

Filtros Colaborativos

Rómulo Condori
Kevin Valverde Huilca
Ruben Huanca Morales
José Torres Lima
Diego Bellido Ramos
Luis Mamani Chirinos

*Escuela Ciencia de la Computación
Universidad Nacional de San Agustín

22 de Abril del 2019



Contenido de la exposición I

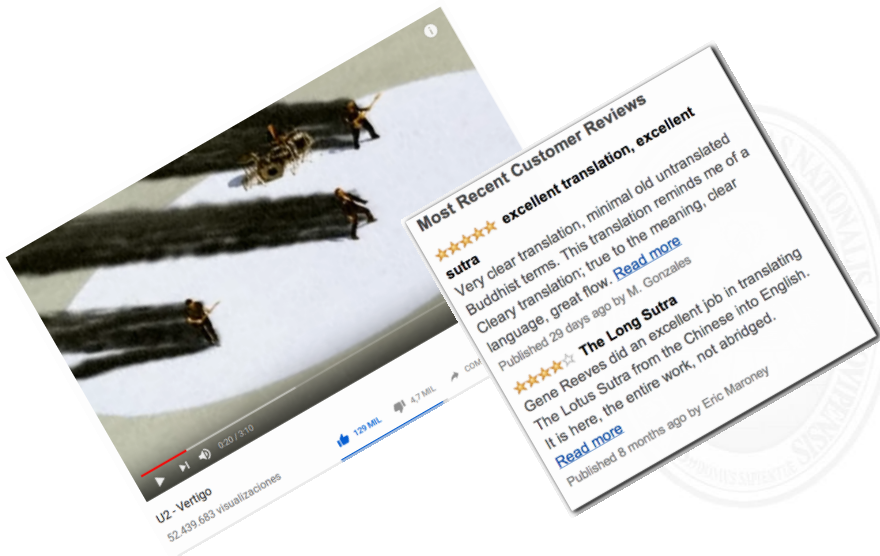


Los algoritmos descritos en el capítulo 2 son de propósito general y podrían usarse con una variedad de datos. Los usuarios calificaron diferentes artículos en una escala de cinco o diez puntos y los algoritmos encontraron otros usuarios que tenían calificaciones similares. Como se mencionó, existe evidencia que sugiere que los usuarios normalmente no usan esta distinción de bien detallada y en cambio tienden a otorgar la calificación más alta o la más baja. En este capítulo, examinaremos formas de afinar el filtrado colaborativo para producir recomendaciones más precisas de una manera eficiente.

Calificaciones Explícitas



Calificaciones Explícitas



Piensa en otros videos de YouTube que has visto esta semana y compáralos con este. ¿Qué opinas de él?

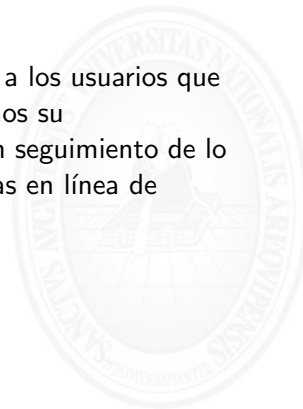


Acoustic Rock Songs | Top 20 Greatest Rock Songs On Spotify

- ☐ Es uno de los peores videos que he visto
- ☐ No es gran cosa
- ☐ Es del montón
- ☐ Es muy bueno
- ☐ Es uno de los mejores videos que he visto
- ☐ No lo he visto o no me acuerdo del video





ENVIAR

Para las calificaciones implícitas, no les pedimos a los usuarios que proporcionen ninguna calificación, solo observamos su comportamiento. Un ejemplo de esto es hacer un seguimiento de lo que un usuario hace clic en el sistema de compras en línea de Amazon.



Considere qué información podemos obtener de la recopilación de los productos en los que un usuario hace clic en Amazon. En su página de inicio personalizada de Amazon se mostraría esta información:

More Items to Consider

You viewed	Customers who viewed this also viewed		
 <p>Jupiters Travels: Four Years Around... › Ted Simon Paperback ★★★★☆ (65) \$24.95 \$16.47</p>	 <p>Long Way Round Ewan McGregor, Charley Boorman, ... DVD ★★★★☆ (325) \$24.95 \$16.93</p>	 <p>One More Day Everywhere: Crossing 50... › Glen Heggstad Paperback ★★★★☆ (47) \$18.95 \$13.87</p>	 <p>Dreaming of Jupiter › Ted Simon Paperback ★★★★☆ (6) \$16.22</p>

Calificaciones implícitas

Otra calificación implícita es lo que el cliente realmente compra. Amazon también realiza un seguimiento de esta información y la utiliza para sus recomendaciones “Comprados con frecuencia juntos” y “Los clientes que vieron este artículo también compraron”.



Calificaciones implícitas

Otra calificación implícita es lo que el cliente realmente compra. Amazon también realiza un seguimiento de esta información y la utiliza para sus recomendaciones “Comprados con frecuencia juntos” y “Los clientes que vieron este artículo también compraron”.

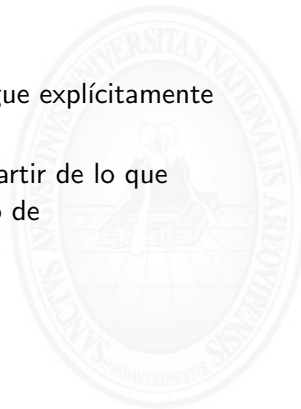


Imagina qué información puede adquirir un programa al monitorear tu comportamiento en spotify.

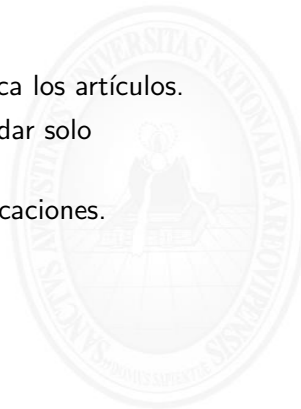
Name	Time	Artist	Plays
Anchor	3:24	Zee Avi	52
My Companjera	3:22	Gogol Bordello	27
Wake Up Everybody	4:25	John Legend & the...	17
Milestone Moon	3:40	Zee Avi	17
...			

Entonces:

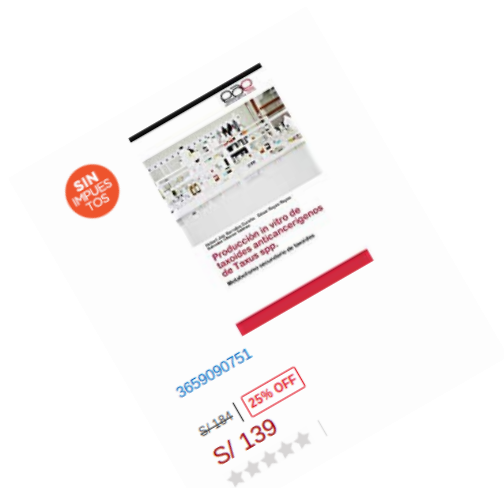
- Será más preciso tener un usuario que otorgue explícitamente la calificación de un producto?
- O será más preciso obtener información a partir de lo que compra o hace un usuario (ejemplo: número de reproducciones)?



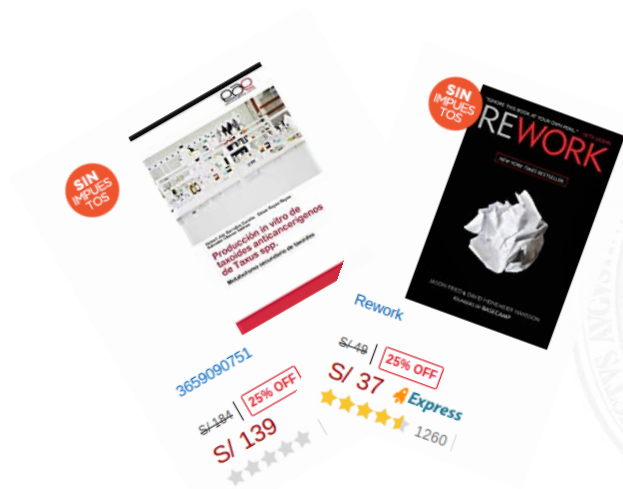
- Problema 1: la gente es perezosa y no califica los artículos.
- Problema 2: Las personas pueden mentir o dar solo información parcial.
- Problema 3: La gente no actualiza sus calificaciones.



Problemas con calificaciones implícitas



Problemas con calificaciones implícitas



Problemas con calificaciones implícitas



Problemas con calificaciones implícitas



Problemas con calificaciones implícitas

SIN
IMPUESTOS



Melissa & Doug Giant Golden...

~~S/ 277~~

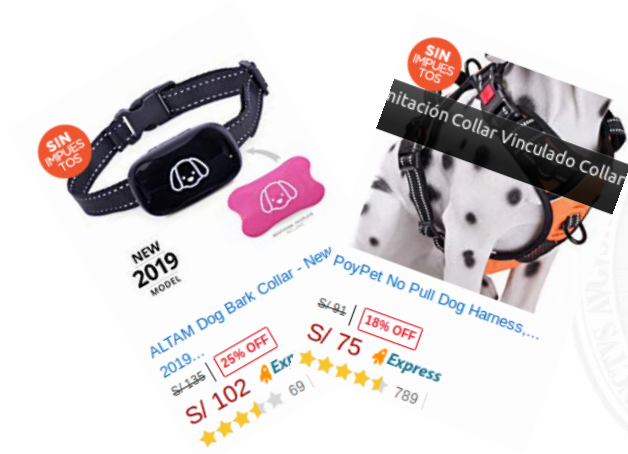
44% OFF

S/ 156  Express

★★★★★ 172 |



Problemas con calificaciones implícitas



Problemas con calificaciones implícitas



- ¿Qué podemos usar como datos implícitos cuando observamos el comportamiento de una persona en una computadora?
 - Páginas web
 - Clicks en el enlace
 - Tiempo empleado en la web
 - Visitas repetidas
 - Recomendación de la página a otros.
 - Reproductores de música
 - Qué reproduce una persona
 - Canciones saltadas
 - Número de veces que reproduce una canción



Los problemas del éxito

Tiene un servicio de transmisión de música exitoso con un sistema de recomendación incorporado. ¿Qué podría salir mal?

Supongamos que tienes un millón de usuarios. Cada vez que desee hacer una recomendación para alguien, debe calcular un millón de distancias (comparando a esa persona con las otras 999.999 personas). Si hacemos varias recomendaciones por segundo, la cantidad de cálculos se vuelve extrema. A menos que desgaste un monto de servidores en el problema, el sistema se ralentizará. Para decir esto de una manera más formal, la latencia puede ser un gran inconveniente de los sistemas de recomendación basados en vecinos. Afortunadamente, hay una solución.

Hasta ahora hemos estado haciendo filtrado colaborativo basado en el usuario. Estamos comparando un usuario con cualquier otro usuario para encontrar las coincidencias más cercanas. Hay dos problemas principales con este enfoque:

- Escalabilidad
- Escasez de datos

Debido a estos dos problemas, podría ser mejor hacer lo que se denomina filtrado basado en elementos. El filtrado basado en el usuario también se denomina filtrado colaborativo basado en la memoria. ¿Por qué? Porque necesitamos almacenar todas las clasificaciones para poder hacer recomendaciones.

Supongamos que tengo un algoritmo que identifica los productos que son más similares entre sí. El filtrado basado en elementos también se denomina filtrado colaborativo basado en modelos. ¿Por qué? Porque no necesitamos almacenar todas las clasificaciones. Construimos un modelo que representa qué tan cerca está cada elemento de cualquier otro elemento.

Filtro basado en productos

Ejemplo: Supongamos que nuestro sitio de transmisión de música tiene m usuarios y n bandas, donde los usuarios califican bandas.

Esto se muestra en la siguiente tabla. Como antes, las filas representan a los usuarios y las columnas representan bandas.

	Users	...	Phoenix	...	Passion Pit	...	n
1	Tamera Young		5				
2	Jasmine Abbey				4		
3	Arturo Alvarez		1		2		
...	...						
u	Cecilia De La Cueva		5		5		
...	...						
m-1	Jessica Nguyen		4		5		
m	Jordyn Zamora		4				

Similitud del coseno ajustada

$$s(i,j) = \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_u)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \bar{R}_u)^2}}$$



Similitud del coseno ajustada

Para ilustrar la similitud de coseno ajustada usaremos los siguientes datos donde cinco estudiantes calificaron a cinco artistas musicales.

Users	average rating	Kacey Musgraves	Imagine Dragons	Daft Punk	Lorde	Fall Out Boy
David			3	5	4	1
Matt			3	4	4	1
Ben		4	3		3	1
Chris		4	4	4	3	1
Torri		5	4	5		3

Similitud del coseno ajustada

Para ilustrar la similitud de coseno ajustada usaremos los siguientes datos donde cinco estudiantes calificaron a cinco artistas musicales.

Users	average rating	Kacey Musgraves	Imagine Dragons	Daft Punk	Lorde	Fall Out Boy
David	3.25		3	5	4	1
Matt	3.0		3	4	4	1
Ben	2.75	4	3		3	1
Chris	3.2	4	4	4	3	1
Tori	4.25	5	4	5		3

Similitud del coseno ajustada

$$s(Musgraves, Dragons) = \frac{\sum_{u \in U} (R_{u, Musgraves} - \bar{R}_u)(R_{u, Dragons} - \bar{R}_u)}{\sqrt{\sum_{u \in U} (R_{u, Musgraves} - \bar{R}_u)^2} \sqrt{\sum_{u \in U} (R_{u, Dragons} - \bar{R}_u)^2}}$$

Ben's
ratings

Chris's
ratings

Tori's
ratings

$$\begin{aligned} &= \frac{(4 - 2.75)(3 - 2.75) + (4 - 3.2)(4 - 3.2) + (5 - 4.25)(4 - 4.25)}{\sqrt{(4 - 2.75)^2 + (4 - 3.2)^2 + (5 - 4.25)^2} \sqrt{(3 - 2.75)^2 + (4 - 3.2)^2 + (4 - 4.25)^2}} \\ &= \frac{0.7650}{\sqrt{2.765} \sqrt{0.765}} = \frac{0.7650}{(1.6628)(0.8746)} = \frac{0.7650}{1.4543} = 0.5260 \end{aligned}$$

Similitud de coseno de las userRatings

	Fall Out Boy	Lorde	Daft Punk	Imagine Dragons
Kacey Musgraves	-0.9549	0.3210	1.0000	0.5260
Imagine Dragons	-0.3378	-0.2525	0.0075	
Daft Punk	-0.9570	0.7841		
Lorde	-0.6934			

Matriz de similitud

Similitud de coseno - implementación

```
def computeSimilarity(band1, band2, userRatings):
    averages = {}
    for (key, ratings) in userRatings.items():
        averages[key] = (float(sum(ratings.values()))
                        / len(ratings.values()))

    num = 0 # numerator
    dem1 = 0 # first half of denominator
    dem2 = 0
    for (user, ratings) in userRatings.items():
        if band1 in ratings and band2 in ratings:
            avg = averages[user]
            num += (ratings[band1] - avg) * (ratings[band2] - avg)
            dem1 += (ratings[band1] - avg)**2
            dem2 += (ratings[band2] - avg)**2
    return num / (sqrt(dem1) * sqrt(dem2))
```

Similitud de coseno - formato de las userRatings

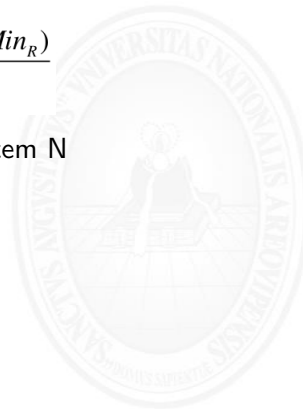
```
users3 = {"David": {"Imagine Dragons": 3, "Daft Punk": 5,  
                  "Lorde": 4, "Fall Out Boy": 1},  
          "Matt": {"Imagine Dragons": 3, "Daft Punk": 4,  
                  "Lorde": 4, "Fall Out Boy": 1},  
          "Ben": {"Kacey Musgraves": 4, "Imagine Dragons": 3,  
                 "Lorde": 3, "Fall Out Boy": 1},  
          "Chris": {"Kacey Musgraves": 4, "Imagine Dragons": 4,  
                   "Daft Punk": 4, "Lorde": 3, "Fall Out Boy": 1},  
          "Tori": {"Kacey Musgraves": 5, "Imagine Dragons": 4,  
                  "Daft Punk": 5, "Fall Out Boy": 3}}
```


$$p(u,i) = \frac{\sum_{N \in \text{similarTo}(i)} (S_{i,N} \times R_{u,N})}{\sum_{N \in \text{similarTo}(i)} |S_{i,N}|}$$

- $p(u, i)$, significa que predecimos la calificación del usuario u dado el item i
- $S_{i,N}$, matriz de similitud
- $R_{u,N}$, Calificación del usuario u dado el item N

$$NR_{u,N} = \frac{2(R_{u,N} - Min_R) - (Max_R - Min_R)}{(Max_R - Min_R)}$$

- $R_{u,N}$ Es la calificación del usuario dado el item N
- $NR_{u,N}$ Es la calificación normalizada
- min_R valor mínimo de la clasificación
- max_R valor máximo de la clasificación



Similitud de coseno - calificación normalizada

Artist	R	NR
Imagine Dragons	3	0
Daft Punk	5	1
Lorde	4	0.5
Fall Out Boy	1	-1

Ratings ya normalizados



David's Ratings

Artist	R	NR
Imagine Dragons	3	0
Daft Punk	5	1
Lorde	4	0.5
Fall Out Boy	1	-1

Similarity Matrix

	Fall Out Boy	Lorde	Daft Punk	Imagine Dragons
Kacey Musgraves	-0.9549	0.3210	1.0000	0.5260
Imagine Dragons	-0.3378	-0.2525	0.0075	
Daft Punk	-0.9570	0.7841		
Lorde	-0.6934			

imagenes/3_2.png

$$= \frac{0 + 1 + 0.1605 + 0.955}{2.802} = \frac{2.1105}{2.802} = 0.753$$

So we predict that David will rate Kacey Musgraves a 0.753 on a scale of -1 to 1. To get back to our scale of 1 to 5 we need to denormalize:

$$R_{u,N} = \frac{1}{2}((NR_{u,N} + 1) \times (Max_R - Min_R)) + Min_R$$
$$= \frac{1}{2}((0.753 + 1) \times 4) + 1 = \frac{1}{2}(7.012) + 1 = 3.506 + 1 = 4.506$$

Slope One

	PSY	Whitney Houston
Amy	3	4
Ben	4	?

Part 1: Calculando la desviación

	Taylor Swift	PSY	Whitney Houston
Amy	4	3	4
Ben	5	2	?
Clara	?	3.5	4
Daisy	5	?	3

The first step is to compute the deviations. The average deviation of an item i with respect to item j is:

$$dev_{i,j} = \sum_{u \in S_{i,j}(X)} \frac{u_i - u_j}{card(S_{i,j}(X))}$$

Calculando la desviación

$$dev_{swift,psy} = \frac{(4-3)}{2} + \frac{(5-2)}{2} = \frac{1}{2} + \frac{3}{2} = 2$$

So the deviation from PSY to Taylor Swift is 2 meaning that on average users rated Taylor Swift 2 better than PSY. What is the deviation from Taylor Swift to PSY?

$$dev_{psy,swift} = \frac{(3-4)}{2} + \frac{(2-5)}{2} = -\frac{1}{2} + -\frac{3}{2} = -2$$

Completando la tabla de valores de desviación

Compute the rest of the values in this table:

Taylor Swift with respect to Whitney Houston:

$$dev_{swift,houston} = \frac{(4-4)}{2} + \frac{(5-3)}{2} = \frac{0}{2} + \frac{2}{2} = 1$$

PSY with respect to Whitney Houston:

$$dev_{psy,houston} = \frac{(3-4)}{2} + \frac{(3.5-4)}{2} = \frac{-1}{2} + \frac{-.5}{2} = -.75$$

	Taylor Swift	PSY	Whitney Houston
Taylor Swift	0	2	1
PSY	-2	0	-0.75
Whitney Houston	-1	0.75	0

Parte 2: Haciendo predicciones con la Slope One Ponderada

Usando la colección de desviaciones calculadas. Para hacer las predicciones podemos usar la Slope One Ponderada o P^{ws1} para la predicción. La fórmula:

$$P^{ws1}(u)_j = \frac{\sum_{i \in S(u) - \{j\}} (dev_{ji} + u_i) c_{ji}}{\sum_{i \in S(u) - \{j\}} c_{ji}}$$

Donde:

$$c_{ji} = card(S_{j,i}(\chi))$$

$P^{ws1}(u)_j$ Significa nuestra predicción utilizando la Slope One Ponderada de las calificaciones del usuario u para el elemento j .

Parte 2: Haciendo predicciones con la Slope One Ponderada

- Entonces, por ejemplo, $P^{wS1}(Ben)_{Whitney\ Houston}$ significa nuestra predicción de lo que Ben calificaría a Whitney Houston.
- Analizando el numerador

$$\sum_{i \in S(u) - \{j\}}$$

- Significa para cada músico que Ben ha calificado (a excepción de Whitney Houston que es la parte j).
El numerador completo significa para cada músico i que Ben ha calificado (a excepción de Whitney Houston) buscaremos la desviación de Whitney Houston a ese músico y lo agregaremos a la calificación de Ben para el músico i . Multiplicamos eso por la cardinalidad de esa pareja, la cantidad de personas que calificaron a ambos músicos (Whitney y el músico i).

Parte 2: Haciendo predicciones con la Slope One Ponderada

Los pasos para realizar esto:

- Primero, aquí están las calificaciones de Ben y nuestra tabla de desviaciones de antes:

	Taylor Swift	PSY	Whitney Houston
Ben	5	2	?

	Taylor Swift	PSY	Whitney Houston
Taylor Swift	0	2	1
PSY	-2	0	-0.75
Whitney Houston	-1	0.75	0

- 1 Ben ha calificado a Taylor Swift y le dio un 5, esa es la u_i .

Parte 2: Haciendo predicciones con la Slope One Ponderada

	Taylor Swift	PSY	Whitney Houston
Ben	5	2	?

	Taylor Swift	PSY	Whitney Houston
Taylor Swift	0	2	1
PSY	-2	0	-0.75
Whitney Houston	-1	0.75	0

- La desviación de Whitney Houston con respecto a Taylor Swift es -1: esta es la $dev_{j,i}$
- $dev_{j,i} + u_i$
- Anteriormente se tenía que a dos personas (Amy y Daisy) calificaron tanto a Taylor Swift como a Whitney Houston $c_{j,i} = 2$

Parte 2: Haciendo predicciones con la Slope One Ponderada

	Taylor Swift	PSY	Whitney Houston
Ben	5	2	?

	Taylor Swift	PSY	Whitney Houston
Taylor Swift	0	2	1
PSY	-2	0	-0.75
Whitney Houston	-1	0.75	0

- 5 Por lo que $(dev_{j,i} + u_i)c_{j,i} = 4 \times 2 = 8$
- 6 Ben ha calificado PSY y le dio un 2
- 7 La desviación de Whitney Houston con respecto a PSY es de 0.75.

Parte 2: Haciendo predicciones con la Slope One Ponderada

	Taylor Swift	PSY	Whitney Houston
Ben	5	2	?

	Taylor Swift	PSY	Whitney Houston
Taylor Swift	0	2	1
PSY	-2	0	-0.75
Whitney Houston	-1	0.75	0

- 8 Entonces $dev_{j,i} + u_i = 2,75$
- 9 Dos personas calificaron tanto a Whitney Houston como a PSY, entonces $(dev_{j,i} + u_i)c_{j,i} = 2,75 \times 2 = 5,5$
- 10 Resumimos los pasos 5 y 9 para obtener 13.5 para el numerador

Parte 2: Haciendo predicciones con la Slope One Ponderada

Denominador

- 11 Al analizar el denominador, obtenemos algo así como para cada músico que Ben ha calificado, sumando las cardinalidades de esos músicos (cuántas personas calificaron tanto a ese músico como a Whitney Houston). Así que Ben ha calificado a Taylor Swift y la cardinalidad de Taylor Swift y Whitney Houston (es decir, el número total de personas que los calificaron a ambos) es 2. Ben ha calificado a PSY y su cardinalidad también es 2. Por lo tanto, el denominador es 4.
- 12 Así que nuestra predicción de qué tan bien le gustará a Whitney Houston es $13,5/4 = 3,375$

Poniendo esto en Python

Vamos a extender la clase de Python desarrollada en el capítulo 2. Recordando que los datos para esa clase estaban en el siguiente formato:

```
users2 = {"Amy": {"Taylor Swift": 4, "PSY": 3, "Whitney Houston": 4},  
          "Ben": {"Taylor Swift": 5, "PSY": 2},  
          "Clara": {"PSY": 3.5, "Whitney Houston": 4},  
          "Daisy": {"Taylor Swift": 5, "Whitney Houston": 3}}
```

Calculando la Desviación

$$dev_{i,j} = \sum_{u \in S_{i,j}(X)} \frac{u_i - u_j}{card(S_{i,j}(X))}$$

La salida debe ser una representación de los siguientes datos:

	Taylor Swift	PSY	Whitney Houston
Taylor Swift	0	2 (2)	1 (2)
PSY	-2 (2)	0	-0.75 (2)
Whitney Houston	-1 (2)	0.75 (2)	0

Figura: Frecuencia y Desviación

Construyendo el método paso a paso

Step 1:

```
def computeDeviations(self):  
    # for each person in the data:  
    #     get their ratings  
    for ratings in self.data.values():
```

Los diccionarios de Python(también conocidos como tablas hash) son pares de clave y valor. Extraeremos solo los valores del diccionario

```
jos@manu:~/Descargas/TBD$ python recommender3.py  
{ 'PSY': 3, 'Taylor Swift': 4, 'Whitney Houston': 4}  
{ 'PSY': 3.5, 'Whitney Houston': 4}  
{ 'PSY': 2, 'Taylor Swift': 5}  
{ 'Taylor Swift': 5, 'Whitney Houston': 3}
```

Construyendo el método paso a paso

Step 2

```
def computeDeviations(self):  
    # for each person in the data:  
    #     get their ratings  
    for ratings in self.data.values():  
        #for each item & rating in that set of ratings:  
        for (item, rating) in ratings.items():  
            self.frequencies.setdefault(item, {})  
            self.deviations.setdefault(item, {})
```

En el método init de la clase recommender, inicializa las frecuencias y las desviaciones para que sean diccionarios.

```
def __init__(self, data, k=1, metric='pearson', n=5):  
    ...  
  
    #  
    # The following two variables are used for Slope One  
    #  
    self.frequencies = {}  
    self.deviations = {}
```

Construyendo el método paso a paso

```
josh@nanu:~/Descargas/IBD$ python recommender3.py
***** Frecuencia *****
{'PSY': {}}
***** Desviación *****
{'PSY': {}}

***** Frecuencia *****
{'PSY': {'Taylor Swift': 1, 'Whitney Houston': 1}, 'Taylor Swift': {}}
***** Desviación *****
{'PSY': {'Taylor Swift': -1.0, 'Whitney Houston': -1.0}, 'Taylor Swift': {}}

***** Frecuencia *****
{'PSY': {'Taylor Swift': 1, 'Whitney Houston': 1}, 'Taylor Swift': {'PSY': 1, 'Whitney Houston': 1, 'Whitney Houston': {}}
***** Desviación *****
{'PSY': {'Taylor Swift': -1.0, 'Whitney Houston': -1.0}, 'Taylor Swift': {'PSY': 1.0, 'Whitney Houston': 0.0}, 'Whitney Houston': {}}

***** Frecuencia *****
{'PSY': {'Taylor Swift': 1, 'Whitney Houston': 1}, 'Taylor Swift': {'PSY': 1, 'Whitney Houston': 1, 'Whitney Houston': {'PSY': 1, 'Taylor Swift': 1}}
***** Desviación *****
{'PSY': {'Taylor Swift': -1.0, 'Whitney Houston': -1.0}, 'Taylor Swift': {'PSY': 1.0, 'Whitney Houston': 0.0}, 'Whitney Houston': {'PSY': 1.0, 'Taylor Swift': 0.0}}
```

Construyendo el método paso a paso

Step 3

```
def computeDeviations(self):  
    # for each person in the data:  
    #     get their ratings  
    for ratings in self.data.values():  
        # for each item & rating in that set of ratings:  
        for (item, rating) in ratings.items():  
            self.frequencies.setdefault(item, {})  
  
            self.deviations.setdefault(item, {})  
            # for each item2 & rating2 in that set of ratings:  
            for (item2, rating2) in ratings.items():  
                if item != item2:  
                    # add the difference between the ratings  
                    # to our computation  
                    self.frequencies[item].setdefault(item2, 0)  
                    self.deviations[item].setdefault(item2, 0.0)  
                    self.frequencies[item][item2] += 1  
                    self.deviations[item][item2] += rating - rating2
```

Construyendo el método paso a paso

	Taylor Swift	PSY	Whitney Houston
Amy	4	3	4
Ben	5	2	?
Clara	?	3.5	4
Daisy	5	?	3

Step 4:

```
for (item, ratings) in self.deviations.items():  
    for item2 in ratings:  
        ratings[item2] /= self.frequencies[item][item2]
```


Construyendo el método paso a paso

```
josh@manu:~/Descargas/TBDS$ python recommender3.py
***** Frecuencia *****
{'PSY': {'Taylor Swift': 1, 'Whitney Houston': 1}, 'Taylor Swift': {'PSY': 1, 'Whitney Houston': 1}, 'Whitney Houston': {'PSY': 1, 'Taylor Swift': 1}}
***** Desviacion *****
{'PSY': {'Taylor Swift': -1.0, 'Whitney Houston': -1.0}, 'Taylor Swift': {'PSY': 1.0, 'Whitney Houston': 0.0}, 'Whitney Houston': {'PSY': 1.0, 'Taylor Swift': 0.0}}

***** Frecuencia *****
{'PSY': {'Taylor Swift': 1, 'Whitney Houston': 2}, 'Taylor Swift': {'PSY': 1, 'Whitney Houston': 1}, 'Whitney Houston': {'PSY': 2, 'Taylor Swift': 1}}
***** Desviacion *****
{'PSY': {'Taylor Swift': -1.0, 'Whitney Houston': -1.5}, 'Taylor Swift': {'PSY': 1.0, 'Whitney Houston': 0.0}, 'Whitney Houston': {'PSY': 1.5, 'Taylor Swift': 0.0}}

***** Frecuencia *****
{'PSY': {'Taylor Swift': 2, 'Whitney Houston': 1}, 'Taylor Swift': {'PSY': 2, 'Whitney Houston': 1}, 'Whitney Houston': {'PSY': 2, 'Taylor Swift': 1}}
***** Desviacion *****
{'PSY': {'Taylor Swift': -4.0, 'Whitney Houston': -1.5}, 'Taylor Swift': {'PSY': 4.0, 'Whitney Houston': 0.0}, 'Whitney Houston': {'PSY': 1.5, 'Taylor Swift': 0.0}}

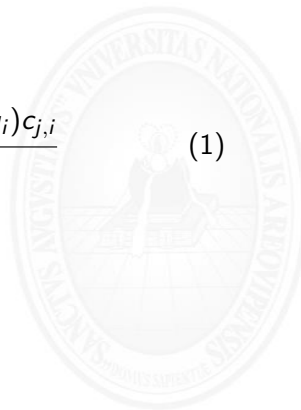
***** Frecuencia *****
{'PSY': {'Taylor Swift': 2, 'Whitney Houston': 2}, 'Taylor Swift': {'PSY': 2, 'Whitney Houston': 2}, 'Whitney Houston': {'PSY': 2, 'Taylor Swift': 2}}
***** Desviacion *****
{'PSY': {'Taylor Swift': -4.0, 'Whitney Houston': -1.5}, 'Taylor Swift': {'PSY': 4.0, 'Whitney Houston': 2.0}, 'Whitney Houston': {'PSY': 1.5, 'Taylor Swift': -2.0}}
```

Construyendo el método paso a paso

```
>>> r = recommender(users2)
>>> r.computeDeviations()
>>> r.deviations
{'PSY': {'Taylor Swift': -2.0, 'Whitney Houston': -0.75}, 'Taylor Swift': {'PSY': 2.0, 'Whitney Houston': 1.0}, 'Whitney Houston': {'PSY': 0.75, 'Taylor Swift': -1.0}}
```

	Taylor Swift	PSY	Whitney Houston
Taylor Swift	0	2	1
PSY	-2	0	-0.75
Whitney Houston	-1	0.75	0

$$P^{wS1}(u)_j = \frac{\sum_{i \in S(u) - \{j\}} (dev_{j,i} + u_i) c_{j,i}}{\sum_{i \in S(u) - \{j\}} c_{j,i}} \quad (1)$$



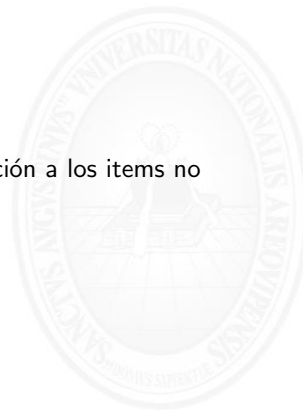
Analizando el código

```
def slopeOneRecommendations(self, userRatings):
    recommendations = {}
    frequencies = {}
    # for every item and rating in the user's recommendations
    for (userItem, userRating) in userRatings.items():
        # for every item in our dataset that the user didn't rate
        for (diffItem, diffRatings) in self.deviations.items():
            if diffItem not in userRatings and \
               userItem in self.deviations[diffItem]:
                freq = self.frequencies[diffItem][userItem]
                recommendations.setdefault(diffItem, 0.0)
                frequencies.setdefault(diffItem, 0)
                # add to the running sum representing the numerator
                # of the formula
                recommendations[diffItem] += (diffRatings[userItem] +
                                                userRating) * freq
                # keep a running sum of the frequency of diffitem
                frequencies[diffItem] += freq
    recommendations = [(self.convertProductID2name(k),
                        v / frequencies[k])
                       for (k, v) in recommendations.items()]
    # finally sort and return
    recommendations.sort(key=lambda artistTuple: artistTuple[1],
                        reverse = True)
    # I am only going to return the first 50 recommendations
    return recommendations[:50]
```

Figura: Implementación de la ecuación (1)

Users	Frecuencia
-------	------------

Cuadro: Para cada item y usuario calcula una desviación a los items no seleccionados por el usuario





Ron Zacharski. A Programmer's Guide to Data Mining: The Ancient Art of the Numerati.



Filtros Colaborativos

Rómulo Condori
Kevin Valverde Huilca
Ruben Huanca Morales
José Torres Lima
Diego Bellido Ramos
Luis Mamani Chirinos

*Escuela Ciencia de la Computación
Universidad Nacional de San Agustín

22 de Abril del 2019

