

Métricas de Evaluación


IIC 3633 - Sistemas Recomendadores

Denis Parra

Profesor Asistente, DCC, PUC CHile

Tarea 1

- [Enlace secreto](#)



Creación y evaluación de un sistema recomendador

Secret url: https://competitions.codalab.org/competitions/111817?secret_key=0adbc4be-062d-4c59-93cc-4710a65b946e

Organized by Idonoso - Current server time: Aug. 15, 2016, 7:41 p.m. UTC

First phase	End
Training	Competition Ends
Aug. 16, 2016, midnight UTC	Sept. 4, 2016, midnight UTC

Learn the Details Phases Participate Results Forums ↗

- Overview
- Evaluation**
- Terms and Conditions

Entregables

- Informe
 - Análisis exploratorio de los datos
 - Definición de experimentos
 - Exposición de resultados
 - Análisis de sensibilidad de los parámetros
 - Conclusiones

TOC

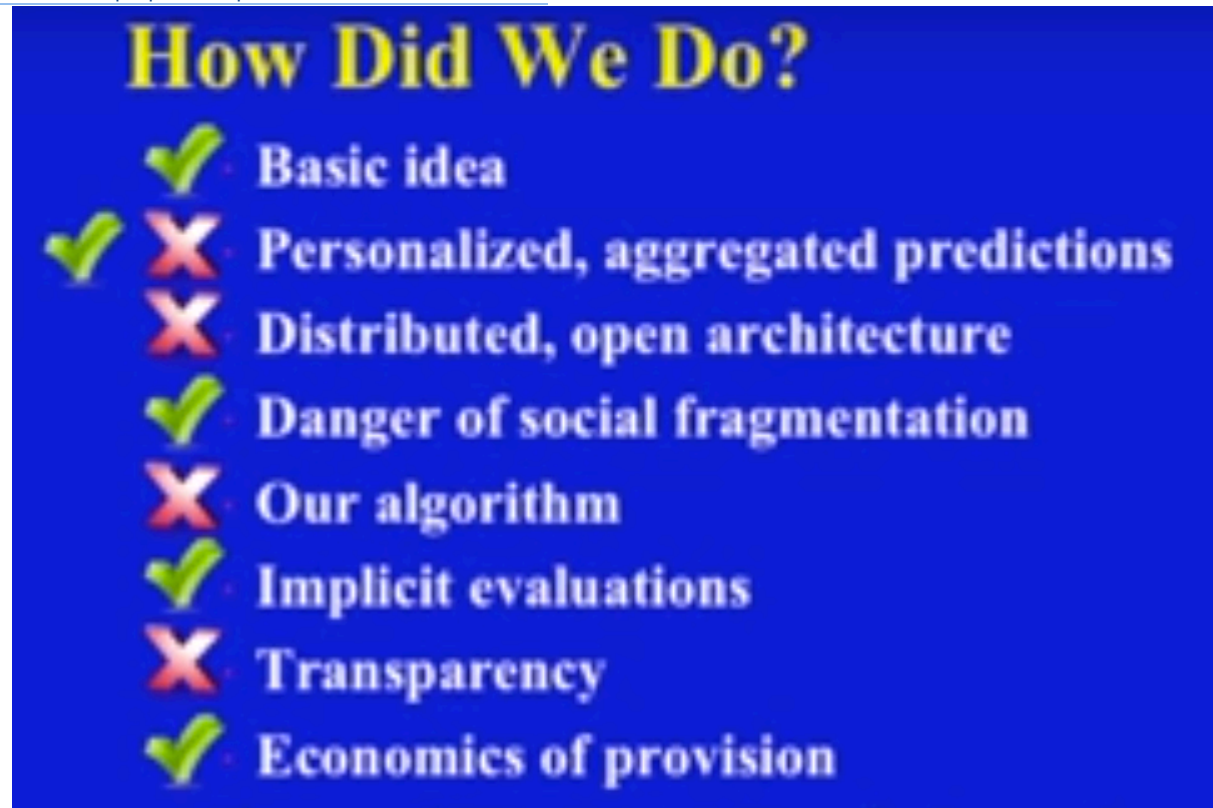
En esta clase

1. Resumen + Próxima Semana
2. Predicción de Ratings: MAE, MSE, RMSE
3. Evaluación via Precision-Recall
4. Métricas P@n, MAP,
5. Métricas de Ranking: DCG, nDCG,
6. Métricas en Tarea 1

Con respecto al paper sobre CF de Resnick et al. (1994)

- Ver Video de "re-presentación" del paper por P. Resnick y John Riedl en CSCW 2013, conmemorando que ha sido el paper más citado de dicha conferencia:

[Video CF paper re-presented at CSCW2013](#)



Resumen + Próxima Semana

- **Ranking no personalizado:** Ordenar items considerando el porcentaje de valoraciones positivas y la cantidad total de valoraciones.
- **Filtrado Colaborativo:** Basado en Usuario y en Items. Parámetros principales (K, métrica de distancia), ajustes por baja cantidad de valoraciones.
- Slope One: Eficiencia y Escalabilidad por sobre la precisión
- Métricas de Evaluación
- Próxima Semana: Content-based filtering y tag-based recommenders

Evaluación Tradicional: Predicción de Ratings

MAE: Mean Absolute Error

$$MAE = \frac{\sum_{i=1}^n |\hat{r}_{ui} - r_{ui}|}{n}$$

MSE: Mean Squared Error

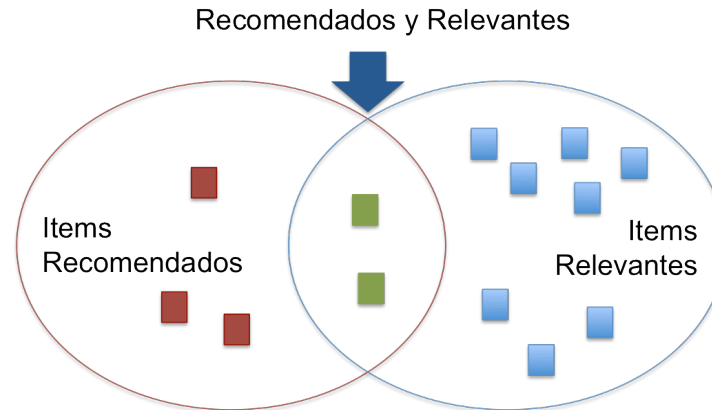
$$MSE = \frac{\sum_{i=1}^n (\hat{r}_{ui} - r_{ui})^2}{n}$$

RMSE: Root Mean Squared Error

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{r}_{ui} - r_{ui})^2}{n}}$$

Evaluación de una Lista de Recomendaciones

Si consideramos los elementos recomendados como un conjunto S y los elementos relevantes como el conjunto R , tenemos:



Luego, Precision es:

$$Precision = \frac{|Recomendados \cap Relevantes|}{|Recomendados|}, y$$

$$Recall = \frac{|Recomendados \cap Relevantes|}{|Relevantes|}$$

Ejemplo 1: Precision y Recall

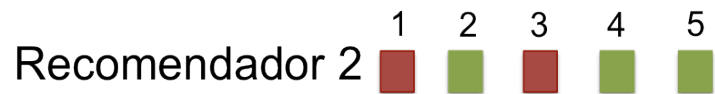
Si bien la lista de recomendaciones está rankeada, para estas métricas la lista se entiende más bien como un conjunto.

Total Relevantes  X 20



Precision =??

Recall =??



Precision =??

Recall =??

Ejemplo 1: Precision y Recall

Total Relevantes  X 20



$$Precision = \frac{5}{10} = 0,5$$

$$Recall = \frac{5}{20} = 0,25$$

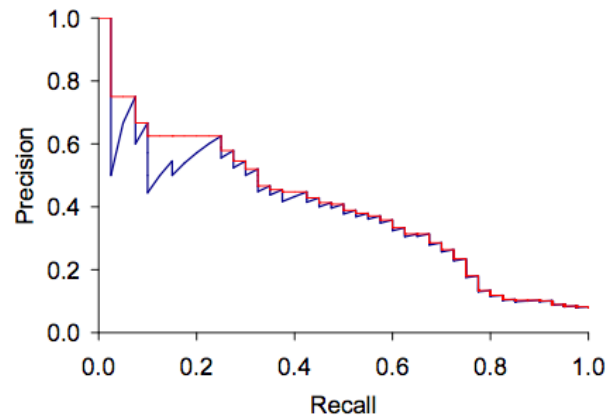


$$Precision = \frac{3}{5} = 0,6$$

$$Recall = \frac{3}{20} = 0,15$$

Compromiso entre Precision y Recall

Al aumentar el Recall (la proporción de elementos relevantes) disminuimos la precision, por lo cual hay un compromiso entre ambas métricas.



► Figure 8.2 Precision/recall graph.

Por ello, generalmente reportamos la media armónica entre ambas métricas:

$$F_{\beta=1} = \frac{2 * Precision * Recall}{P + R}$$

- Ref: <http://nlp.stanford.edu/IR-book/pdf/08eval.pdf>

De evaluación de Conjuntos a Ranking

- Mean Reciprocal Rank (MRR)
- Precision@N
- MAP
- Rank score
- DCG
- nDCG

Mean Reciprocal Rank (MRR)

Consideramos la posición en la lista del primer elemento relevante.

$$MRR = \frac{1}{r}, \text{ donde } r: \text{ ranking del 1er elemento relevante}$$



$$MRR_1 = ??$$



$$MRR_2 = ??$$

Problema: Usualmente tenemos más de un elemento relevante!!

Mean Reciprocal Rank (MRR)

Consideramos la posición en la lista del primer elemento relevante.

$$MRR = \frac{1}{r}, \text{ donde } r: \text{ ranking del 1er elemento relevante}$$



$$MRR_1 = \frac{1}{2} = 0,5$$



$$MRR_2 = \frac{1}{2} = 0,5$$

Problema: Usualmente tenemos más de un elemento relevante!!

Precision at N (P@N)

Corresponde a la *precision* en puntos específicos de la lista de items recomendados. En otras palabras, dado un ranking específica en la lista de recomendaciones, qué proporción de elementos relevantes hay hasta ese punto

$$Precision@n = \frac{\sum_{i=1}^n Rel(i)}{n}, \text{ donde } Rel(i) = 1 \text{ si elemento es relevante}$$



Precision@5 = ??



Precision@5 = ??

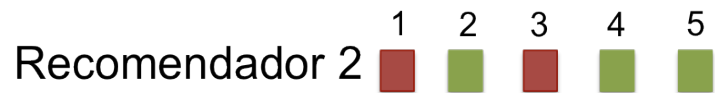
Precision at N (P@N)

Corresponde a la *precision* en puntos específicos de la lista de items recomendados. En otras palabras, dado un ranking específica en la lista de recomendaciones, qué proporción de elementos relevantes hay hasta ese punto

$$Precision@n = \frac{\sum_{i=1}^n Rel(i)}{n}, \text{ donde } Rel(i) = 1 \text{ si elemento es relevante}$$



$$Precision@5 = \frac{2}{5} = 0,4$$



$$Precision@5 = \frac{3}{5} = 0,6$$

Pro: permite evaluar topN; Problema: aún no permite una evaluación orgánica del los items con *ranking* < *n*.

Mean Average Precision (MAP)

Average Precision (AP)

- El AP se calcula sobre una lista única de recomendaciones, al promediar la precisión cada vez que encontramos un elemento relevante, es decir, en cada recall point.

$$AP = \frac{\sum_{k \in K} P@k \times rel(k)}{|relevantes|}$$

donde $P@k$ es la precisión en el recall point k , $rel(k)$ es una función que indica 1 si el ítem en el ranking j es relevante (0 si no lo es), y K son posiciones de ranking con elementos relevantes.

MAP es la media de varias "Average Precision"

- Considerando n usuarios en nuestro dataset y que a cada uno de ellos le damos una lista de recomendaciones,

$$MAP = \frac{\sum_{u=1}^n AP(u)}{m}, \text{ donde } m \text{ es el número de usuarios.}$$

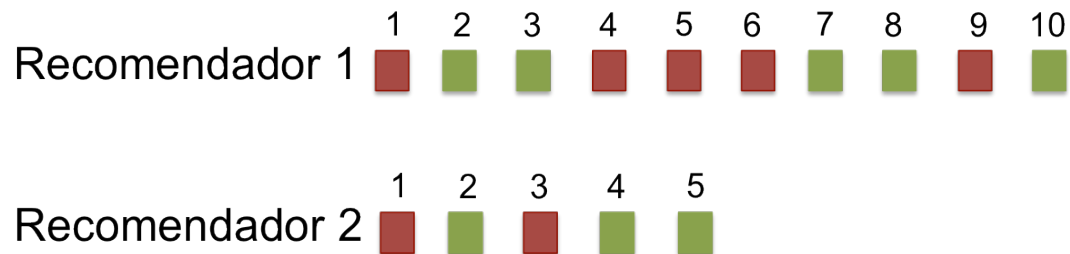
Mean Average Precision (MAP) - II

Como no siempre sabemos de antemano el número de relevantes o puede que hagamos una lista que no alcanza a encontrar todos los elementos relevantes, podemos usar una formulación alternativa** para **Average Precision (AP@n)**

$$AP@n = \frac{\sum_{k \in K} P@k \times rel(k)}{\min(m, n)}$$

donde n es el máximo número de recomendaciones que estoy entregando en la lista, y m es el número de elementos relevantes.

- Ejercicio: calcule $AP@n$ y luego $MAP@n$, con $n = 10$, y $m = 20$ de:



** <https://www.kaggle.com/wiki/MeanAveragePrecision>

Rankscore

- Rank Score se define como la tasa entre el Rank Score de los items correctos respecto al mejor Rank Score alcanzable por el usuario en teoría.

PARAMETROS

- h el conjunto de items correctamente recomendados, i.e. hits
- rank retorna la posición (rank) de un item
- T es el conjunto de items de interés
- α es el ranking half life, i.e. un factor de reducción exponencial

FORMULA

$$rankscore = \frac{rankscore_p}{rankscore_{max}}$$

$$rankscore_p = \sum_{i \in h} 2^{-\frac{rank(i)-1}{\alpha}}$$

$$rankscore_{max} = \sum_{i=1}^{|T|} 2^{-\frac{i-1}{\alpha}}$$

DCG y nDCG

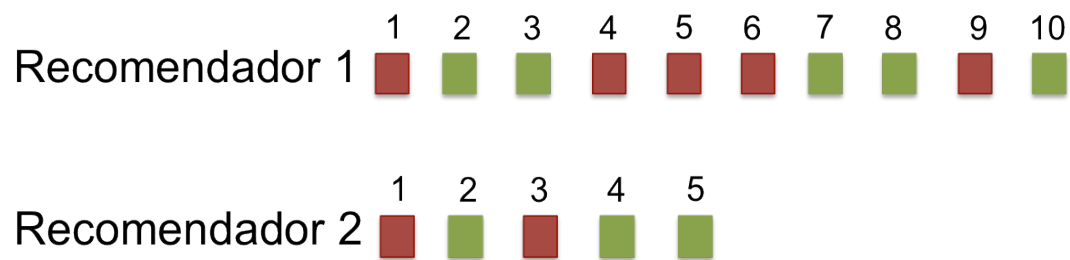
- DCG: Discounted cumulative Gain

$$DCG = \sum_i^p \frac{2^{rel_i} - 1}{\log_2(1 + i)}$$

- nDCG: normalized Discounted cumulative Gain, para poder comparar listas de distinto largo

$$nDCG = \frac{DCG}{iDCG}$$

Ejercicio: Calcular nDCG para



Coverage

- Como no a todos los usuarios se logran hacer recomendaciones, consideramos en la evaluación el **User Coverage**, el porcentaje de usuarios a los cuales se les pudo hacer recomendaciones.
- Como no a todos los items pueden ser recomendaciones, consideramos en la evaluación el **Item Coverage**, el porcentaje de items que fueron recomendados al menos una vez.

Métricas para Tarea 1

- $\text{Precision@10} = \text{Recall@10}$, (ya que estamos "forzando" recomendados = relevantes)
- MAP (en realidad, será MAP@10)
- nDCG

Ejemplo con R

- Paquete rrecsys

```
library(rrecsys)
```

```
data("mlLatest100k")
```

```
ML <- defineData(mlLatest100k, minimum = .5, maximum = 5, halfStar = TRUE)  
ML
```

```
## Dataset containing 718 users and 8927 items.
```

R library(rrecsys)

```
# rowRatings(ML) : number of ratings per row  
# colRatings(ML) : number of ratings per columns  
  
numRatings(ML)
```

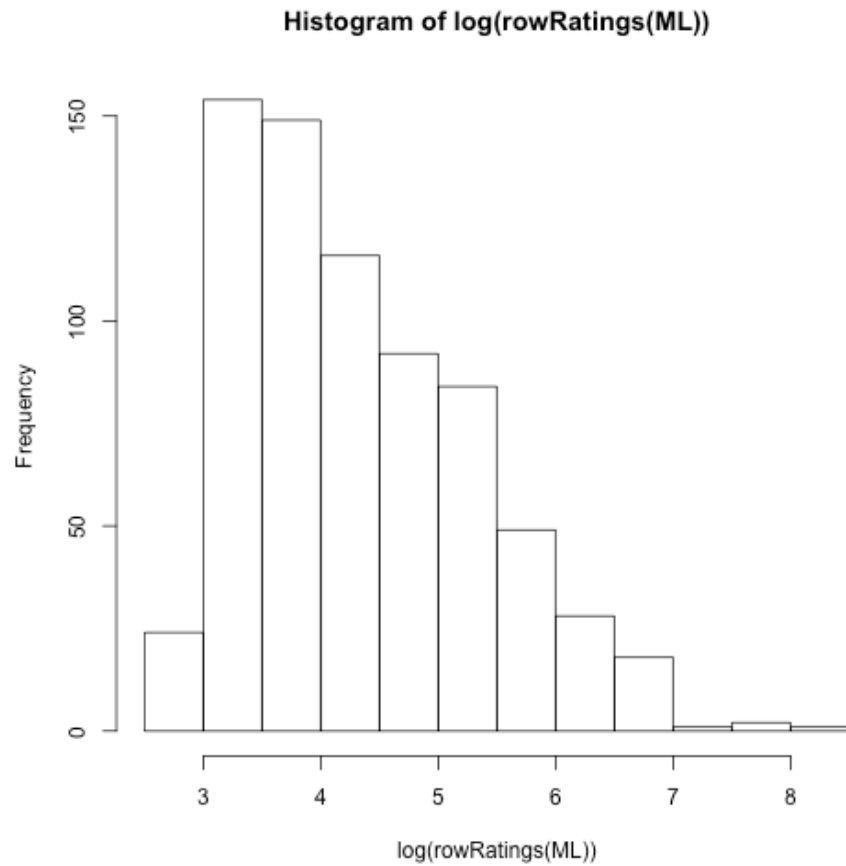
```
## [1] 100234
```

```
sparsity(ML)
```

```
## [1] 0.9843619
```

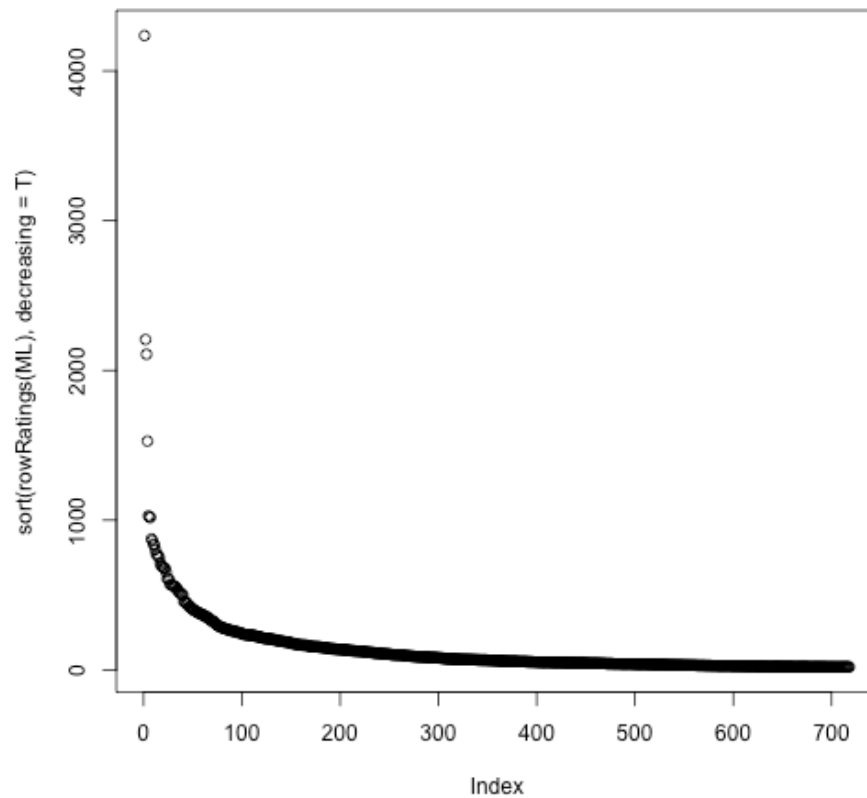
R library(rrecsys)

```
# Ratings por usuario  
hist( log(rowRatings(ML)) )
```



R library(rrecsys)

```
plot(sort(rowRatings(ML), decreasing = T))
```



Filtering dataset

```
subML <- ML[rowRatings(ML)>=40, colRatings(ML)>=30]  
sparsity(subML)
```

```
## [1] 0.8683475
```

Recomendación No personalizada

```
# RecSys no personalizado ----  
globAv <- rrecsys(subML, alg = "globalaverage")  
# predict and recommend for all  
p_globAv <- predict(globAv) # output: matriz de predicciones  
r_globAv <- recommend(globAv, topN = 2) # output: lista con recomendaciones
```

Evaluación de predicciones

```
e <- evalModel(subML, folds = 5)
evalPred(e, 'globalaverage')
```

```
##
## Fold: 1 / 5 elapsed. Time: 0.03804708
##
##
## Fold: 2 / 5 elapsed. Time: 0.03415608
##
##
## Fold: 3 / 5 elapsed. Time: 0.03495908
##
##
## Fold: 4 / 5 elapsed. Time: 0.04891706
##
##
## Fold: 5 / 5 elapsed. Time: 0.04247308
##
## The model was trained on the dataset using globalAverage algorithm.
```

```
##          MAE      RMSE globalMAE globalRMSE
## 1-fold  0.8372151 0.9930604 0.8069286 1.0100655
## 2-fold  0.8431685 0.9997370 0.8093622 1.0120210
## 3-fold  0.8228661 0.9798062 0.7970749 1.0010374
## 4-fold  0.8259744 0.9794638 0.7957841 0.9992901
## 5-fold  0.8242592 0.9792238 0.7988995 1.0038996
## Average 0.8306966 0.9862583 0.8016099 1.0052627
```

Evaluación de recomendación topN

```
evalRec(e, 'globalaverage', goodRating = 4, topN = 5)
```

```
## Evaluating top- 5 recommendation with globalAverage .
##
## Fold: 1 / 5 elapsed. Time: 0.229358
##
##
## Fold: 2 / 5 elapsed. Time: 0.229378
##
##
## Fold: 3 / 5 elapsed. Time: 0.2041221
##
##
## Fold: 4 / 5 elapsed. Time: 0.188154
##
##
## Fold: 5 / 5 elapsed. Time: 0.1923082
##
## The model was trained on the dataset using globalAverage algorithm.
## Item coverage: 2.115159 %.
##
## User coverage: 100 %.
```

```
##          TP      FP      TN      FN precision  recall
## 1-fold  0.1391129  4.860887  833.2641  12.73589  0.02782258  0.02432124
## 2-fold  0.1653226  4.834677  833.1492  12.85081  0.03306452  0.02173401
## 3-fold  0.1391129  4.860887  833.1169  12.88306  0.02782258  0.03048048
## 4-fold  0.1653226  4.834677  833.2198  12.78024  0.03306452  0.03343806
## 5-fold  0.1310484  4.868952  833.3226  12.67742  0.02620968  0.02756662
## Average 0.1479839  4.852016  833.2145  12.78548  0.02959677  0.02750808
##          F1      nDCG rankscore
## 1-fold  0.01332506  0.1679583  0.1907407
## 2-fold  0.01691046  0.1965084  0.2187366
## 3-fold  0.01502101  0.1726703  0.2029356
## 4-fold  0.01725036  0.1902900  0.2223472
## 5-fold  0.01245742  0.1729913  0.2001172
## Average 0.01499286  0.1800837  0.2069755
```

Referencias

- Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval (Vol. 1, p. 6). Cambridge: Cambridge university press.
- Baeza-Yates, R., & Ribeiro-Neto, B. (1999). Modern information retrieval (Vol. 463). New York: ACM press.
- Slides "Evaluating Recommender Systems"
http://www.math.uci.edu/icamp/courses/math77b/lecture_12w/pdfs/Chapter%2007%20-%20Evaluating%20recommender%20systems.pdf