In [1]:
```
### Rating baseline: compute averages for each user, or return the globa
l average if we've never seen the user before
import gzip
import random

def readGz(f):
    for l in gzip.open(f):
        yield eval(l)

users = set()
items = set()
set_purchased = []
for l in readGz("train.json.gz"):
    u,i = l['reviewerID'],l['itemID']
    users.add(u)
    items.add(i)
    set_purchased+=[(u,i)]

train_set = set_purchased[:100000]
validation_set1 = set_purchased[100000:200000]

len_u = len(users)
len_i = len(items)

# Now randomly generate unpurchased pairs
set_unpurchased = set()
while(len(set_unpurchased) < 100000):
    u = random.sample(users,1)[0]
    i = random.sample(items,1)[0]
    if (u,i) not in set_purchased:
        set_unpurchased.add((u,i))
```

In [2]:
```
#This part is for prediction
from collections import defaultdict
businessCount = defaultdict(int)
totalPurchases = 0

for (user,business) in train_set:
    businessCount[business] += 1
    totalPurchases += 1

mostPopular = [(businessCount[x], x) for x in businessCount]
mostPopular.sort()
mostPopular.reverse()

return1 = set()
count = 0
for ic, i in mostPopular:
    count += ic
    return1.add(i)
    if count > totalPurchases/2: break

predictions = []
for (u,i) in validation_set1:
    if i in return1:
        predictions+=[1]
    else:
        predictions+=[0]

for (u,i) in set_unpurchased:
    if i in return1:
        predictions+=[1]
    else:
        predictions+=[0]

accuracy = (sum([predictions[i] == 1 for i in range(100000)]) + sum([pre
dictions[i] == 0 for i in range(100000,200000)]))/200000
print(accuracy)
```

```
0.62918
```

In [4]:
```
#Q2:
businessCount = defaultdict(int)
totalPurchases = 0

for (user,business) in train_set:
    businessCount[business] += 1
    totalPurchases += 1

mostPopular = [(businessCount[x], x) for x in businessCount]
mostPopular.sort()
mostPopular.reverse()

return1 = set()
count = 0
for ic, i in mostPopular:
    count += ic
    return1.add(i)
    if count > totalPurchases/1.9: break

predictions = []
for (u,i) in validation_set1:
    if i in return1:
        predictions+=[1]
    else:
        predictions+=[0]

for (u,i) in set_unpurchased:
    if i in return1:
        predictions+=[1]
    else:
        predictions+=[0]

accuracy = (sum([predictions[i] == 1 for i in range(100000)]) + sum([pre
dictions[i] == 0 for i in range(100000,200000)]))/200000
print("accuracy",accuracy)
```

```
accuracy 0.62974
```

When the number is 1.9, the accuracy is higher that in Q1.

In [5]:
```
#Q3:
import gzip
from collections import defaultdict

def readGz(f):
    for l in gzip.open(f):
        yield eval(l)
train = [l for l in readGz("train.json.gz")][:100000]
training_set = []
for l in train:
    u,i,c = l['reviewerID'],l['itemID'],l['categoryID']
    training_set +=[[u,i,c]]

purchase_history = defaultdict(list)
item_cate = defaultdict(int)
for l in training_set:
    u,i,c = l[0],l[1],l[2]
    if c not in purchase_history[u]:
        purchase_history[u] += [c]
    item_cate[i] = c
```

In [6]:
```
#Q4:

# predictions = open("predictions_Purchase.txt", 'w')
# for l in open("pairs_Purchase.txt"):
#     if l.startswith("reviewerID"):
#     #header
#         predictions.write("reviewerID-itemID,prediction\n")
#         continue
#     u,i = l.strip().split('-')
#     if item_cate[i] in purchase_history[u]:
#         predictions.write(u + '-' + i + ",1\n")
#     else:
#         predictions.write(u + '-' + i + ",0\n")

for l in readGz("train.json.gz"):
    user,business = l['reviewerID'],l['itemID']
    businessCount[business] += 1
    totalPurchases += 1

mostPopular = [(businessCount[x], x) for x in businessCount]
mostPopular.sort()
mostPopular.reverse()

return1 = set()
count = 0
for ic, i in mostPopular:
    count += ic
    return1.add(i)
    if count > totalPurchases/1.9: break

predictions = open("predictions_Purchase.txt", 'w')
for l in open("pairs_Purchase.txt"):
    if l.startswith("reviewerID"):
    #header
        predictions.write(l)
        continue
    u,i = l.strip().split('-')
    if i in return1:
        predictions.write(u + '-' + i + ",1\n")
    else:
        predictions.write(u + '-' + i + ",0\n")

predictions.close()
predictions.close()
print("Submitted on Kaggle")
```

```
Submitted on Kaggle
```

username:
xih108

In [13]:
```
#Q5:
import random
d = [l for l in readGz("train.json.gz") if 'categoryID' in l][:40000]
random.shuffle(d)

all_set = []
for l in d:
    u,i,c = l['reviewerID'],l['itemID'],l['categoryID']
    all_set +=[[u,i,c]]

train_set = all_set[:20000]
validation_set = all_set[20000:40000]

sum_cate = [0]*5
user_cate = defaultdict(list)
item_cate = defaultdict(int)
for l in train_set:
    u,i,c = l[0],l[1],int(l[2])
    if u not in user_cate:
        user_cate[u] = [0]*5
    user_cate[u][c] += 1
    sum_cate[c] += 1
    item_cate[i] = c

pred_cate = []
for l in validation_set:
    u,i,c = l[0],l[1],int(l[2])
    if u not in user_cate:
        pred_cate += [0]
    else:
        max_cate = max(user_cate[u])
        tie = [index for index in range(5) if user_cate[u][index] == max
_cate]
        if len(tie) == 1:
            pred_cate += tie
        else:
            max_tie = max([sum_cate[i] for i in tie])
            pred_cate += [index for index in tie if sum_cate[index] == m
ax_tie]

acc = sum([int(validation_set[i][2]) == pred_cate[i] for i in range(len(
validation_set))])
accuracy = acc/len(validation_set)
print("accuracy",accuracy)
```

```
accuracy 0.7394
```

In [14]:
```
#Q6:
import string

d = [l for l in readGz("train.json.gz") if 'categoryID' in l]
train_set = d[:10000]
validation_set = d[10000:20000]

wordCount = defaultdict(int)
count_cat_word = [defaultdict(int),defaultdict(int),defaultdict(int),def
aultdict(int),defaultdict(int)]
wordCount_cate = defaultdict(int)
punctuation = set(string.punctuation)

for l in train_set:
    r = ''.join([c for c in l['reviewText'].lower() if c not in punctuat
ion])
    c = int(l['categoryID'])
    for w in r.split():
        wordCount[w] +=1
        count_cat_word[c][w] += 1

counts = [(wordCount[w],w) for w in wordCount]
counts.sort()
counts.reverse()

words = [p[1] for p in counts[:500]]
# print(words)

freq = defaultdict(float)
sum_app = sum([p[0] for p in counts[:500]])
for w in words:
    freq[w] = wordCount[w]/sum_app
# print(freq)

freq_category = [defaultdict(float),defaultdict(float),defaultdict(float
),defaultdict(float),defaultdict(float)]
for c in range(5):
    for w in words:
        freq_category[c][w] = count_cat_word[c][w]/sum([count_cat_word[c
][w] for w in words])

print("Women",[w for w in words if freq_category[0][w] - freq[w]>0][:10
])
print("Men",[w for w in words if freq_category[1][w] - freq[w]>0][:10])
print("Girls",[w for w in words if freq_category[2][w] - freq[w]>0][:10
])
print("Boys",[w for w in words if freq_category[3][w] - freq[w]>0][:10])
print("Baby",[w for w in words if freq_category[4][w] - freq[w]>0][:10])
```

```
Women ['i', 'a', 'it', 'in', 'this', 'but', 'have', 'not', 'them', 'ver
y']
Men ['the', 'and', 'a', 'to', 'is', 'for', 'of', 'they', 'my', 'are']
Girls ['and', 'a', 'to', 'it', 'is', 'for', 'of', 'this', 'my', 'on']
Boys ['the', 'is', 'for', 'they', 'my', 'are', 'these', 'not', 'them',
'was']
Baby ['and', 'to', 'for', 'of', 'in', 'they', 'my', 'are', 'these', 'o
n']
```

In [10]:
```
#Q7:
from sklearn import svm

d = [l for l in readGz("train.json.gz") if 'categoryID' in l]
def feature(l):
    feature = []
    r = ''.join([c for c in l['reviewText'].lower() if c not in punctuat
ion])
    r = r.split()
    for w in words:
        feature.append(int(w in r))
    return feature

X_train = [feature(l) for l in d[:10000]]
Y_train = [int(l['categoryID']) == 0 for l in d[:10000]]

X_validation = [feature(l) for l in d[10000:20000]]
Y_validation = [int(l['categoryID']) == 0 for l in d[10000:20000]]

for c in [0.01,0.1,1,10,100]:
    clf = svm.LinearSVC(C =c)
    clf.fit(X_train,Y_train)
    train_pred = clf.predict(X_train)
    validation_pred = clf.predict(X_validation)
    accuracy = sum([Y_validation[i] == validation_pred[i] for i in range
(10000)])/10000
    print("C:", c,"accuracy:",accuracy)
```

```
C: 0.01 accuracy: 0.7902
C: 0.1 accuracy: 0.7865
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
C: 1 accuracy: 0.7853
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
C: 10 accuracy: 0.7765
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
C: 100 accuracy: 0.7011
```

When C = 0.01, the binary classifier performs best with an accuracy of 0.7902.

In [11]:
```
#Q8:
from sklearn import svm
import numpy

X_train = [feature(l) for l in d[:10000]]
Y_train = []
X_validation = [feature(l) for l in d[10000:20000]]
Y_validation = [int(l['categoryID']) for l in d[10000:20000]]

for i in range(5):
    Y_train.append([int(l['categoryID'])== i for l in d[:10000]])


def classifiers_5(c):
    clf = []
    Y_pred = []
    scores = []
    for i in range(5):
        clf.append(svm.LinearSVC(C = c))
        clf[i].fit(X_train,Y_train[i])
        scores.append(clf[i].decision_function(X_validation))

    for j in range(0, 10000):
        max_score = max([scores[i][j] for i in range(5)])
        for i in range(5):
            if scores[i][j] == max_score:
                Y_pred.append(i)
                break

#     print(Y_pred)
    accuracy = sum([Y_pred[j] == Y_validation[j] for j in range(10000)])
/10000
    print(c, accuracy)


classifiers_5(0.01)
classifiers_5(0.1)
classifiers_5(1)
classifiers_5(10)
classifiers_5(100)
```

```
0.01 0.7807
0.1 0.7798
```

```
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

```
1 0.7736
```

```
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

```
10 0.7571
```

```
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

```
100 0.6945
```

```
/Users/xinyihe/Library/Python/3.7/lib/python/site-packages/sklearn/svm/
base.py:922: ConvergenceWarning: Liblinear failed to converge, increase
the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

Choose C = 0.01 since it has the highest accuracy.

In [12]:
```
data =[l for l in readGz('test_Category.json.gz')]
X_validation = [feature(l) for l in data]

predictions = open("predictions_Category.txt", 'w')
predictions.write("reviewerID-reviewHash,category\n")

clf = []
Y_pred = []
scores = []
for i in range(5):
    clf.append(svm.LinearSVC(C = 0.01))
    clf[i].fit(X_train,Y_train[i])
    scores.append(clf[i].decision_function(X_validation))

for j in range(len(X_validation)):
    max_score = max([scores[i][j] for i in range(5)])
    for i in range(5):
        if scores[i][j] == max_score:
            Y_pred.append(i)
            break

for i in range(len(data)):
    rid,rh = data[i]['reviewerID'],data[i]['reviewHash']
    predictions.write(rid +"-"+rh+","+str(Y_pred[i])+"\n")


predictions.close()
print("Submitted on Kaggle")
```

```
Submitted on Kaggle
```

username:
xih108

In [ ]:

In [ ]: