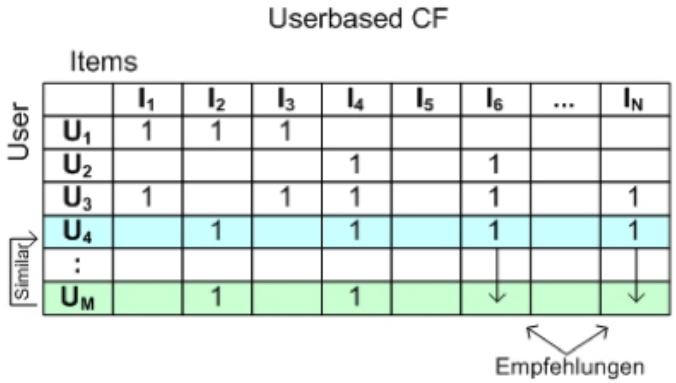
1.4 Vor dem Versuch zu klärende Fragen

Beschreiben Sie das Prinzip des userbasierten Collaborativen Filtering (UCF).

UCF benuzt die Ähnlichkeit zwischen Nutzern. Um für einen Nutzer eine Empfehlung zu machen, wird zuerst der Kunde (oder eine Kundenmenge ermittelt), der dem Nutzer am Ähnlichsten ist. Dann werden dem Nutzer die Produkte gezeigt, welcher der ähnlichste Kunde gekauft hat, der Nutzer selbst aber nicht.



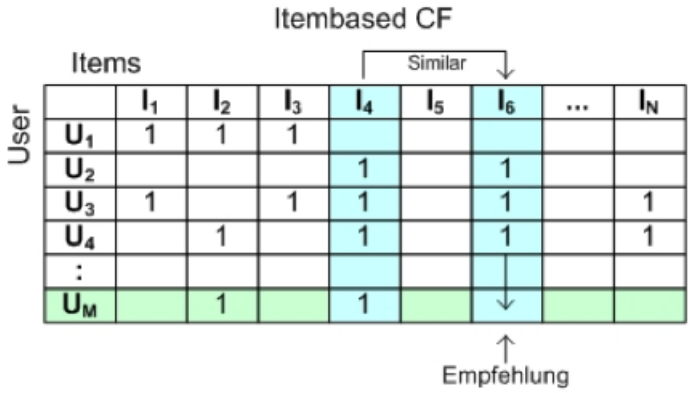
	I_1	I_2	I_3	I_4	I_5	I_6	...	I_N
U_1	1	1	1					
U_2				1		1		
U_3	1		1	1		1		1
U_4		1		1		1		1
:								
U_M		1		1				

Welche Nachteile hat das UCF?

- Skaliert schlecht bei sehr großen User/Item-Matrizen
- Der Ansatz ist unnzuverlässig für Kunden, die erst wenige Produkte gekauft haben.

Worin besteht der Unterschied zwischen UCF und itembasierten Collaborativen Filtering (ICF)?

ICF benutzt die Ähnlichkeit zwischen Produkten. Produkte sind umso ähnlicher, je mehr Kunde diese zusammen gekauft haben. Für die Produkte, welche ein Referenznutzer bereits gekauft hat, werden die ähnlichsten Produkte ermittelt und empfohlen, wenn er diese nicht bereits selbst gekauft hat.



Gegeben seien die Vektoren a = [1,2,3,4,5,6] b = [3,3,5,6,7,8]
Zeigen Sie am Beispiel des Vektors a wie Mittelwert und Varianz eines Vektors berechnet werden.

Mittelwert: (1+2+3+4+5+6)/6 = 3,5

$$\frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Varianz: ⅙*(1^2+2^2+3^2+4^2+5^2+6^2)-3,5 = 2.91666666667

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{m} x_i^2 H_i - \bar{x}^2$$

Wie werden Mittelwert und Varianz mit numpy berechnet?
Mittelwert: `narray.mean()`
Varianz: `narray.var()`

Wie groß ist die: Euklidische Ähnlichkeit, Pearson Ähnlichkeit, Cosinus Ähnlichkeit zwischen den Vektoren a und b?

Euklidische Ähnlichkeit:

$$s_E(\underline{a},\underline{b}) = \frac{1}{1 + d_E(\underline{a},\underline{b})}$$

Zwei Vektoren sind also umso ähnlicher, je geringer die Euklidische Distanz ($d_E = \sqrt{21} \approx 4.58258$) zwischen ihnen ist.

Ergebnis: 0,179

Pearson Ähnlichkeit:

$$\rho_{\underline{a},\underline{b}} = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{(a_i - \bar{a})}{\sigma_a} \frac{(b_i - \bar{b})}{\sigma_b}$$

N die Länge der Vektoren, ***ā*** den Mittelwert und $\boldsymbol{\sigma_a}$ die Standardabweichung des Vektors a. Der Wert liegt dabei zwischen +1 (steigende Gerade) und -1 (fallende Gerade). Ein Wert von 0 bedeutet, dass keine Abhängigkeit zwischen den Vektoren besteht.

Ergebnis: 0.983

Cosinus Ähnlichkeit:

$$\cos(\underline{a},\underline{b}) = \frac{\underline{a} \cdot \underline{b}}{\|\underline{a}\| \cdot \|\underline{b}\|}$$

Im Zähler steht das Skalarprodukt der Vektoren und im Nenner das Produkt der Beträge der Vektoren. Es wird Kosinus des Winkels zwischen beiden Vektoren bestimmt. Also bestimmt, ob sie in die gleiche Richtung zeigen.

Ergebnis: 0.991

In welchen Fällen sind Cosinus- und Pearsonähnlichkeit der euklidischen Ähnlichkeit vorzuziehen?

Wenn die Vektoren in absoluten Zahlen weit auseinander liegen, aber in gleicher Weise variieren. Also beispielsweise eine Zusammenhang zwischen Alter (kleine Zahlen) und Gehalt (große Zahlen) gefunden werden soll. Bei Cosinus-Ähnlichkeit und dem Pearson-Korrelationswert ist das Ergebnis ungebunden (unabhängig von der Skalierung).

Zudem, wenn a und b mittelwertfrei sind, ist die Cosinus-Ähnlichkeit gleich dem Pearson-Korrelationswert.
[http://brenocon.com/blog/2012/03/cosine-similarity-pearson-correlation-and-ols-coefficients/]

Wie wird in Python ein doppelt verschachteltes Dictionary angelegt und wie greift man auf dessen Elemente zu?

```
dict = {"key1":{"key2": 1337}}
print dict["key1"]["key2"]
// Ergebnis: 1337
```

Wie können mit Hilfe der last.fm-Api pylast.py alle Alben einer Band bestimmt werden?

```
# Artist holen
artist = network.get_artist("Backstreet Boys")

# Get albums from artist
topAlbums = artist.get_top_albums()
```

2.2 Ähnlichkeitsbestimmung

1. Welche Bedeutung hat der Übergabeparameter normed in der Funktion sim_euclid?

sim_euclid gibt den Euklidische Ähnlichkeitswert zwischen zwei Personen zurück. Dazu werden alle Filmbewertungen herangezogen. Von einer oder beiden Personen nicht-bewertete Filme bleiben unbeachtet.

Mit dem Übergabeparameter “normed=true” wird folgende Funktion ausgeführt:

```
if normed:
     sum_of_squares= 1.0/len(si)*sum_of_squares
```

Das bedeutet, dass bei genormten Resultate dem entgegen gewirkt wird, dass Personen, die mehr Filme zusammen bewertet haben, sich pauschal ähnlicher sind als Personen, welche nur wenige Filme gemeinsam bewertet haben.

Zudem werden die Ergebnisse auf einen Wert zwischen 0 und 1 genormt.

2. Schreiben Sie eine Funktion topMatches(prefs,person,similarity), welche für eine beliebige in critics (wird an prefs übergeben) enthaltene Person die Ähnlichkeitswerte zu allen anderen Personen berechnet und in einer geordneten Liste zurück gibt. Der Funktion soll als Übergabeparameter auch die anzuwendende Ähnlichkeitsfunktion (sim_euclid oder sim_pearson) übergeben werden können. Berechnen Sie mit dieser Funktion für jede Person die top matches, zunächst unter Verwendung der euklidischen- dann unter Verwendung der Pearson-Ähnlichkeit.

Euclid normed to ‘Toby’:

	Similarity
Mick LaSalle	0.666667
Claudia Puig	0.624639
Lisa Rose	0.615912
Michael Phillips	0.558482
Jack Matthews	0.522774
Gene Seymour	0.510875

Pearson to 'Toby:

	Similarity
Lisa Rose	0.991241
Mick LaSalle	0.924473
Claudia Puig	0.893405
Jack Matthews	0.662849
Gene Seymour	0.381246
Michael Phillips	-1.000000

3. Vergleichen Sie die beiden Ähnlichkeitsmaße. Welches Ähnlichkeitsmaß erscheint Ihnen für diesen Anwendungsfall sinnvoller und warum?

Der euklidische Algorithmus betrachtet die absolute Bewertung der einzelnen Personen. Tendenzen werden dabei außen vor gelassen. Wenn Person A die zwei Filme mit “1” und “3” bewertet und Person B diese mit “3” und “5”, so sind diese bei Euklid nicht sehr ähnlich.

Pearson betrachtet auch diese “Bewertungstendenz”. Die beiden Personen haben die Filme zueinander betrachtet ähnlich bewertet und sind somit ähnlich.

Pearson ist somit sinnvoller, wenn es um solche Bewertungskriterien geht. Manche Nutzer bewerten beispielsweise immer mit weniger Punkten, während andere immer mehr geben, auch wenn die Tendenz bei beiden genau gleich ist.

2.3 Berechnung von Empfehlungen mit User basiertem Collaborative Filtering

Recommendations Euklid normed:

	kSum
The Night Listener	3.427348
Lady in the Water	2.795737
Just My Luck	2.407393

Recommendations Pearson:

	kSum
The Night Listener	3.347790
Lady in the Water	2.832550
Just My Luck	2.530981

2.4 Berechnung von Empfehlungen mit Item basiertem Collaborative Filtering

2.4.1.1. Transformierte Matrix

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

2.4.1.2. calculateSimilarItems(euclidean)

'Lady in the Water':	**Similarity**
You, Me and Dupree	0.765588
The Night Listener	0.759747
Snakes on a Plane	0.727714
Just My Luck	0.615912
Superman Returns	0.612574

'Snakes on a Plane':	**Similarity**
Superman Returns	0.757898
The Night Listener	0.738796
Lady in the Water	0.727714
You, Me and Dupree	0.582459
Just My Luck	0.578413

'Just My Luck':	**Similarity**
You, Me and Dupree	0.653454
The Night Listener	0.630397
Lady in the Water	0.615912
Snakes on a Plane	0.578413
Superman Returns	0.512303

'Superman Returns':	**Similarity**
Snakes on a Plane	0.757898
The Night Listener	0.669789
Lady in the Water	0.612574
You, Me and Dupree	0.587482
Just My Luck	0.512303

'You, Me and Dupree':	**Similarity**
Lady in the Water	0.765588
The Night Listener	0.675866
Just My Luck	0.653454
Superman Returns	0.587482
Snakes on a Plane	0.582459

```
'The Night Listener': Similarity
Lady in the Water     0.759747
Snakes on a Plane     0.738796
You, Me and Dupree    0.675866
Superman Returns      0.669789
Just My Luck          0.630397}
```

2.4.1.3. getRecommendedItems

getRecommendedItems(EUCLID)

```
                      Normalized
The Night Listener    3.204491
Lady in the Water     3.082137
Just My Luck          3.041862
```

getRecommendedItems(PEARSON)

```
                      Normalized
Lady in the Water     3.610031
The Night Listener    0.000000
Just My Luck          0.000000
```

3. last.fm Musikempfehlungen

Komischerweise wird die ausgesuchte Band ebenfalls als „Toptreffer“ zurückgegeben.

(The Rolling Stones 0.555629)
Pink Floyd 0.333302
Fleetwood Mac 0.333302
U2 0.333208
Van Morrison 0.222233
The Pretenders 0.222139
The Police 0.222139
Procol Harum 0.111163
Regina Spektor 0.111163
Nat King Cole 0.111163
Rasputina 0.111163
Faith No More 0.111163
Elvis Presley 0.111163
Elmore James 0.111163
Patti Smith 0.111163
…