Chapter 2.
Making Recommendations

- 한 무리 사람들의 선호 정보를 이용하여 다른 사람들이 원할 만한 것을 추천하는 방법
  - 온라인 쇼핑몰의 제품추천
  - 웹 사이트 추천
  - 음악, 영화 검색 도우미 등

1. 협업 필터링(Collaborative Filtering)

- 기술없이 가장 간단하게 추천받는 방법
  - 사용자와 같은 것을 좋아하고 다른 사람들보다 세련된 취향을 가진 몇몇 친구에게 물어보기
- 협업 필터링 알고리즘
  - 큰 무리의 사람들을 검색해서 사용자와 유사한 취향을 가진 작은 집합을 발견하는 방식으로 동작

2. 선호정보 수집(Collecting Preferences)

- 사람과 그들의 선호도를 표현할 방식이 필요
  - Python의 nested dictionary 이용
- 데이터 집합 : recommendations.py

```
# A dictionary of movie critics and their ratings of a small
# set of movies
critics={'Lisa Rose': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.5,
 'Just My Luck': 3.0, 'Superman Returns': 3.5, 'You, Me and Dupree': 2.5,
 'The Night Listener': 3.0},
'Gene Seymour': {'Lady in the Water': 3.0, 'Snakes on a Plane': 3.5,
 'Just My Luck': 1.5, 'Superman Returns': 5.0, 'The Night Listener': 3.0,
 'You, Me and Dupree': 3.5},
'Michael Phillips': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.0,
 'Superman Returns': 3.5, 'The Night Listener': 4.0},
'Claudia Puig': {'Snakes on a Plane': 3.5, 'Just My Luck': 3.0,
 'The Night Listener': 4.5, 'Superman Returns': 4.0,
 'You, Me and Dupree': 2.5},
'Mick LaSalle': {'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0,
 'Just My Luck': 2.0, 'Superman Returns': 3.0, 'The Night Listener': 3.0,
 'You, Me and Dupree': 2.0},
'Jack Matthews': {'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0,
 'The Night Listener': 3.0, 'Superman Returns': 5.0, 'You, Me and Dupree': 3.5},
'Toby': {'Snakes on a Plane':4.5,'You, Me and Dupree':1.0,'Superman Returns':4.0}}
```

- 데이터 표현
  - 선호도를 반드시 숫자화 시켜야
  - recommendations.py
    - 한 영화에 대해 좋아하는 비평 정도를 1에서 5사이의 값으로 표현
  - 사용자의 선택/행위를 숫자로 표현한 예

Concert tickets		Online shopping		Site recommender	
Bought	1	Bought	2	Liked	1
Didn't buy	0	Browsed	1	No vote	0
		Didn't buy	0	Disliked	–1

- Python을 대화식으로 실행
  - 생성파일은 python/lib 등에 저장
  - 또는 파일 저장 디렉토리에서 Python 시작

- 선호 정보를 dictionary를 이용하여 저장
  - 검색과 수정 용이
- 큰 데이터 집합인 경우
  - 선호 정보를 데이터베이스에 저장

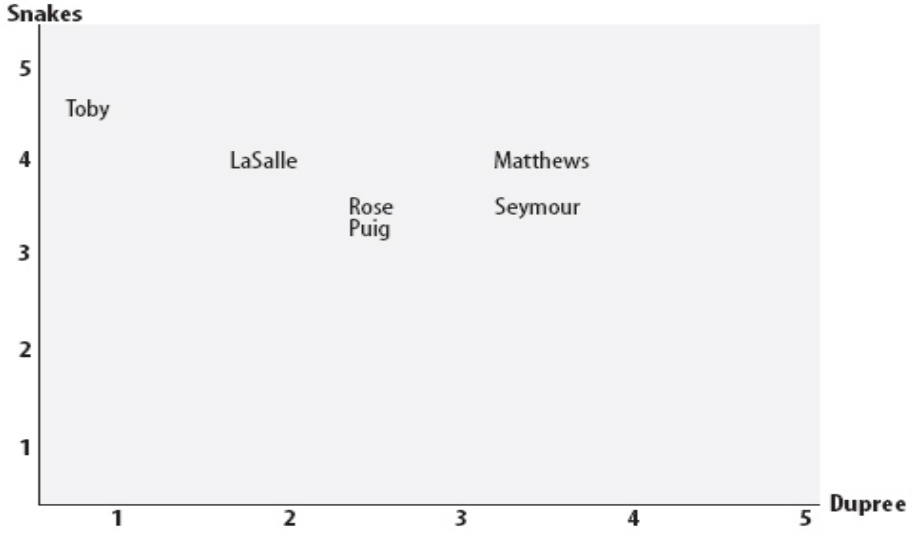
```
c:\code\collective\chapter2> python
Python 2.4.1 (#65, Mar 30 2005, 09:13:57) [MSC v.1310 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>> from recommendations import critics
>> critics['Lisa Rose']['Lady in the Water']
2.5
>> critics['Toby']['Snakes on a Plane']=4.5
>> critics['Toby']
{'Snakes on a Plane':4.5,'You, Me and Dupree':1.0}
```

3. 유사 사용자 찾기(Finding Similar Users)

- 사용자간 취향이 얼마나 비슷한지 결정
- 유사도 계산방법
  - 유클리디안 거리점수(Euclidean distance score)
    - 가장 간단한 방법
  - 피어슨 상관점수(Pearson correlation score)

Euclidean Distance Score

- 사람들 간의 선호 공간 예시
  - 사람들이 점수를 공통으로 매긴 항목을 취해 차트의 축으로 사용하고, 차트에 사람들을 출력
  - 가까울수록 선호도가 유사함을 의미
    - 다차원 데이터도 동일한 원칙 적용



- 유클리디안 거리 공식
  - 두 점 (p_1, p_2, …, p_n)과 (q_1, q_2, …, q_n)의 거리

$$\sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\ldots+(p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}$$

- Toby와 LaSalle간의 차트 거리 계산
  - 각 축을 기준으로 차이를 구하고 제곱을 구한 후, 그것을 합함
  - 이 총합의 제곱근이 거리점수
  - 숫자제곱 구하기 : pow(n, 2)
  - 제곱근 구하기 : sqrt
  - 유사한 사람일수록 작은 값을 가짐

```
>>> from math import sqrt
>>> sqrt(pow(4.5-4, 2) + pow(1-2, 2))
1.1180339887498949
```

- 유사할수록 더 높은 값을 가지는 함수로 변환
  - 앞 함수에 1을 더하고 역을 취함
    - 항상 0과 1사이의 값을 리턴
    - 두 사람이 동일한 선호도를 가지는 경우 1 리턴

```
>>> 1 / (1 + sqrt(pow(4.5-4, 2) + pow(1-2, 2)))
0.47213595499957939
```

- 다음 코드를 recommendations.py에 추가

```
from math import sqrt

# Returns a distance-based similarity score for person1 and person2
def sim_distance(prefs,person1,person2):
  # Get the list of shared_items
  si={}
  for item in prefs[person1]:
    if item in prefs[person2]: si[item]=1

  # if they have no ratings in common, return 0
  if len(si)==0: return 0

  # Add up the squares of all the differences
  sum_of_squares=sum([pow(prefs[person1][item]-prefs[person2][item],2)
                      for item in prefs[person1] if item in prefs[person2]])

  return 1/(1+sqrt(sum_of_squares))
```

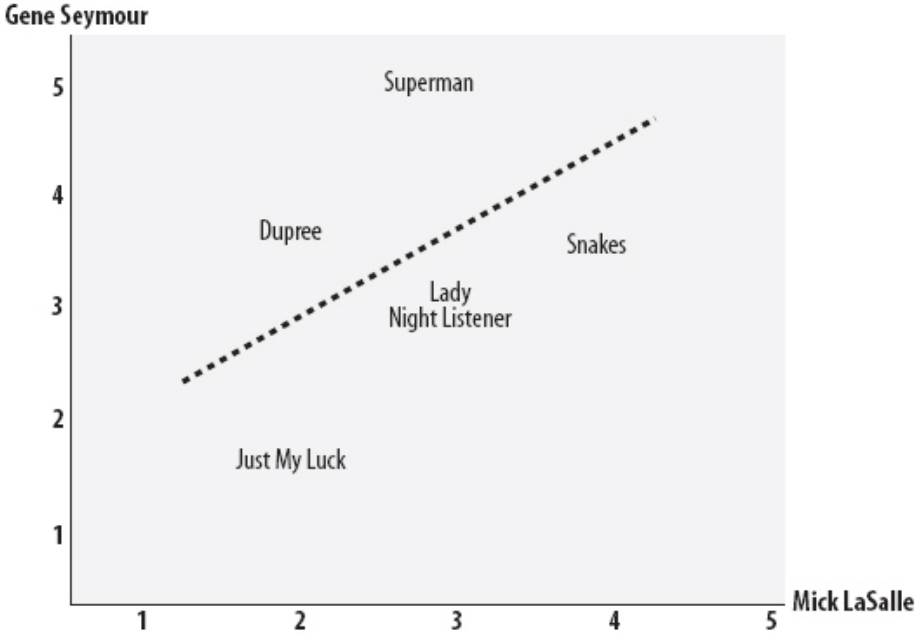
– 실행결과

```
C:\temp\python\chapter2>python
Python 2.5.2 (r252:60911, Feb 21 2008, 13:11:45) [MSC v.1310 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import recommendations
>>> recommendations.sim_distance(recommendations.critics,
... 'Lisa Rose', 'Gene Seymour')
0.29429805508554946
>>>
>>> recommendations.sim_distance(recommendations.critics,
... 'Lisa Rose', 'Claudia Puig')
0.38742588672279304
>>> recommendations.sim_distance(recommendations.critics,
... 'Lisa Rose', 'Toby')
0.34833147735478831
>>> recommendations.sim_distance(recommendations.critics,
... 'Lisa Rose', 'Mick LaSalle')
0.41421356237309509
>>> recommendations.sim_distance(recommendations.critics,
... 'Lisa Rose', 'Jack Matthews')
0.34054242658316669
```

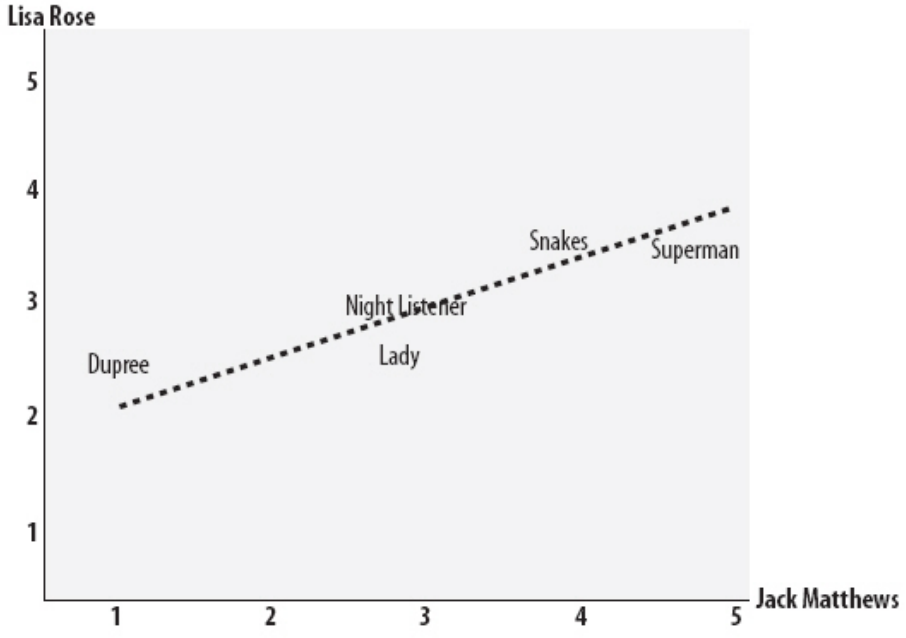
Pearson Correlation Score

- 상관계수
  - 두 개의 데이터 집합이 한 직선으로 얼마나 잘 표현되는가를 나타내는 측정값
  - 잘 정규화되지 않은 데이터의 경우에 보다 나은 결과 제공

- 산포도로 두 영화 평론가 비교하기
  - 직선 : 최적 맞춤선(best-fit line)
    - **차트 내 모든 항목들과 가능한 가장 가까운 직선**
    - 모든 영화를 동일하게 평가했다면 **대각선**이 되고 이 선위에 모든 항목들이 위치하게 됨 → **완전 상관점수 1**이 됨
    - 아래는 상관점수가 0.4 정도



- 높은 상관점수를 가진 두 평론가
  - 상관점수가 0.75 정도
  - 점수 부풀리기(grade inflation) 특성
    - 만일 한 평론가가 다른 평론가에 비해 더 높은 점수를 주었어도, 그들 간의 점수 차이가 일정하다면 두 사람은 완벽한 상관도 가짐



  - 적용분야에 따라 이러한 특성이 유익할 수도, 그렇지 않을 수도 있음

  - 피어슨 상관계수 공식
    - 두 변수간 연관 강도용 측정 지표
    - 1 ~-1 사이의 값

      1 : 완전히 연관되었음

      0 : 연관이 전혀 없음

      -1 : 완전히 반대로 연관되었음

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{\left(\sum X^2 - \frac{(\sum X)^2}{N}\right)\left(\sum Y^2 - \frac{(\sum Y)^2}{N}\right)}}$$

```python
# Returns the Pearson correlation coefficient for p1 and p2
def sim_pearson(prefs,p1,p2):
  # Get the list of mutually rated items
  si={}
  for item in prefs[p1]:
    if item in prefs[p2]: si[item]=1

  # if they are no ratings in common, return 0
  if len(si)==0: return 0

  # Sum calculations
  n=len(si)

  # Sums of all the preferences
  sum1=sum([prefs[p1][it] for it in si])
  sum2=sum([prefs[p2][it] for it in si])

  # Sums of the squares
  sum1Sq=sum([pow(prefs[p1][it],2) for it in si])
  sum2Sq=sum([pow(prefs[p2][it],2) for it in si])

  # Sum of the products
  pSum=sum([prefs[p1][it]*prefs[p2][it] for it in si])

  # Calculate r (Pearson score)
  num=pSum-(sum1*sum2/n)
  den=sqrt((sum1Sq-pow(sum1,2)/n)*(sum2Sq-pow(sum2,2)/n))
  if den==0: return 0

  r=num/den

  return r
```

– 실행결과

```
>>> reload(recommendations)
<module 'recommendations' from 'recommendations.pyc'>
>>> recommendations.sim_pearson(recommendations.critics,
... 'Lisa Rose', 'Gene Seymour')
0.39605901719066977
```

다른 유사도 측정 지표

- 데이터 집합의 유사도를 측정하는 다른 방법도 많음
  - 어디에 적용할 것인가에 따라 다른 선택 가능
- 기타 유사도 측정 지표
  - 자카드 계수(Jaccard coefficient)
  - 맨해튼 거리(Manhattan distance)
  - 참고링크
    - http://en.wikipedia.org/wiki/Metric_%28mathematics%29#Examples

평론가 순위 구하기

- 특정인과 유사한 취향을 가진 사람들의 순서 구하기

```
# Returns the best matches for person from the prefs dictionary.
# Number of results and similarity function are optional params.
def topMatches(prefs,person,n=5,similarity=sim_pearson):
  scores=[(similarity(prefs,person,other),other)
                  for other in prefs if other!=person]
  scores.sort()
  scores.reverse()
  return scores[0:n]
```

– 실행결과

```
>>> from recommendations import *
>>> reload(recommendations)
<module 'recommendations' from 'recommendations.pyc'>
>>> recommendations.topMatches(recommendations.critics, 'Toby', n=3)
[(0.99124070716192991, 'Lisa Rose'), (0.92447345164190486, 'Mick LaSalle'), (0.8
9340514744156474, 'Claudia Puig')]
>>> recommendations.topMatches(recommendations.critics, 'Toby', n=3,
... similarity=sim_distance)
[(0.40000000000000002, 'Mick LaSalle'), (0.38742588672279304, 'Michael Phillips'
), (0.35678917232533092, 'Claudia Puig')]
>>>
```

4. 항목 추천 (Recommending Items)

- 평론가 순위를 결정하는 가중점수를 만들어 항목점수를 계산
- Toby에 대한 추천 영화 계산

Critic	Similarity	Night	S.xNight	Lady	S.xLady	Luck	S.xLuck
Rose	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Seymour	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Puig	0.89	4.5	4.02			3.0	2.68
LaSalle	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Matthews	0.66	3.0	1.99	3.0	1.99		
Total			12.89		8.38		8.07
Sim. Sum			3.84		2.95		3.18
Total/Sim. Sum			3.35		2.83		2.53

- 나와 유사한 사람의 평가가 나와는 다른 사람의 평가보다 전체 점수에 더 큰 기여
- 단순 가중치 합계는
  - 많은 평론가가 리뷰한 영화가 큰 이득 얻음
  - 가중치 합계를 해당 영화를 리뷰한 모든 평론가들의 유사도 합으로 나눌 필요

```
# Gets recommendations for a person by using a weighted average
# of every other user's rankings
def getRecommendations(prefs,person,similarity=sim_pearson):
  totals={}
  simSums={}
  for other in prefs:
    # don't compare me to myself
    if other==person: continue
    sim=similarity(prefs,person,other)

    # ignore scores of zero or lower
    if sim<=0: continue
    for item in prefs[other]:

      # only score movies I haven't seen yet
      if item not in prefs[person] or prefs[person][item]==0:
        # Similarity * Score
        totals.setdefault(item,0)
        totals[item]+=prefs[other][item]*sim
        # Sum of similarities
        simSums.setdefault(item,0)
        simSums[item]+=sim

  # Create the normalized list
  rankings=[(total/simSums[item],item) for item,total in totals.items()]

  # Return the sorted list
  rankings.sort()
  rankings.reverse()
  return rankings
```

- Toby를 위한 추천 영화 생성결과

```
>>> from recommendations import *
>>> getRecommendations(critics, 'Toby')
[(3.3477895267131013, 'The Night Listener'), (2.8325499182641614, 'Lady in the W
ater'), (2.5309807037655645, 'Just My Luck')]
>>>
>>> getRecommendations(critics, 'Toby', similarity=sim_distance)
[(3.457128694491423, 'The Night Listener'), (2.7785840038149239, 'Lady in the Wa
ter'), (2.4224820423619167, 'Just My Luck')]
```

5. 제품 매칭 (Matching Products)

- 한 제품과 다른 제품이 얼마나 비슷한가
  - 쇼핑 웹 사이트에 자주 방문하지 않는 사람들을 위한 기능
- "파이썬 프로그래밍"책을 다룬 아마존 웹 페이지

Customers who bought this item also bought

Learning Python, Second Edition by Mark Lutz

Python Cookbook by Alex Martelli

Python in a Nutshell by Alex Martelli

Python Essential Reference (2nd Edition) by David Beazley

Foundations of Python Network Programming (Foundations) by John Goerzen

▸ **Explore similar items** : Books (42)

- 특정 물건을 좋아한 사람들이 좋아한 다른 것들을 살펴보아 유사도 구함
- 앞의 두 사람 간 유사도 측정방식과 동일
  - 다만 사람을 물건으로 대치하면 됨
  - 사전 구조를 다음과 같이 변경

```
{'Lisa Rose': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.5},
'Gene Seymour': {'Lady in the Water': 3.0, 'Snakes on a Plane': 3.5}}
```

to:

```
{'Lady in the Water':{'Lisa Rose':2.5,'Gene Seymour':3.0},
'Snakes on a Plane':{'Lisa Rose':3.5,'Gene Seymour':3.5}} etc..
```

```
def transformPrefs(prefs):
  result={}
  for person in prefs:
    for item in prefs[person]:
      result.setdefault(item,{})

      # Flip item and person
      result[item][person]=prefs[person][item]
  return result
```

- 실행결과
  - "Superman Returns"와 가장 유사한 영화 찾기

```
>>> movies = transformPrefs(critics)
>>>
>>> topMatches(movies, 'Superman Returns')
[(0.65795169495976946, 'You, Me and Dupree'), (0.48795003647426888, 'Lady in the
 Water'), (0.11180339887498941, 'Snakes on a Plane'), (-0.17984719479905439, 'Th
e Night Listener'), (-0.42289003161103106, 'Just My Luck')]
```

- 앞 실행결과에서 리턴된 음수의 의미
  - "Superman Returns"를 좋아한 사람들이 "Just My Luck"을 싫어하는 경향이 있음을 의미

- 영화에 대한 평론가를 추천하고자 하는 경우

```
>>> getRecommendations(movies, 'Just My Luck')
[(4.0, 'Michael Phillips'), (3.0, 'Jack Matthews')]
>>> getRecommendations(movies, 'Superman Returns')
[]
```

7. 항목기반 필터링(Item-Based Filtering)

- 사용자 기반 협력 필터링 방법
  - 사용자들이 평가한 랭킹 정보 데이터 있어야
  - 큰 사이트에서는 느리게 동작
  - 많은 제품 보유 사이트에선 사용자간 중첩 거의 없어 유사 판단 어려움
- 항목 기반 필터링
  - 각 항목별로 가장 유사한 항목 **미리 계산**
  - 항목간 비교는 자주 바뀌지 않음

항목 비교 데이터 세트 생성

- 유사항목 담은 완전한 데이터 세트 구축 함수
  - Recommendations.py에 추가

```
def calculateSimilarItems(prefs,n=10):
  # Create a dictionary of items showing which other items they
  # are most similar to.
  result={}
  # Invert the preference matrix to be item-centric
  itemPrefs=transformPrefs(prefs)
  c=0
  for item in itemPrefs:
    # Status updates for large datasets
    c+=1
    if c%100==0: print "%d / %d" % (c,len(itemPrefs))
    # Find the most similar items to this one
    scores=topMatches(itemPrefs,item,n=n,similarity=sim_distance)
    result[item]=scores
  return result
```

```
>>> itemsim=calculateSimilarItems(critics)
>>> itemsim
{'Lady in the Water': [(0.4494897427831781, 'You, Me and Dupree'), (0.3874258867
2279304, 'The Night Listener'), (0.34833147735478831, 'Snakes on a Plane'), (0.3
4833147735478831, 'Just My Luck'), (0.2402530733520421, 'Superman Returns')], 'S
nakes on a Plane': [(0.34833147735478831, 'Lady in the Water'), (0.3203772410170
4074, 'The Night Listener'), (0.3090169943749474, 'Superman Returns'), (0.255396
79298968671, 'Just My Luck'), (0.18863786477264649, 'You, Me and Dupree')], 'Jus
t My Luck': [(0.34833147735478831, 'Lady in the Water'), (0.32037724101704074, '
You, Me and Dupree'), (0.29893508442482553, 'The Night Listener'), (0.2553967929
8968671, 'Snakes on a Plane'), (0.20799159651347807, 'Superman Returns')], 'Supe
rman Returns': [(0.3090169943749474, 'Snakes on a Plane'), (0.25265030858707199,
```

- 항목 유사도를 최신으로 유지하기 충분할 정도로만 이 함수 실행
  - 사용자수와 평가 수 작을 때는 자주 실행
  - 사용자수가 많아질수록 항목들 간 유사도 점수는 안정적이 됨

추천 생성

- Toby를 위한 항목기반 추천
  - 사용자가 평가했던 항목과 유사한 항목을 찾은 후, 그들간에 유사정도를 통해 가중치 계산
  - 이전과 달리, 평론가 정보 없고 Toby가 평가한 영화와 평가하지 않은 영화 간의 비교만 포함

Movie	Rating	Night	R.xNight	Lady	R.xLady	Luck	R.xLuck
Snakes	4.5	0.182	0.818	0.222	0.999	0.105	0.474
Superman	4.0	0.103	0.412	0.091	0.363	0.065	0.258
Dupree	1.0	0.148	0.148	0.4	0.4	0.182	0.182
Total		0.433	1.378	0.713	1.764	0.352	0.914
Normalized			3.183		2.598		2.473

```
def getRecommendedItems(prefs,itemMatch,user):
  userRatings=prefs[user]
  scores={}
  totalSim={}
  # Loop over items rated by this user
  for (item,rating) in userRatings.items( ):

    # Loop over items similar to this one
    for (similarity,item2) in itemMatch[item]:

      # Ignore if this user has already rated this item
      if item2 in userRatings: continue
      # Weighted sum of rating times similarity
      scores.setdefault(item2,0)
      scores[item2]+=similarity*rating
      # Sum of all the similarities
      totalSim.setdefault(item2,0)
      totalSim[item2]+=similarity

  # Divide each total score by total weighting to get an average
  rankings=[(score/totalSim[item],item) for item,score in scores.items( )]

  # Return the rankings from highest to lowest
  rankings.sort( )
  rankings.reverse( )
  return rankings
```

- 사용자 기반 협력 필터링 결과

```
>>> getRecommendations(critics, 'Toby')
[(3.3477895267131013, 'The Night Listener'), (2.8325499182641614, 'Lady in
the Water'), (2.5309807037655645, 'Just My Luck')]
```

- 항목 기반 협력 필터링 결과

```
>>> getRecommendedItems(critics, itemsim, 'Toby')
[(3.1667425234070894, 'The Night Listener'), (2.9366294028444351, 'Just My Luck'
), (2.868767392626467, 'Lady in the Water')]
```

- getRecommendedItems 함수는 항목 유사도 데이터 세트가 미리 준비되어 있어, 다른 평론가들과의 유사도 점수는 계산하지 않음

8. MovieLens 데이터세트 이용하기

- MovieLens
  - 미네소타 대학의 GroupLens 프로젝트에 의해 개발
  - 영화, 책, 농담에 관한 평가 데이터 구축
  - Download
    - http://www.grouplens.org/node/73
    - 1000만, 100만 또는 10만 영화 평가자료 제공
    - 10만 데이터 세트 압축파일 안의 u.item 파일 사용
      - 943명이 1,682개 영화 평가
      - 각 사용자는 적어도 20개 영화를 평가

- u.item 파일 형식
  - movie id | movie title | release date | video release date | IMDb URL | unknown | Action | Adventure | Animation | Children's | Comedy | Crime | Documentary | Drama | Fantasy | Film-Noir | Horror | Musical | Mystery | Romance | Sci-Fi | Thriller | War | Western

```
1|Toy Story
(1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)
|0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0|0
2|GoldenEye
(1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?GoldenEye%20(1995)|0
|1|1|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
3|Four Rooms
(1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995
)|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
4|Get Shorty
(1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995
)|0|1|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0
```

- u.data 파일 형식
  - user id | item id | rating | timestamp

```
196 242 3   881250949
186 302 3   891717742
22  377 1   878887116
244 51  2   880606923
166 346 1   886397596
298 474 4   884182806
115 265 2   881171488
253 465 5   891628467
305 451 3   886324817
```

- MovieLens 데이터 세트를 로드하는 함수

```
def loadMovieLens(path='C:\Python25\data\movielens'):
  # Get movie titles
  movies={}
  for line in open(path+'\u.item'):
    (id,title)=line.split('|')[0:2]
    movies[id]=title

  # Load data
  prefs={}
  for line in open(path+'\u.data'):
    (user,movieid,rating,ts)=line.split('\t')
    prefs.setdefault(user,{})
    prefs[user][movies[movieid]]=float(rating)
  return prefs
```

- 데이터 세트 로드 후 임의 사용자(사용자 ID 1번)의 평가점수 확인 방법

```
>>> prefs = loadMovieLens()
>>> len(prefs)
943
>>> prefs['1']
{'Birdcage, The (1996)': 4.0, 'Back to the Future (1985)': 5.0, 'Nikita
(La Femme Nikita) (1990)': 5.0, 'Diabolique (1996)': 4.0, 'Welcome to th
e Dollhouse (1995)': 5.0, 'Sting, The (1973)': 4.0, 'French Twist (Gazon
maudit) (1995)': 5.0, 'Star Trek: The Wrath of Khan (1982)': 5.0, 'Gross
e Pointe Blank (1997)': 4.0, 'Kansas City (1996)': 3.0, 'Stargate (1994)
': 3.0, 'Horseman on the Roof, The (Hussard sur le toit, Le) (1995)': 5.
0, 'Return of the Pink Panther, The (1974)': 4.0, "Carlito's Way (1993)"
```

```
>>> getRecommendations(prefs, '1')[0:30]
[(5.0, 'They Made Me a Criminal (1939)'), (5.0, 'Star Kid (1997)'), (5.0
, "Someone Else's America (1995)"), (5.0, 'Saint of Fort Washington, The
(1993)'), (5.0, 'Prefontaine (1997)'), (5.0, 'Marlene Dietrich: Shadow a
nd Light (1996) '), (5.0, 'Little City (1998)'), (5.0, 'Great Day in Har
lem, A (1994)'), (5.0, 'Aiqing wansui (1994)'), (4.999999999999991, 'Sa

>>> itemsim = calculateSimilarItems(prefs, n=50)
100 / 1664
200 / 1664
300 / 1664
400 / 1664

>>> getRecommendedItems(prefs, itemsim, '1')[0:30]
[(5.0, 'Winnie the Pooh and the Blustery Day (1968)'), (5.0, "What's Lov
e Got to Do with It (1993)"), (5.0, 'Up Close and Personal (1996)'), (5.
0, 'Stand by Me (1986)'), (5.0, 'Speed (1994)'), (5.0, "Sophie's Choice
(1982)"), (5.0, 'Something to Talk About (1995)'), (5.0, 'Some Like It H
ot (1959)'), (5.0, 'Some Folks Call It a Sling Blade (1993)'), (5.0, 'Sh
```

- 항목기반 유사도 사전 구축에 많은 시간 걸리지만, 일단 만들고 나면 추천은 즉시 가능

9. 사용자 기반 필터링 vs. 항목기반 필터링

- 사용자 기반 필터링
  - 구현 용이, 추가 단계 없음
  - 자주 변경되는 작은 데이터 세트에 적당
- 항목 기반 필터링
  - 큰 데이터 세트인 경우 훨씬 빠름
  - 항목 유사도 테이블 유지 위한 추가 부담
  - 희박한 데이터 세트인 경우 더 잘 동작
  - 조밀한 데이터 세트인 경우에는 둘 다 비슷