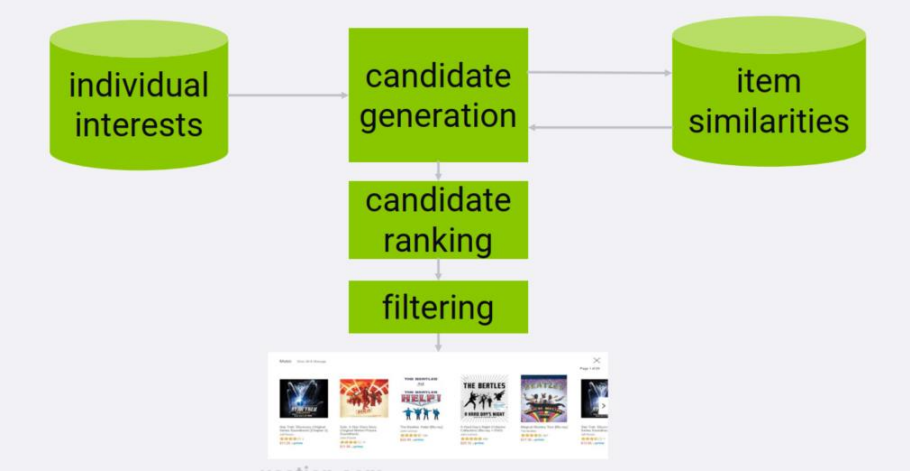
Системи за препоруку засновани на сарадњи (Collaborative filtering)

Сарадничко филтрирање је заправо идеја искоришћавања понашања других како бисмо информисали себе у чему бисмо могли да уживамо и шта би то нама било интересантно. Другим речима, то значи да треба пронаћи друге људе који су попут вас и препоручити вам ствари које су се њима свиделе (очекујући да ће се и вама свидети). Или то може да значи проналажење других ствари сличних стварима које волите. Односно, препоручивање ствари које су купили људи, а који су такође купили ствари које су вам се свиделе. У сваком случају, идеја је узимање знакова од људи попут вас, људи из вашег окружења, и препоручивање ствари заснованих на стварима које воле, а које још нисте видели.

Дакле, прво се генеришу ствари које потенцијално можемо да препоручимо кориснику, а на основу сличних ствари стварима које корисник воли (за сваког корисника посебно). Затим се ти „кандидати“ на неки начин оцењују и рангирају. Након тога се кандидати додатно филтрирају (нпр. не приказујемо производе које је корисник већ купио) и кориснику се приказује топ-Н листа препорука.



За разлику од *система за препоруку на основу садржаја,* у којима се користе атрибути/описи производа, овде се користи понашање корисника.

Највећи проблем приступа који се заснива на мерењу сличности између понашања корисника је ретка матрица (sparse matrix).

	Бекство из Шошенка	Казабланка	Ратови звезда	Одбегла млада	Титаник
Бранко	4				
Милош			5		
Ана					5
Петар					

Када се ради са ретким матрицама, треба водити рачуна о меморији.
Један од начина да се предупреди овај проблем је коришћење **Sparse matrix**:

```
from numpy import array
from scipy.sparse import csr_matrix
# create dense matrix
A = array([[1, 0, 0, 1, 0, 0], [0, 0, 2, 0, 0, 1], [0, 0, 0, 2, 0, 0]])
print(A)
# convert to sparse matrix (CSR method)
S = csr_matrix(A)
print(S)
```

```
[[1 0 0 1 0 0]
 [0 0 2 0 0 1]
 [0 0 0 2 0 0]]

  (0, 0)  1
  (0, 3)  1
  (1, 2)  2
  (1, 5)  1
  (2, 3)  2
```

Неке друге мере сличности између два вектора

1. Cosine similarity

$$CosSim(x,y) = \frac{\sum_i (x_i y_i)}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

Сличност се гледа само за производе који су оцењени од стране оба корисника!

	Бекство из Шошенка	Казабланка	Ратови звезда	Одбегла млада	Титаник
Бранко	4	5			
Милош		2	5	2	2

CosSim(Бранко, Милош)=1 (у којем случају ово важи?)
CosSim(Бранко, Милош)=(5*2)/(sqrt(4^2+5^2)*sqrt(3 * 2^2+5^2))

Ретке матрице стварају велике проблеме! Често је потребно да се унапред дефинише колико корисници морају да имају заједничких филмова које су оценили.

2. **Adjusted cosine similarity (user-based Pearson similarity):**

$$CosSim(x,y) = \frac{\sum_i((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_i(x_i - \bar{x})^2}\sqrt{\sum_i(y_i - \bar{y})^2}}$$

Различити начини оцењивања, другачији референтни систем имају људи. Друге културе, поднебље итд. $\bar{x}$ је просечна вредност свих рејтинга које је тај корисник дао. Разлика у односу на стандардни начин рачунања *CosSim* вредност је што овде посматрамо и варијансу у односу на просек свих корисникових оцена, а не само дате вредности.

Проблем: Шта значи просек у ретким матрицама? Све зависи какви су подаци са којима се ради... Шта ако је корисник дао само једну оцену?
Што више људи, то је боље, али у пракси се показало да се онда не добија превише на тачности када се пореде ове две верзије косинусне сличности.

3. **Пирсонова сличност (item-based Pearson similarity):**

$$CosSim(x,y) = \frac{\sum_i((x_i - \bar{\imath})(y_i - \bar{\imath}))}{\sqrt{\sum_i(x_i - \bar{\imath})^2}\sqrt{\sum_i(y_i - \bar{\imath})^2}}$$

Једина разлика у односу на претходну метрику је што се уместо разлике између рејтинга и просечног рејтинга корисника користи разлика између рејтинга и просечне вредности рејтинга свих корисника за дати производ i.
Делује да је ово доста бољи приступ за реалне податке и продукцију. Зашто?

Пирсонова сличност се може интерпретирати као мера сличности људи према томе колико се разилазе/одвајају од просечног човековог понашања. Замислите филм који већина људи воли, попут *Ратова звезда*, људи који не воле *Ратове звезда* имаће врло јаку оцену сличности према Пеарсоновој сличности, јер деле мишљења која нису мејнстрим.

4. **Spearman similarity**

Уместо рејтинга користи појам рангирања. Лоше ради у пракси.

5. **Mean square difference (MSD)**

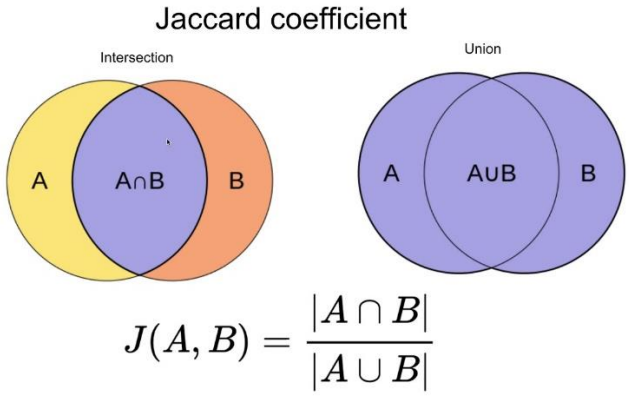
$$MSD(x,y) = \frac{\sum_{i \in I_{xy}} (x_i - y_i)^2}{|I_{xy}|}$$

$$MSDsim(x,y) = \frac{1}{MSD(x,y) + 1}$$

На исти начин може да се дефинише и за производе (items). *У пракси се показало да косинусна сличност боље ради.*

6. **Jaccard similarity**

Jaccard сличност између два корисника се рачуна као количник два броја: пресека између између рејтинга два корисника, и уније свих рејтинга за та два корисника. Када треба користити ову меру?



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

САРАДНИЧКО ФИЛТРИРАЊЕ БАЗИРАНО НА КОРИСНИКУ (USER-BASED COLLABORATIVE FILTERING)

Циљ је пронаћи друге кориснике сличне себи, на основу њихове историје оцена, а затим препоручити оне ствари које су се њима свиделе, а ви их још увек нисте видели.

Кораци у имплементацији:

1. Одредити векторе за сваког корисника. **Како?**
2. Одреди матрицу сличности између корисника користећи неку од поменутих метрика.
3. Узети k (у коду ћемо користити ову ознаку) најсличнијих корисника за посматраног корисника и генерисати кандидате за препоруку.
4. Доделити скор сваком филму. Постоје различити начини за доделу скора.

Чини се разумним узети оцену која је додељена сваком кандидату (филму) као део његовог скора. Ми хоћемо кориснику да прикажемо шта други корисници воле, а не шта не воле. Рејтинге би требало нормализовати, јер су део производа (нпр. оцена 5 постаје 1.0). Или пак можемо поделити филмове на позитивне и негативне...

Вероватно би требало узети у обзир и сличност са корисником који је генерисао ове оцене. Погледати имплементацију коју смо урадили.

Размислити шта би још све могло да дође у обзир!

На пример: могло би да се деси да се у кандидатима појави исти филм више пута, ако га је оценило више сличних корисника. Вероватно би требало и то узети у обзир.

5. Сортирати их у опадајућем редоследу
6. Филтрирати кандидате

- Сарадничко филтрирање базирано на кориснику/производу предвиђа _____?
- Сарадничко филтрирање базирано на **кориснику** **VS** Сарадничко филтрирање базирано на **производу**? (покушајте сами да имплементирате за производе на основу кода за кориснике да бисте схватили које су разлике).

Python имплентација за сарадничко филтрирање базирано на кориснику (UB CF)

Python имплентација за сарадничко филтрирање базирано на производу (IB CF)

Предикције за UB CF:

```
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Finding Nemo (2003) 4.0
Terminator 3: Rise of the Machines (2003) 3.4
Bruce Almighty (2003) 3.2
The Lord of the Rings: The Two Towers (2002) 3.0
The Matrix Reloaded (2003) 3.0
The Hulk (2003) 3.0
X2: X-Men United (2003) 3.0
American Wedding (2003) 3.0
The Lord of the Rings: The Fellowship of the Ring (2001) 2.0
S.W.A.T. (2003) 2.0
Seabiscuit (2003) 2.0
```

Предикције за IB CF:

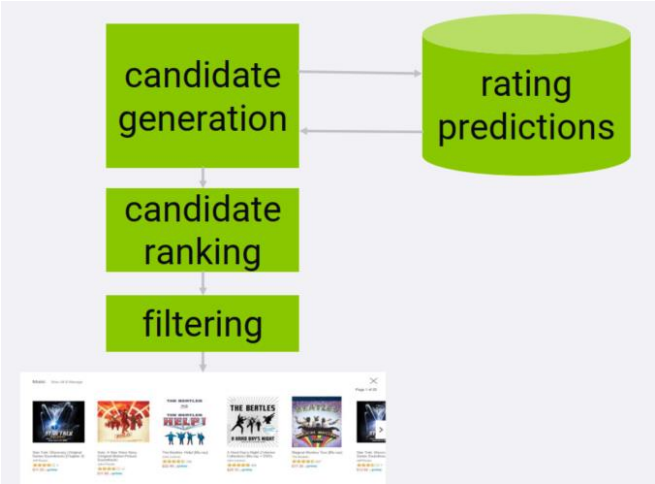
```
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Rio Bravo (1959) 9.979822846313532
Auto Focus (2002) 9.97439537810189
Foul Play (1978) 9.952479569545577
How the West Was Won (1962) 9.943590812665304
The Pride of the Yankees (1942) 9.942567032724133
Casablanca (1942) 9.938554951567449
The Ten Commandments (1956) 9.932095431650938
Lassie (1994) 9.92951867325116
Father of the Bride (1992) 9.911855457372521
Boiler Room (2000) 9.908471397031333
Dead Man Walking (1995) 9.906860604478506
```

Домаћи:

- Евалуација алгоритма помоћу *HitRate* метрике.
- Пронаћи два-три корисника који су вама блиски. Штампати топ-Н резултате користећи неке варијације, и пратите резултате.

KNN системи за препоруку

Како предвидети рејтинг који би корисник дао производу, а да се и даље алгоритам заснива на концептима које користи **сарадничко филтрирање**.



Кориснички базирани KNN

Подсетити се алгоритма који је представљен на другом термину.

И даље у основи користи матрица сличности између **корисника** и **корисника**, али се не препоручују само ствари које су се свиделе људима сличним посматраном кориснику. Уместо тога, посеже се дубље у податке да би се направило предвиђање рејтинга за сваку могућу ставку за сваког могућег корисника. То је сложенији приступ, и обично је то лоше.

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} (sim(u,v) \cdot r_{vi})}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

$\hat{r}_{ui}$ - предикција рејтинга корисника u за производ i

$N_i^k(u)$ - скуп k најближих корисника кориснику u који су оценили производ i

$sim(u, v)$ - функција сличности која рачуна степен сличности између корисника u и v (нпр. *Cosine Similarity*).
r_{vi} - рејтинг којим је корисник v оценио производ i

KNN заснован на производима

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} (sim(i,j) \cdot r_{ui})}{\sum_{j \in N_u^k(j)} sim(i,j)}$$

Додатни коментари: Коришћење сарадничког филтрирања није проблем, али проблем настаје када се сарадничко филтрирање форсира да би се направило предвиђања рејтинга.
Срж проблема: Оцене нису континуалне природе, а KNN их третира као да су континуалне вредности, тј. као да се могу предвидети на континуалној скали.

Домаћи за кориснички KNN:

- Тестирати различите метрике за одређивање сличности између вектора.
- KNNWithMeans
- KNNWithZScore