In [4]:
```
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
```

In [5]:
```
def parseData(fname):
  for l in urllib.urlopen(fname):
    yield ast.literal_eval(l)

def parseDataFromFile(fname):
  for l in open(fname):
    yield ast.literal_eval(l)

data_ = list(parseDataFromFile("goodreads_reviews_comics_graphic.json"))
print(len(data_))
data = data_[:20000]
train = data[:10000]
test = data[10000:]
print(len(train))
print(len(test))
```

```
542338
10000
10000
```

In [6]:
```
wordCount = defaultdict(int)
word2Count = defaultdict(int)
totalWords = 0
totalWords2 = 0
punct = string.punctuation
print(punct)
print(data[0])

for d in data:
    t = d['review_text']
    t = t.lower() # lowercase string
    t = [c for c in t if not (c in punct)] # non-punct characters
    t = ''.join(t) # convert back to string
    words = t.strip().split() # tokenizes
    words2 = [' '.join(x) for x in list(zip(words[:-1],words[1:]))]
    for w in words:
        totalWords += 1
        wordCount[w] += 1

    for w in words2:
        totalWords2 += 1
        word2Count[w] += 1

print(totalWords)
print(totalWords2)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
{'user_id': 'dc3763cdb9b2cae805882878eebb6a32', 'book_id': '18471619', 'review_id':
'66b2ba840f9bd36d6d27f46136fe4772', 'rating': 3, 'review_text': 'Sherlock Holmes and the Vampires of Lon
don \n Release Date: April 2014 \n Publisher: Darkhorse Comics \n Story by: Sylvain Cordurie \n Art by:
Laci \n Colors by: Axel Gonzabo \n Cover by: Jean Sebastien Rossbach \n ISDN: 9781616552664 \n MSRP:
$17.99 Hardcover \n "Sherlock Holmes died fighting Professor Moriarty in the Reichenbach Falls. \n At le
ast, that\'s what the press claims. \n However, Holmes is alive and well and taking advantage of his pre
sumed death to travel the globe. \n Unfortunately, Holmes\'s plans are thwarted when a plague of
vampirism haunts Britain. \n This book collects Sherlock Holmes and the Vampires of London Volumes 1 and
2, originally created by French publisher Soleil." - Darkhorse Comics \n When I received this copy of "S
herlock Holmes and the Vampires of London" I was Ecstatic! The cover art was awesome and it was about tw
o of my favorite things, Sherlock Holmes and Vampires. I couldn\'t wait to dive into this! \n
Unfortunately, that is where my excitement ended. The story takes place a month after Sherlock Holmes sup
posed death in his battle with Professor Moriarty. Sherlock\'s plan to stay hidden and out of site are r
uined when on a trip with his brother Mycroft, they stumble on the presence of vampires. That is about a
s much of Sherlock\'s character that comes through the book. I can\'t even tell you the story really
because nothing and I mean nothing stuck with me after reading it. I never, ever got the sense of
Sherlock Holmes anywhere in this graphic novel, nor any real sense of mystery or crime. It was just
Sherlock somehow battling vampires that should have had absolutely no trouble snuffing him out in a figh
t, but somehow always surviving and holding his own against supernatural, super powerful, blazingly fast
creatures. \n The cover art is awesome and it truly made me excited to read this but everything else
feel completely flat for me. I tried telling myself that "it\'s a graphic novel, it would be hard to
translate mystery, details, emotion" but then I remembered reading DC Comic\'s "Identity Crisis" and rea
lized that was a load of crap. I know it\'s unfair to compare the two as "Identity Crisis" had popular m
ystery author Brad Meltzer writing it right? Yeah....no. The standard was set that day and there is more
than enough talent out there to create a great story in a graphic novel. \n That being said, it wasn\'t
a horrible story, it just didn\'t grip me for feel anything like Sherlock Holmes to me. It was easy enou
gh to follow but I felt no sense of tension, stakes or compassion for any of the characters. \n As far a
s the vampires go, it\'s hard to know what to expect anymore as there are so many different versions
these days. This was the more classic version which I personally prefer, but again I didn\'t find
anything that portrayed their dominance, calm confidence or sexuality. There was definitely a presence of
their physical prowess but somehow that was lost on me as easily as Sherlock was able to defend himself.
I know it, wouldn\'t do to kill of the main character, but this would have a been a great opportunity to
build around the experience and beguiling nature of a vampire that had lived so many years of
experience. Another chance to showcase Sherlock\'s intellect in a battle of wits over strength in
something more suitable for this sort of story as apposed to trying to make it feel like an action
movie. \n Maybe I expected to much and hoped to have at least a gripping premise or some sort of
interesting plot or mystery but I didn\'t find it here. This may be a must have for serious Sherlock
Holmes fans that have to collect everything about him, but if you are looking for a great story inside a
graphic novel, I would have to say pass on this one. \n That artwork is good, cover is great, story is la
cking so I am giving it 2.5 out of 5 stars.', 'date_added': 'Thu Dec 05 10:44:25 -0800 2013',
'date_updated': 'Thu Dec 05 10:45:15 -0800 2013', 'read_at': 'Tue Nov 05 00:00:00 -0800 2013',
'started_at': '', 'n_votes': 0, 'n_comments': 0}
1814797
1794812
```

In [7]:

```
print(len(wordCount))
counts = [(wordCount[w], w) for w in wordCount]
counts.sort()
counts.reverse()
print(counts[:10])
print(counts[5000:5010])

print(len(word2Count))
counts2 = [(word2Count[w], w) for w in word2Count]
counts2.sort()
counts2.reverse()
print(counts2[:10])
print(counts2[5000:5010])

countsadd = counts
countsadd.extend(counts2)
countsadd.sort()
countsadd.reverse()
print(countsadd[:10])
print(countsadd[5000:5010])

unigrams = [w[1] for w in counts[:1000]]
print(unigrams[:10])
unigramId = dict(zip(unigrams, range(len(unigrams))))
unigramSet = set(unigrams)

bigrams = [w[1] for w in counts2[:1000]]
print(bigrams[:10])
bigramId = dict(zip(bigrams, range(len(bigrams))))
bigramSet = set(bigrams)
```

```
combined = [w[1] for w in countsadd[:1000]]
print(combined[:50])
combinedId = dict(zip(combined, range(len(combined))))
combinedSet = set(combined)
```

```
59814
[(100875, 'the'), (52202, 'and'), (49717, 'a'), (46551, 'of'), (41950, 'to'), (34692, 'i'), (30131, 'is')
, (24381, 'this'), (24158, 'it'), (23899, 'in')]
[(22, 'prominent'), (22, 'primary'), (22, 'powered'), (22, 'poco'), (22, 'plein'), (22, 'photo'), (22, '
personagens'), (22, 'pause'), (22, 'patient'), (22, 'outfit')]
528310
[(12580, 'of the'), (6821, 'in the'), (4887, 'the story'), (4625, 'and the'), (4472, 'is a'), (4055, 'to
the'), (3379, 'this is'), (3157, 'to be'), (2785, 'it was'), (2781, 'with the')]
[(37, 'well told'), (37, 'was disappointed'), (37, 'war ii'), (37, 'wanted more'), (37, 'violent and'),
(37, 'used in'), (37, 'track down'), (37, 'to volume'), (37, 'to spoil'), (37, 'to rate')]
[(100875, 'the'), (52202, 'and'), (49717, 'a'), (46551, 'of'), (41950, 'to'), (34692, 'i'), (30131, 'is')
, (24381, 'this'), (24158, 'it'), (23899, 'in')]
[(65, 'discovered'), (65, 'develop'), (65, 'destruction'), (65, 'combined'), (65, 'cliched'), (65,
'characterization of'), (65, 'cases'), (65, 'calls'), (65, 'but his'), (65, 'but all')]
['the', 'and', 'a', 'of', 'to', 'i', 'is', 'this', 'it', 'in']
['of the', 'in the', 'the story', 'and the', 'is a', 'to the', 'this is', 'to be', 'it was', 'with the']
['the', 'and', 'a', 'of', 'to', 'i', 'is', 'this', 'it', 'in', 'that', 'but', 'was', 'with', 'as', 'story
', 'for', 'of the', 'its', 'on', 'are', 'not', 'have', 'you', 'so', 'be', 'one', 'book', 'read', 'in
the', 'more', 'like', 'all', 'at', 'an', 'his', 'just', 'really', 'about', 'from', 'me', 'some', 'my', 'h
e', 'up', 'what', 'her', 'good', 'the story', 'by']
```

Train a regressor

In [8]:
```
def feature_unigram(datum):
    feat = [0]*len(unigrams)
    r = ''.join([c for c in datum['review_text'].lower() if not c in punct])
    ws = r.split()

    for w in ws :
        if w in unigrams:
            feat[unigramId[w]] += 1
    feat.append(1) #offset
    return feat
```

In [9]:
```
def feature_bigram(datum):
    feat = [0]*len(bigrams)
    r = ''.join([c for c in datum['review_text'].lower() if not c in punct])
    ws = r.split()
    ws2 = [' '.join(x) for x in list(zip(ws[:-1],ws[1:]))]

    for w in ws2 :
        if w in bigrams:
            feat[bigramId[w]] += 1
    feat.append(1) #offset
    return feat
```

In [10]:
```
def feature_combined(datum):
    feat = [0]*len(combined)
    r = ''.join([c for c in datum['review_text'].lower() if not c in punct])
    ws = r.split()
    ws2 = [' '.join(x) for x in list(zip(ws[:-1],ws[1:]))]

    for w in ws + ws2 :
        if w in combined:
            feat[combinedId[w]] += 1
    feat.append(1) #offset
    return feat
```

In [11]:
```
# unigram
X_train = [feature_unigram(d) for d in train]
y_train = [d['rating'] for d in train]
print(X_train[0][:20])
X_test = [feature_unigram(d) for d in test]
y_test = [d['rating'] for d in test]


clf = linear_model.Ridge(1.0, fit_intercept=False) # MSE + 1.0 l2
```

```
    clf.fit(X_train, y_train)
    theta = clf.coef_
    pred_train = clf.predict(X_train)
    MSEtrain = sum((y_train - pred_train)**2)/len(y_train)
    pred_test = clf.predict(X_test)
    MSEtest = sum((y_test - pred_test)**2)/len(y_test)
    print(theta[:20])
    print("MSEtrain = ",MSEtrain)
    print("MSEtest = ",MSEtest)

    weights = list(zip(theta, words + ['constant_feat']))
    weights.sort()
    weights.reverse()
    print(weights[:5])
    print(weights[-5:])
```

```
[20, 16, 18, 23, 21, 17, 8, 10, 13, 7, 13, 9, 11, 3, 8, 8, 6, 0, 3, 4]
[-0.00457135  0.02714538 -0.00551991 -0.00079496  0.01669224 -0.01136056
 -0.00395026 -0.00347088 -0.009181    0.01306438  0.01329566 -0.03784267
 -0.0252959  -0.0184116   0.02904519 -0.04263389 -0.01868626  0.
 -0.00255318  0.03551502]
MSEtrain =  0.8366988157914573
MSEtest =  1.2175286628938813
[(0.16980834842572978, 'graphic'), (0.05298929605128229, 'and'), (0.046125233949278255, 'did'),
(0.040575182490206335, 'fear'), (0.03551501622887232, 'with')]
[(-0.03784267266626414, 'like'), (-0.039731448165314126, 'an'), (-0.042633890048506984, 'kill'), (-
0.05083494692111321, 'and'), (-0.12129747395215885, 'many')]
```

In [12]:

```
    # bigram
    X_train = [feature_bigram(d) for d in data]
    y_train = [d['rating'] for d in data]
    print(X_train[0][:20])
    X_test = [feature_bigram(d) for d in test]
    y_test = [d['rating'] for d in test]


    clf = linear_model.Ridge(1.0, fit_intercept=False) # MSE + 1.0 l2
    clf.fit(X_train, y_train)
    theta = clf.coef_
    pred_train = clf.predict(X_train)
    MSEtrain = sum((y_train - pred_train)**2)/len(y_train)
    pred_test = clf.predict(X_test)
    MSEtest = sum((y_test - pred_test)**2)/len(y_test)
    print(theta[:20])
    print("MSEtrain = ",MSEtrain)
    print("MSEtest = ",MSEtest)

    weights = list(zip(theta, words + ['constant_feat']))
    weights.sort()
    weights.reverse()
    print(weights[:5])
    print(weights[-5:])
```

```
[2, 1, 2, 3, 0, 0, 0, 0, 3, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1]
[ 0.01311357 -0.0175345  -0.06386067  0.01209601  0.02495477  0.01678908
  0.04453342 -0.03019126 -0.01181329 -0.02630695  0.0032872   0.00020249
  0.04096715  0.0414431   0.06604158  0.04040725  0.01215056 -0.02004027
  0.01831881  0.02489957]
MSEtrain =  1.0065412723734712
MSEtest =  1.1220572272163036
[(0.14191392729040753, 'reassuring'), (0.0851562815472533, 'it'), (0.06604158286633115, 'i'),
(0.05882911866021295, 'but'), (0.05640433763721195, 'graphic')]
[(-0.06961567690231196, 'kind'), (-0.08464565450241211, 'novel'), (-0.10207313941865265, 'so'), (-
0.11931400125694432, 'inadequacy'), (-0.1336222036339421, 'hope')]
```

In [13]:

```
    # combined
    X_train = [feature_combined(d) for d in data]
    y_train = [d['rating'] for d in data]
    print(X_train[0][:20])
    X_test = [feature_combined(d) for d in test]
    y_test = [d['rating'] for d in test]



    clf = linear_model.Ridge(1.0, fit_intercept=False) # MSE + 1.0 l2
    clf.fit(X_train, y_train)
    theta = clf.coef_
```

```
pred_train = clf.predict(X_train)
MSEtrain = sum((y_train - pred_train)**2)/len(y_train)
pred_test = clf.predict(X_test)
MSEtest = sum((y_test - pred_test)**2)/len(y_test)
print(theta[:20])
print("MSEtrain = ",MSEtrain)
print("MSEtest = ",MSEtest)

weights = list(zip(theta, words + ['constant_feat']))
weights.sort()
weights.reverse()
print(weights[:5])
print(weights[-5:])
```

```
[20, 16, 18, 23, 21, 17, 8, 10, 13, 7, 13, 9, 11, 3, 8, 8, 6, 2, 3, 4]
[ 0.00250727  0.01543956  0.00369686  0.00560763  0.0112723   0.00771787
 -0.00843389 -0.0059452   0.0012322   0.00744561  0.00133197 -0.04881002
 -0.04028023 -0.00953129  0.01934328 -0.01307313 -0.03214375 -0.00487283
 -0.00175648  0.00416778]
MSEtrain =  0.9494877022028863
MSEtest =  1.0534730062253814
[(0.17720752773970394, 'graphic'), (0.06759523019416336, 'and'), (0.042655790155269345, 'did'),
(0.039480098883190866, 'great'), (0.03297489843505927, 'teens')]
[(-0.041554104888092115, 'an'), (-0.04375117959870784, 'and'), (-0.04881001987645827, 'like'), (-
0.0539720160958572, 'inspiration'), (-0.12737492382679572, 'many')]
```

Ans for Problem1 :

a. unigram feature

MSEtrain = 0.8366988157914573

MSEtest = 1.2175286628938813

five most positive tokens : [(0.16980834842572978, 'graphic'), (0.05298929605128229, 'and'), (0.046125233949278255, 'did'), (0.040575182490206335, 'fear'), (0.03551501622887232, 'with')]

five most negative tokens : [(-0.03784267266626414, 'like'), (-0.039731448165314126, 'an'), (-0.042633890048506984, 'kill'), (-0.05083494692111321, 'and'), (-0.12129747395215885, 'many')]

b. bigram feature

MSEtrain = 1.0065412723734712

MSEtest = 1.1220572272163036

five most positive tokens : [(0.14191392729040753, 'reassuring'), (0.0851562815472533, 'it'), (0.06604158286633115, 'i'), (0.05882911866021295, 'but'), (0.05640433763721195, 'graphic')]

five most negative tokens : [(-0.06961567690231196, 'kind'), (-0.08464565450241211, 'novel'), (-0.10207313941865265, 'so'), (-0.11931400125694432, 'inadequacy'), (-0.1336222036339421, 'hope')]

c. combined feature

MSEtrain = 0.9494877022028863

MSEtest = 1.0534730062253814

five most positive tokens : [(0.17720752773970394, 'graphic'), (0.06759523019416336, 'and'), (0.042655790155269345, 'did'), (0.039480098883190866, 'great'), (0.03297489843505927, 'teens')]

five most negative tokens : [(-0.041554104888092115, 'an'), (-0.04375117959870784, 'and'), (-0.04881001987645827, 'like'), (-0.0539720160958572, 'inspiration'), (-0.12737492382679572, 'many')]

In [14]:

```
import math

unigrams = [w[1] for w in counts[:1000]]
# tf-idf
df = defaultdict(int)
for d in train:
    r = ''.join([c for c in d['review_text'].lower() if not c in punct])
    for w in set(r.split()):
        if w in unigrams:
            df[w] += 1

rev = train[0]
tf = defaultdict(int)
```

```
r = ''.join([c for c in rev['review_text'].lower() if not c in punct])
for w in r.split():
    if w in unigrams:
        tf[w] += 1

tfidfQuery = [0]*1000
for i, w in enumerate(unigrams):
    if df[w]!=0:
        tfidfQuery[i] = tf[w]*math.log2(len(train)/df[w])
```

In [17]:

```
# cosine similarity
def Cosine(x1,x2):
    numer = 0
    norm1 = 0
    norm2 = 0
    for a1,a2 in zip(x1,x2):
        numer += a1*a2
        norm1 += a1**2
        norm2 += a2**2
    if norm1*norm2:
        return numer / math.sqrt(norm1*norm2)
    return 0

similarities = []
for rev2 in train:
    tf = defaultdict(int)#date_updated
    r = ''.join([c for c in rev2['review_text'].lower() if not c in punct])
    for w in r.split():
        if w in unigrams:
            tf[w] += 1
    tfidf2 = [0]*1000
    for i, w in enumerate(unigrams):
        if df[w]!=0:
            tfidf2[i] = tf[w]*math.log2(len(train)/df[w])
    similarities.append((Cosine(tfidfQuery, tfidf2), rev2['review_id'] ,rev2['review_text']))

similarities.sort(reverse=True)
print([ similarities[i][0] for i in range(10) ])
print(similarities[1])
```

[1.0, 0.5138592638638093, 0.5078040760482171, 0.5023689098425748, 0.48664311392282844, 0.47870877498155306, 0.47870877498155306, 0.47870877498155306, 0.47870877498155306, 0.47870877498155306]
(0.5138592638638093, '2908fbc62d0e573c9032424792c30663', 'Originally posted on My Book Musings. \n *Copy provided by launch event organizer for an honest review. \n Anak Bathala: Kalem is the first book in a five-book graphic novel series, revolving around mythology of the old times. The first book is about Kalem\'s search for why he is called "Anak Bathala" (demi-god). The language in the book has a mixture o f English and Filipino terms, with Baybayin and Surat Mangyan scripts written in some parts of the book. \n A brief overview of the whole series from their Facebook page: \n Anak Bathala is the epic adventure of Kalem that showcases rich Filipino mythology and culture. It exhibits native Filipino beliefs and fol k work intertwined with values. The other books included in the series collection are: Yamal, Arau, Yesha and Anak Bathala. Each character embodies intrinsic Filipino traits like admirable virtues of cour age, determination, self-discipline, prudence, camaraderie and leadership. Their stories also uncover the deceiving sphere of glorious power. The elven persona of Yesha exudes the remarkable attributes of womanhood that represents Filipino women. Anak Bathala pays homage to the golden age of comic makers wit h its ground of artistic and detailed illustrations. \n The final book of Anak Bathala juxtaposes the re volving characters of books 1-4 and how they will join together to overcome the evil transgression of Haring Nannum and Bathala Karimlan in the Land of Mystical Mindoro. \n -oOo- \n The story is fast-paced and interesting, but the story was too short for me to truly grasp Kalem\'s character. But from what I h ave read so far, he seems nice, responsible, and brave. I thought he was going to have a love interest with his childhood friend, but reading on, I was thinking that it would be slightly hard for him to have a human relationship, especially if he has to go off into a battle. \n I\'m the type of reader who likes learning new things, whether they be fact or fiction. At first, I did not like the scripts used because I kept having to translate them myself. I was on page nine before I discovered that there are Filipino a nd English translations on pages 122 to 123. Gah. Silly me! I must say, though, that the scripts are easy to understand after a couple of pages of going back and forth. If you\'re fluent in Filipino, I suggest you enjoy the scripts and translate them yourself. I think it\'s part of the whole enjoying-the-graphic-novel process. \n My primary beef with the graphic novel is that it is too dark to see the illustrations. I\'m sure the illustrations are very nice, but too often, I had to strain my eyes to see the outlines. It made it hard for me to read and appreciate the graphic novel as a whole. The cover is nice, though. While the background is very dark, Kalem stands out, which I think is the whole point as he is the focus of book 1. \n One thing that I did notice though is that Kalem seemed to grow older as the story progressed. I actually liked it, because I felt like the strain of the battle, the pain of losing Ba\' was showing on his face. I felt the weight of their expectations upon him, and it was nice that the se were reflected on his face. I don\'t know if that was the intent of the creators, but it sure was a n ice touch. \n One part that confused me is the presence of the woman at the start of the novel. Though b y the end of the book I pretty much knew, but it still would have been nice to know her, because I was c onfused as to why there is a woman there, amidst death and war. Also, by the end of the book, I wondered what happened to the mutia that Ba\' Magiting got. Remember that? I don\'t think it was mentioned in the story that Kalem has it, or what it is for. \n Other things I noticed is that there are some errors such as in the line "The panganay left Kalualhatian". I think it lacks the word "for" in between left and Kalualhatian, because the idea is that he joined Bathala in Kalualhatian, leaving this world behind. Anot her is that there was a misplaced apostrophe in "Detinos\' reached further..." \n I\'m hoping the next books would have more dialogue and explanations, as well as clearer images. There\'s an excerpt of Kalem in their website and the graphics are gorgeous! I hope they\'ll publish that edition. I\'m not sure if i t was just with my copy, but mine already had a few loose pages. I\'m hoping that\'s just in my case! \n Am I going to get the next book? Probably! Because my attention was already grabbed by Anak Bathala: Kalem and I started to get curious as to what happens, and how he came to be on earth. Being a demi god, isn\'t he supposed to be uh...somewhere else? So I\'m really curious as to how his journey as and what his role is in the big picture. \n If you\'re a fan of mythology, Filipino fiction, looking for something new, a little bit of history mixed with fantasy, do be sure to check this out. Anak Bathala: K alem is available at Fully Booked branches nationwide.')

Ans for Problem2 :

a. review ID that has the highest cosine similarity compared to the first review : '2908fbc62d0e573c9032424792c30663'

```
text : 'Originally posted on My Book Musings. \n *Copy provided by launch event organizer for
an honest review. \n Anak Bathala: Kalem is the first book in a five-book graphic novel series,
revolving around mythology of the old times. The first book is about Kalem\'s search for why he
is called "Anak Bathala" (demi-god). The language in the book has a mixture of English and
Filipino terms, with Baybayin and Surat Mangyan scripts written in some parts of the book. \n A
brief overview of the whole series from their Facebook page: \n Anak Bathala is the epic
adventure of Kalem that showcases rich Filipino mythology and culture. It exhibits native
Filipino beliefs and folk work intertwined with values. The other books included in the series
collection are: Yamal, Arau, Yesha and Anak Bathala. Each character embodies intrinsic Filipino
traits like admirable virtues of courage, determination, self-discipline, prudence, camaraderie
and leadership. Their stories also uncover the deceiving sphere of glorious power. The elven
persona of Yesha exudes the remarkable attributes of womanhood that represents Filipino women.
Anak Bathala pays homage to the golden age of comic makers with its ground of artistic and
detailed illustrations. \n The final book of Anak Bathala juxtaposes the revolving characters
of books 1-4 and how they will join together to overcome the evil transgression of Haring
Nannum and Bathala Karimlan in the Land of Mystical Mindoro. \n -oOo- \n The story is fast-
paced and interesting, but the story was too short for me to truly grasp Kalem\'s character.
But from what I have read so far, he seems nice, responsible, and brave. I thought he was going
to have a love interest with his childhood friend, but reading on, I was thinking that it would
be slightly hard for him to have a human relationship, especially if he has to go off into a
battle. \n I\'m the type of reader who likes learning new things, whether they be fact or
fiction. At first, I did not like the scripts used because I kept having to translate them
myself. I was on page nine before I discovered that there are Filipino and English translations
on pages 122 to 123. Gah. Silly me! I must say, though, that the scripts are easy to understand
after a couple of pages of going back and forth. If you\'re fluent in Filipino, I suggest you
enjoy the scripts and translate them yourself. I think it\'s part of the whole enjoying-the-
graphic-novel process. \n My primary beef with the graphic novel is that it is too dark to see
the illustrations. I\'m sure the illustrations are very nice, but too often, I had to strain my
eyes to see the outlines. It made it hard for me to read and appreciate the graphic novel as a
whole. The cover is nice, though. While the background is very dark, Kalem stands out, which I
think is the whole point as he is the focus of book 1. \n One thing that I did notice though is
that Kalem seemed to grow older as the story progressed. I actually liked it, because I felt
like the strain of the battle, the pain of losing Ba\' was showing on his face. I felt the
weight of their expectations upon him, and it was nice that these were reflected on his face. I
don\'t know if that was the intent of the creators, but it sure was a nice touch. \n One part
that confused me is the presence of the woman at the start of the novel. Though by the end of
the book I pretty much knew, but it still would have been nice to know her, because I was
confused as to why there is a woman there, amidst death and war. Also, by the end of the book,
I wondered what happened to the mutia that Ba\' Magiting got. Remember that? I don\'t think it
was mentioned in the story that Kalem has it, or what it is for. \n Other things I noticed is
that there are some errors such as in the line "The panganay left Kalualhatian". I think it
lacks the word "for" in between left and Kalualhatian, because the idea is that he joined
Bathala in Kalualhatian, leaving this world behind. Another is that there was a misplaced
apostrophe in "Detinos\' reached further..." \n I\'m hoping the next books would have more
dialogue and explanations, as well as clearer images. There\'s an excerpt of Kalem in their
website and the graphics are gorgeous! I hope they\'ll publish that edition. I\'m not sure if
it was just with my copy, but mine already had a few loose pages. I\'m hoping that\'s just in
my case! \n Am I going to get the next book? Probably! Because my attention was already grabbed
by Anak Bathala: Kalem and I started to get curious as to what happens, and how he came to be
on earth. Being a demi god, isn\'t he supposed to be uh...somewhere else? So I\'m really
curious as to how his journey as and what his role is in the big picture. \n If you\'re a fan
of mythology, Filipino fiction, looking for something new, a little bit of history mixed with
fantasy, do be sure to check this out. Anak Bathala: Kalem is available at Fully Booked
branches nationwide.'

cosine similarity : 0.5138592638638093
```

(a) Latent Factor Model with user and item

In [33]:

```
import gzip
import matplotlib.pyplot as plt
import numpy
import random
import scipy
import tensorflow as tf
from collections import defaultdict
from fastFM import als
from scipy.spatial import distance
```

In [28]:
```
from dateutil.parser import parse

## user only appear successively
## date added is from new to old for each user
cur_user = data[0]['user_id']
cur_date = data[0]['date_added']
userIDs = {}
cnt = 0
for d in data:
    #print(cnt)
    u = d['user_id']
    du = d['date_added']
    #if (u=='1e946b8f76d5a75414946767cd18cff9'): print(cnt)
    if (u!=cur_user and u in userIDs): print("%d!!! %s %s"%(cnt, cur_user,u))
    if (u==cur_user and parse(d['date_added'])>parse(cur_date)): print("%d!!! %s %s"%(cnt, parse(d['date_
    cur_user = u
    cur_date = du
    if not u in userIDs: userIDs[u] = len(userIDs)
    cnt+=1
'''
```

Out[28]:
```
'\nfrom dateutil.parser import parse\n\n## user only appear successively\n## date added is from new to o
ld for each user\ncur_user = data[0][\'user_id\']\ncur_date = data[0][\'date_added\']\nuserIDs = {}\ncnt
= 0\nfor d in data:\n    #print(cnt)\n    u = d[\'user_id\']\n    du = d[\'date_added\']\n    #if (u==\'1
e946b8f76d5a75414946767cd18cff9\'): print(cnt)\n    if (u!=cur_user and u in userIDs): print("%d!!! %s %s
"%(cnt, cur_user,u))\n    if (u==cur_user and parse(d[\'date_added\'])>parse(cur_date)): print("%d!!! %s
%s"%(cnt, parse(d[\'date_added\']),parse(cur_date)))\n    cur_user = u\n    cur_date = du\n    if not u
in userIDs: userIDs[u] = len(userIDs)\n    cnt+=1\n'
```

In [80]:
```
import dateutil.parser
userCount = {}
for d in data:
    u = d['user_id']
    if not u in userCount: userCount[u] = 1
    else : userCount[u] +=1

DataPerUser = defaultdict(list)
userIDs,itemIDs = {},{}
for d in data:
    u = d['user_id']
    i = d['book_id']
    t = d['date_added']
    r = d['rating']
    dt = dateutil.parser.parse(t)
    t = int(dt.timestamp())
    #print(t)
    if userCount[u]<3 : continue
    if not u in userIDs: userIDs[u] = len(userIDs)
    if not i in itemIDs: itemIDs[i] = len(itemIDs)
    DataPerUser[u].append((t,u,i,r))
nUsers,nItems = len(userIDs),len(itemIDs)
print("nUsers = %d, nItems = %d"%(nUsers,nItems))

train = []
test = []
for u in DataPerUser:
    DataPerUser[u].sort()
    length = len(DataPerUser[u])
    for i in range(length-1):
        train.append(DataPerUser[u][i])
    test.append(DataPerUser[u][length-1])

print("len(train) = %d, len(test) = %d"%(len(train),len(test)))
```

```
nUsers = 636, nItems = 11892
len(train) = 18291, len(test) = 636
```

In [82]:
```
import fastFM
X_train = scipy.sparse.lil_matrix((len(train), nUsers + nItems))
for i in range(len(train)): # (t,u,i,r)
    user = userIDs[train[i][1]]
    item = itemIDs[train[i][2]]
    X_train[i,user] = 1 # One-hot encoding of user
    X_train[i,nUsers + item] = 1 # One-hot encoding of item
```

```
y_train = numpy.array([d[3] for d in train])

X_test = scipy.sparse.lil_matrix((len(test), nUsers + nItems))
for i in range(len(test)):
    user = userIDs[test[i][1]]
    item = itemIDs[test[i][2]]
    X_test[i,user] = 1 # One-hot encoding of user
    X_test[i,nUsers + item] = 1 # One-hot encoding of item
y_test = numpy.array([d[3] for d in test])

def MSE(predictions, labels):
    differences = [(x-y)**2 for x,y in zip(predictions,labels)]
    return sum(differences) / len(differences)

fm = fastFM.als.FMRegression(n_iter=1000, init_stdev=0.1, rank=5, l2_reg_w=0.1, l2_reg_V=0.5)
fm.fit(X_train, y_train)
y_pred = fm.predict(X_train)
print("MSE train = ",MSE(y_pred, y_train))

y_pred = fm.predict(X_test)
y_pred[:10]
y_test[:10]
print("MSE test = ",MSE(y_pred, y_test))
```

```
MSE train =  0.013160128282292967
MSE test =  1.3137676309457482
```

(b) MC model which includes item and previous item

In [87]:

```
import dateutil
# itemsPerUser[u], userIDs, itemIDs, interactionsWithPrevious, items, optimizer
userIDs = {}
itemIDs = {}
interactions = []
interactionsPerUser = defaultdict(list)

for d in data:
    u = d['user_id']
    i = d['book_id']
    t = d['date_added']
    r = d['rating']
    dt = dateutil.parser.parse(t)
    t = int(dt.timestamp())
    #print(t)
    if userCount[u]<3 : continue
    if not u in userIDs: userIDs[u] = len(userIDs)
    if not i in itemIDs: itemIDs[i] = len(itemIDs)
    interactions.append((t,u,i,r))
    interactionsPerUser[u].append((t,i,r))
#interactions[0]
interactions.sort()
print(len(interactions))

itemIDs['dummy'] = len(itemIDs)
X_train = []
X_test = []
y_train = []
y_test = []
for u in interactionsPerUser:
    interactionsPerUser[u].sort()
    lastItem = 'dummy'
    length = len(interactionsPerUser[u])
    cnt = 0
    for (t,i,r) in interactionsPerUser[u]:
        cnt+=1
        if cnt < length:
            X_train.append((u,i,lastItem,r))
            y_train.append(r)
            lastItem = i
        else :
            X_test.append((u,i,lastItem))
            y_test.append(r)
print(len(X_train),len(X_test))
print(len(y_train),len(y_test))
```

```
itemsPerUser = defaultdict(set)
for u,i,_,_ in X_train: #(u,i,lastItem,r)
    itemsPerUser[u].add(i)
items = list(itemIDs.keys())
optimizer = tf.keras.optimizers.Adam(0.1)
```

```
18927
18291 636
18291 636
```

In [88]:

```
class MC(tf.keras.Model):
    def __init__(self, K, lamb, IJ=1):
        super(MC, self).__init__()
        # Initialize variables
        self.betaI = tf.Variable(tf.random.normal([len(itemIDs)],stddev=0.001))
        self.gammaIJ = tf.Variable(tf.random.normal([len(itemIDs),K],stddev=0.001))
        self.gammaJI = tf.Variable(tf.random.normal([len(itemIDs),K],stddev=0.001))
        # Regularization coefficient
        self.lamb = lamb
        self.IJ = IJ

    # Prediction for a single instance
    def predict(self, i, j):
        p = self.betaI[i] +  self.IJ * tf.tensordot(self.gammaIJ[i], self.gammaJI[j], 1)
        return p

    # Regularizer
    def reg(self):
        return self.lamb * (tf.nn.l2_loss(self.betaI) +\
                            tf.nn.l2_loss(self.gammaIJ) +\
                            tf.nn.l2_loss(self.gammaJI))

    def call(self, sampleI, # item
                   sampleJ, # previous item
                   sampleK, sampleR): # negative item
        i = tf.convert_to_tensor(sampleI, dtype=tf.int32)
        j = tf.convert_to_tensor(sampleJ, dtype=tf.int32)
        k = tf.convert_to_tensor(sampleK, dtype=tf.int32)
        gamma_ij = tf.nn.embedding_lookup(self.gammaIJ, i)
        gamma_ji = tf.nn.embedding_lookup(self.gammaJI, j)
        beta_i = tf.nn.embedding_lookup(self.betaI, i)
        x_ij = beta_i + self.IJ * tf.reduce_sum(tf.multiply(gamma_ij, gamma_ji), 1)
        gamma_kj = tf.nn.embedding_lookup(self.gammaIJ, k)
        gamma_jk = tf.nn.embedding_lookup(self.gammaJI, j)
        beta_k = tf.nn.embedding_lookup(self.betaI, k)
        x_kj = beta_k + self.IJ * tf.reduce_sum(tf.multiply(gamma_kj, gamma_jk), 1)

        mse = tf.keras.losses.MeanSquaredError()
        return -tf.reduce_mean(tf.math.log(tf.math.sigmoid(x_ij - x_kj)))+ mse(x_ij,sampleR)

modelMC = MC(5, 0.00001)
```

In [89]:

```
def trainingStep(model, interactions):
    with tf.GradientTape() as tape:
        sampleI, sampleJ, sampleK, sampleR = [], [], [], []
        for _ in range(100000):
            u,i,j,r = random.choice(interactions) # positive sample
            k = random.choice(items) # negative sample
            while k in itemsPerUser[u]:
                k = random.choice(items)
            sampleI.append(itemIDs[i])
            sampleJ.append(itemIDs[j])
            sampleK.append(itemIDs[k])
            sampleR.append(float(r))

        loss = model(sampleI,sampleJ,sampleK,sampleR)
        loss += model.reg()
    gradients = tape.gradient(loss, model.trainable_variables)
    optimizer.apply_gradients((grad, var) for
                              (grad, var) in zip(gradients, model.trainable_variables)
                              if grad is not None)
    return loss.numpy()
```

In [92]:

```
for i in range(600):
    obj = trainingStep(modelMC, X_train)
```

```
        if (i % 10 == 9): print("iteration " + str(i+1) + ", objective = " + str(obj))
```

```
iteration 10, objective = 4.13834
iteration 20, objective = 1.4678717
iteration 30, objective = 1.1231085
iteration 40, objective = 0.7891793
iteration 50, objective = 0.688743
iteration 60, objective = 0.6427677
iteration 70, objective = 0.6207298
iteration 80, objective = 0.604971
iteration 90, objective = 0.5938127
iteration 100, objective = 0.5860394
iteration 110, objective = 0.57835126
iteration 120, objective = 0.5731423
iteration 130, objective = 0.56896055
iteration 140, objective = 0.5655522
iteration 150, objective = 0.56131166
iteration 160, objective = 0.55540156
iteration 170, objective = 0.5564665
iteration 180, objective = 0.5507009
iteration 190, objective = 0.5519376
iteration 200, objective = 0.550581
iteration 210, objective = 0.55066055
iteration 220, objective = 0.54869735
iteration 230, objective = 0.54819083
iteration 240, objective = 0.54728955
iteration 250, objective = 0.54727554
iteration 260, objective = 0.54471296
iteration 270, objective = 0.544299
iteration 280, objective = 0.54173803
iteration 290, objective = 0.54505545
iteration 300, objective = 0.54363304
iteration 310, objective = 0.5427178
iteration 320, objective = 0.54215485
iteration 330, objective = 0.54277897
iteration 340, objective = 0.5393973
iteration 350, objective = 0.5434398
iteration 360, objective = 0.5427398
iteration 370, objective = 0.54314435
iteration 380, objective = 0.53990793
iteration 390, objective = 0.54345894
iteration 400, objective = 0.5412914
iteration 410, objective = 0.5371492
iteration 420, objective = 0.5400574
iteration 430, objective = 0.5378474
iteration 440, objective = 0.53764486
iteration 450, objective = 0.53639424
iteration 460, objective = 0.5378227
iteration 470, objective = 0.53672457
iteration 480, objective = 0.53929245
iteration 490, objective = 0.5363729
iteration 500, objective = 0.5380682
iteration 510, objective = 0.5389014
iteration 520, objective = 0.5386449
iteration 530, objective = 0.53421676
iteration 540, objective = 0.53760684
iteration 550, objective = 0.5375755
iteration 560, objective = 0.5361507
iteration 570, objective = 0.5370755
iteration 580, objective = 0.53628033
iteration 590, objective = 0.53795755
iteration 600, objective = 0.5344045
```

In [94]:

```
    for i in range(600):
        obj = trainingStep(modelMC, X_train)
        if (i % 10 == 9): print("iteration " + str(i+1) + ", objective = " + str(obj))
```

```
iteration 10, objective = 0.5326002
iteration 20, objective = 0.5337547
iteration 30, objective = 0.5348592
iteration 40, objective = 0.5333945
iteration 50, objective = 0.5337779
iteration 60, objective = 0.53178316
iteration 70, objective = 0.53358996
iteration 80, objective = 0.5333319
iteration 90, objective = 0.53025126
iteration 100, objective = 0.5328586
iteration 110, objective = 0.5327846
iteration 120, objective = 0.5334447
iteration 130, objective = 0.5328021
iteration 140, objective = 0.5310075
iteration 150, objective = 0.5327859
iteration 160, objective = 0.53231776
iteration 170, objective = 0.5324224
iteration 180, objective = 0.5300671
iteration 190, objective = 0.5288187
iteration 200, objective = 0.53194344
iteration 210, objective = 0.5303365
iteration 220, objective = 0.53093415
iteration 230, objective = 0.53158236
iteration 240, objective = 0.5297749
iteration 250, objective = 0.53151655
iteration 260, objective = 0.5314349
iteration 270, objective = 0.5277928
iteration 280, objective = 0.5314529
iteration 290, objective = 0.53008497
iteration 300, objective = 0.52824694
iteration 310, objective = 0.5267406
iteration 320, objective = 0.53021646
iteration 330, objective = 0.52800006
iteration 340, objective = 0.5272641
iteration 350, objective = 0.52776515
iteration 360, objective = 0.5294466
iteration 370, objective = 0.5290521
iteration 380, objective = 0.52797383
iteration 390, objective = 0.5289345
iteration 400, objective = 0.52735424
iteration 410, objective = 0.5270269
iteration 420, objective = 0.5279036
iteration 430, objective = 0.5276952
iteration 440, objective = 0.5276272
iteration 450, objective = 0.52910197
iteration 460, objective = 0.52851444
iteration 470, objective = 0.5264541
iteration 480, objective = 0.5295156
iteration 490, objective = 0.5272901
iteration 500, objective = 0.5262562
iteration 510, objective = 0.52695566
iteration 520, objective = 0.5266715
iteration 530, objective = 0.5270011
iteration 540, objective = 0.5261657
iteration 550, objective = 0.5250126
iteration 560, objective = 0.5241653
iteration 570, objective = 0.5274565
iteration 580, objective = 0.5269251
iteration 590, objective = 0.5270935
iteration 600, objective = 0.527153
```

In [95]:

```
def MSE(predictions, labels):
    differences = [(x-y)**2 for x,y in zip(predictions,labels)]
    return sum(differences) / len(differences)

pred_train = []
for u,i,j,_ in X_train:
    pred = modelMC.predict(itemIDs[i],itemIDs[j]).numpy()
    pred_train.append(pred)
print("MSE train = ",MSE(pred_train, y_train))

pred_test = []
for u,i,j in X_test:
    pred = modelMC.predict(itemIDs[i],itemIDs[j]).numpy()
    pred_test.append(pred)
print("MSE test = ",MSE(pred_test, y_test))
```

```
MSE train =  0.04514250267013937
MSE test =  13.737801945380893
```

(c) FPMC model which includes user, item and previous item

In [37]:
```
import dateutil
# itemsPerUser[u], userIDs, itemIDs, interactionsWithPrevious, items, optimizer
userIDs = {}
itemIDs = {}
interactions = []
interactionsPerUser = defaultdict(list)

for d in data:
    u = d['user_id']
    i = d['book_id']
    t = d['date_added']
    r = d['rating']
    dt = dateutil.parser.parse(t)
    t = int(dt.timestamp())
    #print(t)
    if userCount[u]<3 : continue
    if not u in userIDs: userIDs[u] = len(userIDs)
    if not i in itemIDs: itemIDs[i] = len(itemIDs)
    interactions.append((t,u,i,r))
    interactionsPerUser[u].append((t,i,r))
#interactions[0]
interactions.sort()
print(len(interactions))

itemIDs['dummy'] = len(itemIDs)
X_train = []
X_test = []
y_train = []
y_test = []
for u in interactionsPerUser:
    interactionsPerUser[u].sort()
    lastItem = 'dummy'
    length = len(interactionsPerUser[u])
    cnt = 0
    for (t,i,r) in interactionsPerUser[u]:
        cnt+=1
        if cnt < length:
            X_train.append((u,i,lastItem,r))
            y_train.append(r)
            lastItem = i
        else :
            X_test.append((u,i,lastItem))
            y_test.append(r)
print(len(X_train),len(X_test))
print(len(y_train),len(y_test))

itemsPerUser = defaultdict(set)
for u,i,_,_ in X_train: #(u,i,lastItem,r)
    itemsPerUser[u].add(i)
items = list(itemIDs.keys())
optimizer = tf.keras.optimizers.Adam(0.1)
```

```
18927
18291 636
18291 636
```

In [58]:
```
class FPMC(tf.keras.Model):
    def __init__(self, K, lamb, UI = 1, IJ = 1):
        super(FPMC, self).__init__()
        # Initialize variables
        self.betaU = tf.Variable(tf.random.normal([len(userIDs)],stddev=0.001))
        self.betaI = tf.Variable(tf.random.normal([len(itemIDs)],stddev=0.001))
        self.gammaUI = tf.Variable(tf.random.normal([len(userIDs),K],stddev=0.001))
        self.gammaIU = tf.Variable(tf.random.normal([len(itemIDs),K],stddev=0.001))
        self.gammaIJ = tf.Variable(tf.random.normal([len(itemIDs),K],stddev=0.001))
        self.gammaJI = tf.Variable(tf.random.normal([len(itemIDs),K],stddev=0.001))
        # Regularization coefficient
        self.lamb = lamb
        # Which terms to include
        self.UI = UI
```

```
            self.IJ = IJ

        # Prediction for a single instance
        def predict(self, u, i, j):
            p = self.betaU[u]+self.betaI[i] + self.UI * tf.tensordot(self.gammaUI[u], self.gammaIU[i], 1
                                self.IJ * tf.tensordot(self.gammaIJ[i], self.gammaJI[j], 1)
            return p

        # Regularizer
        def reg(self):
            return self.lamb * (tf.nn.l2_loss(self.betaU) +\
                                tf.nn.l2_loss(self.betaI) +\
                                tf.nn.l2_loss(self.gammaUI) +\
                                tf.nn.l2_loss(self.gammaIU) +\
                                tf.nn.l2_loss(self.gammaIJ) +\
                                tf.nn.l2_loss(self.gammaJI))

        def call(self, sampleU, # user
                       sampleI, # item
                       sampleJ, # previous item
                       sampleK, sampleR): # negative item
            u = tf.convert_to_tensor(sampleU, dtype=tf.int32)
            i = tf.convert_to_tensor(sampleI, dtype=tf.int32)
            j = tf.convert_to_tensor(sampleJ, dtype=tf.int32)
            k = tf.convert_to_tensor(sampleK, dtype=tf.int32)
            gamma_ui = tf.nn.embedding_lookup(self.gammaUI, u)
            gamma_iu = tf.nn.embedding_lookup(self.gammaIU, i)
            gamma_ij = tf.nn.embedding_lookup(self.gammaIJ, i)
            gamma_ji = tf.nn.embedding_lookup(self.gammaJI, j)
            beta_u = tf.nn.embedding_lookup(self.betaU, u)
            beta_i = tf.nn.embedding_lookup(self.betaI, i)
            x_uij = beta_u + beta_i + self.UI * tf.reduce_sum(tf.multiply(gamma_ui, gamma_iu), 1) +\
                            self.IJ * tf.reduce_sum(tf.multiply(gamma_ij, gamma_ji), 1)
            gamma_uk = tf.nn.embedding_lookup(self.gammaUI, u)
            gamma_ku = tf.nn.embedding_lookup(self.gammaIU, k)
            gamma_kj = tf.nn.embedding_lookup(self.gammaIJ, k)
            gamma_jk = tf.nn.embedding_lookup(self.gammaJI, j)
            beta_u = tf.nn.embedding_lookup(self.betaU, u)
            beta_k = tf.nn.embedding_lookup(self.betaI, k)
            x_ukj = beta_u + beta_k + self.UI * tf.reduce_sum(tf.multiply(gamma_uk, gamma_ku), 1) +\
                            self.IJ * tf.reduce_sum(tf.multiply(gamma_kj, gamma_jk), 1)

            mse = tf.keras.losses.MeanSquaredError()
            return -tf.reduce_mean(tf.math.log(tf.math.sigmoid(x_uij - x_ukj)))+ mse(x_uij,sampleR)

    modelFPMC = FPMC(5, 0.00001)
```

In [59]:

```
    def trainingStep(model, interactions):
        with tf.GradientTape() as tape:
            sampleU, sampleI, sampleJ, sampleK, sampleR = [], [], [], [], []
            for _ in range(100000):
                u,i,j,r = random.choice(interactions) # positive sample
                k = random.choice(items) # negative sample
                while k in itemsPerUser[u]:
                    k = random.choice(items)
                sampleU.append(userIDs[u])
                sampleI.append(itemIDs[i])
                sampleJ.append(itemIDs[j])
                sampleK.append(itemIDs[k])
                sampleR.append(float(r))

            loss = model(sampleU,sampleI,sampleJ,sampleK,sampleR)
            loss += model.reg()
        gradients = tape.gradient(loss, model.trainable_variables)
        optimizer.apply_gradients((grad, var) for
                                  (grad, var) in zip(gradients, model.trainable_variables)
                                  if grad is not None)
        return loss.numpy()
```

In [60]:

```
    for i in range(300):
        obj = trainingStep(modelFPMC, X_train)
        if (i % 10 == 9): print("iteration " + str(i+1) + ", objective = " + str(obj))
```

```
iteration 10, objective = 1.5656526
iteration 20, objective = 1.2348145
iteration 30, objective = 0.895774
iteration 40, objective = 0.65240955
iteration 50, objective = 0.50799006
iteration 60, objective = 0.42495096
iteration 70, objective = 0.37038845
iteration 80, objective = 0.33357084
iteration 90, objective = 0.3100459
iteration 100, objective = 0.29241621
iteration 110, objective = 0.27959573
iteration 120, objective = 0.26924568
iteration 130, objective = 0.26147848
iteration 140, objective = 0.25636426
iteration 150, objective = 0.25175643
iteration 160, objective = 0.24752866
iteration 170, objective = 0.24323055
iteration 180, objective = 0.2398307
iteration 190, objective = 0.23606
iteration 200, objective = 0.23436159
iteration 210, objective = 0.23050463
iteration 220, objective = 0.22710988
iteration 230, objective = 0.22681578
iteration 240, objective = 0.22320634
iteration 250, objective = 0.22324425
iteration 260, objective = 0.21927588
iteration 270, objective = 0.21816015
iteration 280, objective = 0.21729755
iteration 290, objective = 0.21482633
iteration 300, objective = 0.2130304
```

In [61]:

```
for i in range(300):
    obj = trainingStep(modelFPMC, X_train)
    if (i % 10 == 9): print("iteration " + str(i+1) + ", objective = " + str(obj))
```

```
iteration 10, objective = 0.21171278
iteration 20, objective = 0.21088418
iteration 30, objective = 0.20812409
iteration 40, objective = 0.20991042
iteration 50, objective = 0.20626906
iteration 60, objective = 0.20595244
iteration 70, objective = 0.20624606
iteration 80, objective = 0.20444506
iteration 90, objective = 0.20334204
iteration 100, objective = 0.2020027
iteration 110, objective = 0.20169473
iteration 120, objective = 0.20018199
iteration 130, objective = 0.20017701
iteration 140, objective = 0.19955021
iteration 150, objective = 0.19904074
iteration 160, objective = 0.19734436
iteration 170, objective = 0.19762924
iteration 180, objective = 0.19678152
iteration 190, objective = 0.19634858
iteration 200, objective = 0.19579735
iteration 210, objective = 0.19402504
iteration 220, objective = 0.19615518
iteration 230, objective = 0.1951198
iteration 240, objective = 0.19289783
iteration 250, objective = 0.1921893
iteration 260, objective = 0.19239932
iteration 270, objective = 0.19240528
iteration 280, objective = 0.1937513
iteration 290, objective = 0.19290987
iteration 300, objective = 0.19064425
```

In [62]:

```
def MSE(predictions, labels):
    differences = [(x-y)**2 for x,y in zip(predictions,labels)]
    return sum(differences) / len(differences)

pred_train = []
for u,i,j,_ in X_train:
    pred = modelFPMC.predict(userIDs[u],itemIDs[i],itemIDs[j]).numpy()
    pred_train.append(pred)
print("MSE train = ",MSE(pred_train, y_train))

pred_test = []
```

```
    for u,i,j in X_test:
        pred = modelFPMC.predict(userIDs[u],itemIDs[i],itemIDs[j]).numpy()
        pred_test.append(pred)
    print("MSE test = ",MSE(pred_test, y_test))
```

```
MSE train =  0.020084221563801165
MSE test =  10.812561900693636
```

In [65]:

```
    for i in range(300):
        obj = trainingStep(modelFPMC, X_train)
        if (i % 10 == 9): print("iteration " + str(i+1) + ", objective = " + str(obj))
```

```
iteration 10, objective = 0.19159523
iteration 20, objective = 0.1904184
iteration 30, objective = 0.19104567
iteration 40, objective = 0.19058432
iteration 50, objective = 0.18850201
iteration 60, objective = 0.18892846
iteration 70, objective = 0.18817797
iteration 80, objective = 0.18905973
iteration 90, objective = 0.18695144
iteration 100, objective = 0.18653814
iteration 110, objective = 0.18768285
iteration 120, objective = 0.18654492
iteration 130, objective = 0.18747246
iteration 140, objective = 0.18606657
iteration 150, objective = 0.1880572
iteration 160, objective = 0.18655753
iteration 170, objective = 0.18653783
iteration 180, objective = 0.18628897
iteration 190, objective = 0.18616599
iteration 200, objective = 0.18638262
iteration 210, objective = 0.18388262
iteration 220, objective = 0.18572797
iteration 230, objective = 0.18442294
iteration 240, objective = 0.18305612
iteration 250, objective = 0.18171448
iteration 260, objective = 0.1820064
iteration 270, objective = 0.18189988
iteration 280, objective = 0.18252403
iteration 290, objective = 0.18088749
iteration 300, objective = 0.18198305
```

In [66]:

```
    def MSE(predictions, labels):
        differences = [(x-y)**2 for x,y in zip(predictions,labels)]
        return sum(differences) / len(differences)

    pred_train = []
    for u,i,j,_ in X_train:
        pred = modelFPMC.predict(userIDs[u],itemIDs[i],itemIDs[j]).numpy()
        pred_train.append(pred)
    print("MSE train = ",MSE(pred_train, y_train))

    pred_test = []
    for u,i,j in X_test:
        pred = modelFPMC.predict(userIDs[u],itemIDs[i],itemIDs[j]).numpy()
        pred_test.append(pred)
    print("MSE test = ",MSE(pred_test, y_test))
```

```
MSE train =  0.019621448033968612
MSE test =  11.231530281475283
```

Ans for Problem3 :

a. regular latent factor model:

```
MSE train =  0.013160128282292967
MSE test =  1.3137676309457482
```

b. non-personalized Markov Chain model with previous item and current item:

```
MSE train =  0.04514250267013937
MSE test =  13.737801945380893
```

c. Markov Chain model with user, previous item and current item:

```
MSE train =  0.019621448033968612
MSE test =  11.231530281475283
```

In [ ]:

Ans for Problem4 :

a.