

In [1]:

```
import numpy
import urllib
import scipy.optimize
import random
from collections import defaultdict # Dictionaries with default values
import nltk
import string
from nltk.stem.porter import *
from sklearn import linear_model
import ast
import gzip

def findBook(user, userReadBook, bookAllUser):
    l3 = [x for x in list(bookAllUser) if x not in userReadBook[user]]
    proxy = random.choice(l3)
    return proxy
def Jaccard(book1, book2, bookAllUser):
    s1 = bookAllUser[book1]
    s2 = bookAllUser[book2]
    numer = len(s1.intersection(s2))
    denom = len(s1.union(s2))
    return numer / denom

f = gzip.open("train_Interactions.csv.gz", 'rt', encoding="utf8")

header = f.readline()
header = header.strip().split(',')

datatrain = []
datavalid = []
count=0
for line in f:
    fields = line.strip().split(',')
    d = dict(zip(header, fields))
    if count < 190000 :
        datatrain.append(d)
    else:
        datavalid.append(d)
    count=count+1

userReadBook = defaultdict(set)
bookAllUser = defaultdict(set)

for d in datatrain:
    user, book, r = d['userID'], d['bookID'], d['rating']
    userReadBook[user].add(book)
    bookAllUser[book].add(user)

i=0
for d in datavalid:
    if i<10000:
        dd = dict(zip(header, fields))
        dd['userID'] = d['userID']
        dd['bookID'] = findBook(d['userID'], userReadBook, bookAllUser)
        dd['rating'] = 0
        datavalid.append(dd)
        i=i+1
    else:
        break

bookCount = defaultdict(int)
totalRead = 0
for d in datatrain:
    user, book, r = d['userID'], d['bookID'], d['rating']
    bookCount[book] += 1
    totalRead += 1

mostPopular = [(bookCount[x], x) for x in bookCount]
```

```

mostPopular.sort()
mostPopular.reverse()

return1 = set()
count = 0

for ic, i in mostPopular:
    count += ic
    return1.add(i)
    if count > totalRead/1.7: break

prediction = []
thresholdJ = 0.006
#when the book is similar and it is popular, it is true
for d in datavalid:
    user, book, r = d['userID'], d['bookID'], d['rating']
    flag = 0
    for b in userReadBook[user] :
        similarJ = Jaccard(b, book, bookAllUser)
        if similarJ > thresholdJ:
            if book in return1:
                flag = 1
                break
    prediction.append(flag)

count = 0
Tcount = 0
for d in datavalid:
    if prediction[count] > 0 and int(d['rating']) > 0:
        Tcount += 1
    if prediction[count] == 0 and int(d['rating']) == 0:
        Tcount += 1
    count += 1
accuracy = Tcount/len(prediction)
print(accuracy)

```

0.65985

In []: