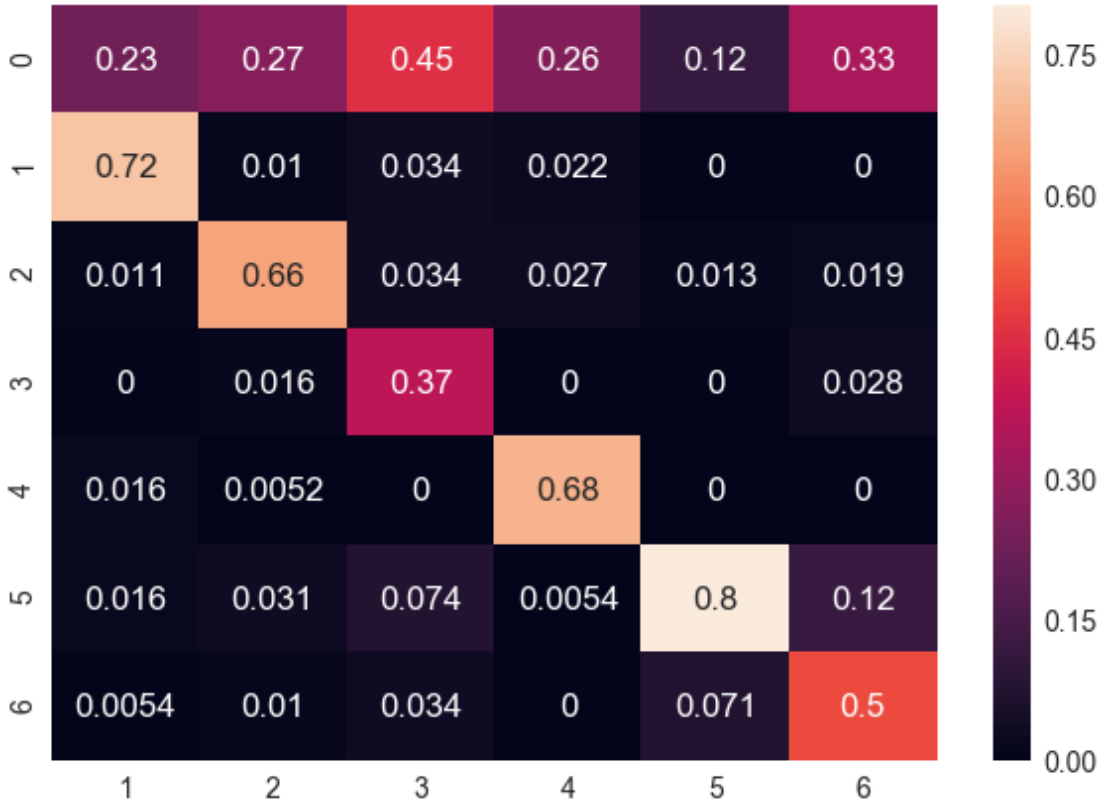
In [108]:
```
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm = pd.DataFrame(con_mat, index = [i for i in "0123456"],
                  columns = [i for i in "123456"])
plt.figure(figsize = (10,7))
sn.heatmap(df_cm, annot=True)
```

Out[108]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x181f3e26160>
```

```
1.
    Perceptron:
    2 Passes
        Train error -
            0.03577981651376147
        Test error -
            0.0610079575596817
    3 Passes
        Train error -
            0.01834862385321101
        Test error -
            0.04509283819628647
    4 Passes
        Train error -
            0.01651376146788991
        Test error -
            0.04509283819628647

    Voted Perceptron:
    2 Passes
        Train error -
            0.03853211009174312
        Test error -
```

```
            0.0610079575596817
    3 Passes
        Train error -
            0.026605504587155965
        Test error -
            0.042440318302387266
    4 Passes
        Train error -
            0.022018348623853212
        Test error -
            0.04509283819628647

    Average Perceptron:
    2 Passes
        Train error -
            0.05137614678899083
        Test error -
            0.08222811671087533
    3 Passes
        Train error -
            0.03486238532110092
        Test error -
            0.0610079575596817
    4 Passes
        Train error -
            0.031192660550458717
        Test error -
            0.050397877984084884

2. The top three words are 'file', 'program' and 'line'
   The bottom three words are 'he', 'team' and 'game'
```

In [121]:
```
plt.show()
```



3. a. Classifier 5
   b. Classifier 3
   c. Classifier 5