

Métricas de Evaluación

IIC 3633 - Sistemas Recomendadores

Denis Parra

TOC

En esta clase

1. Prediccion de Ratings: MAE, MSE, RMSE
2. Evaluacion via Precision-Recall
3. Metricas P@n, MAP,
4. Metricas de Ranking: DCG, nDCG,
5. Metricas en Tarea 1

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

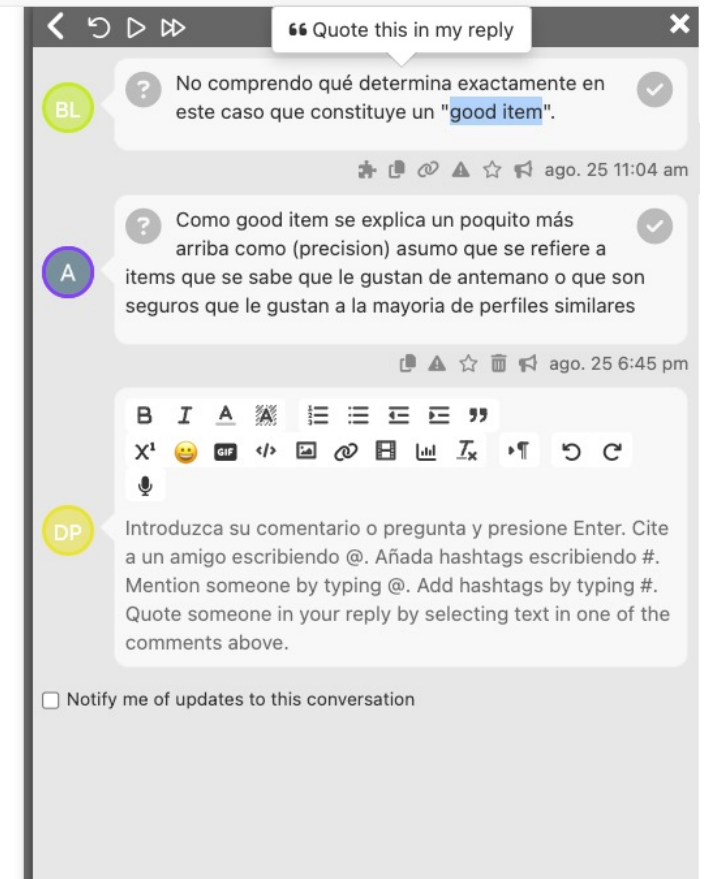
$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{r}_{ui} - r_{ui})^2}{n}}$$

Un “buen item” para recomendar

Support metrics give us a way of understanding how well the recommendation could support a user in deciding which items to consume.

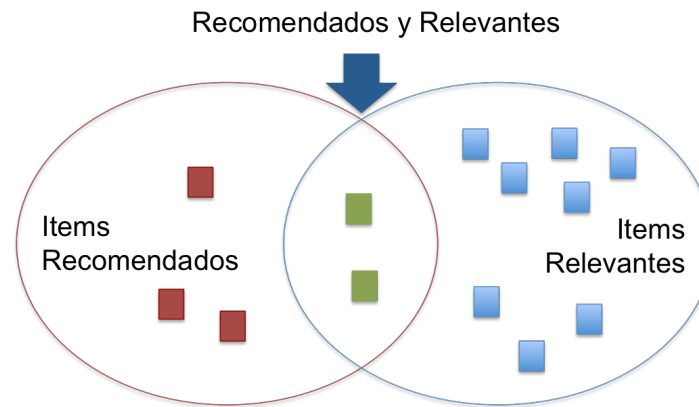
The basic workflow of all decision support metrics is to first perform a recommendation, then compare that recommendation against a previously selected “relevant item set”. The relevant item set represents those items that we know to be good items to recommend to a user. You then count how many of the recommended items were relevant and how many were not.

For items in a larger scale system, a choice needs to be made about which items to consider relevant. The easiest choice would be to take all rated items in the test set as good items, which evaluates an algorithm on its ability to select items that the user will see. More useful, however, is the practice of choosing a cutoff such as four out of five stars at which we consider a recommendation ‘good enough’. Best practices recommend testing with multiple similar cutoffs to ensure that results are robust across various choices for defining relevant items. Evaluation results that favor one algorithm with items rated 4 and above as “good”, but another algorithm if 4.5 and above are “good”, deserve more careful consideration.



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

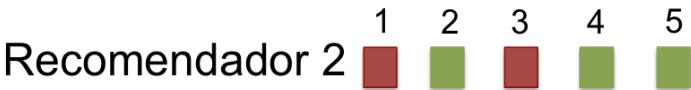
Si bien la lista de recomendaciones está rankeada, para estas métricas la lista se entiende más bien como un conjunto.

Total Relevantes  X 20



Precision =??

Recall =??



Precision =??

Recall =??

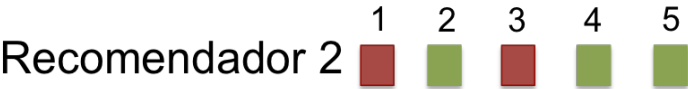
Ejemplo 1: Precision y Recall

Total Relevantes  X 20



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

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

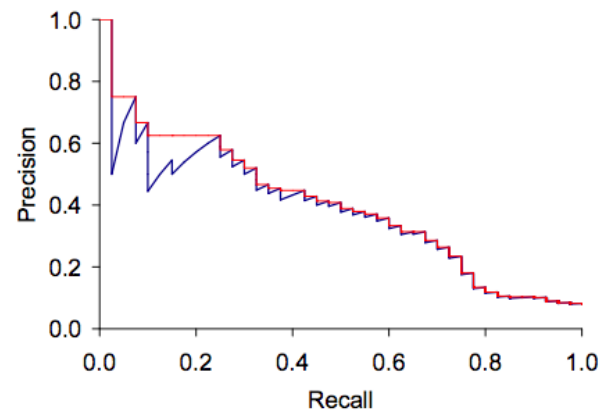


$$Precision = \frac{3}{5} = 0,6$$

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

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

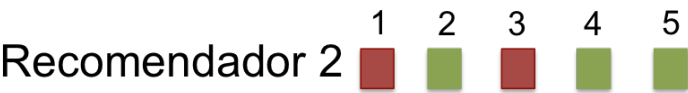
Mean Reciprocal Rank (MRR)

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 precision 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 precision 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,

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

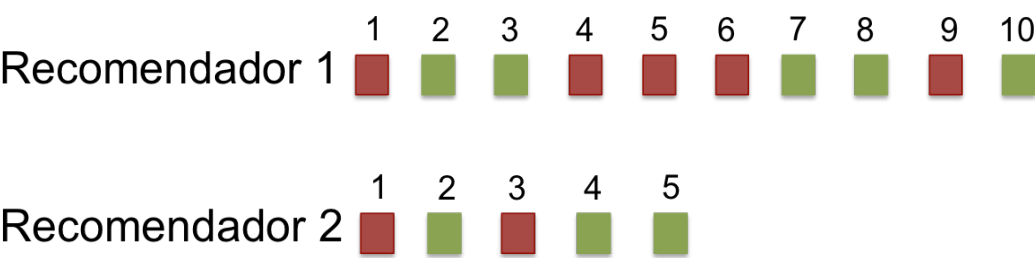
Mean Average Precision (MAP) - II

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>

DCG y nDCG

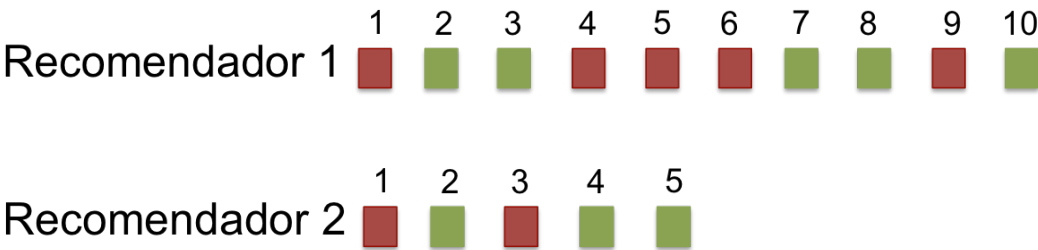
- DCG: Discounted cummulative Gain

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

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

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

Ejercicio: Calcular nDCG para



Half Life

- Una alternativa al nDCG es la métrica llamada Half life, que se define como:

$$HalfLife(u) = \sum_{i \in Rec} \frac{\max(r_{ui} - d, 0)}{2^{(k_i - 1)/(\alpha - 1)}} \quad (35)$$

- Tiene una función de descuento exponencial más rigurosa
- Hay dos parámetros importantes:
 - d : nos indica el “rating neutral”
 - α : velocidad de decaimiento exponencial. Debería fijarse de manera que un ítem con ranking α tiene una chance de 50% de ser observado por usuarios

Comentarios en Perusall

A similar metric to this is the *half life utility metric* [9]. Half life utility uses a faster exponential discounting function. The half life utility is as follows:

$$HalfLife(u) = \sum_{i \in Rec} \frac{\max(r_{ui} - d, 0)}{2^{(k_i - 1)/(\alpha - 1)}} \quad (35)$$

Half life utility has two parameters. The first is d which is a score that should represent the neutral rating value. A recommendation for an item with score d should neither help nor hurt the user, while any item rated above d should be good recommendations. d should also be used as the “default” rating value for

NE

“Half Life Utility” sugiere un equilibrio entre recomendar ítems altamente calificados y mantener la diversidad para explorar nuevas opciones. Ajustar alfa para que los ítems en posiciones inferiores aún tengan una oportunidad razonable de ser vistos puede ayudar a promover la diversidad en las recomendaciones, evitando que los usuarios se limiten a un conjunto específico.

EC

Supongo que esta métrica es mejor para listas muy cortas dado su decaimiento exponencial.

PG

Creo que igual puede ser muy útil para listas largas, ya que hay más elementos que puedes comparar y así puedes elegir con mayor precisión los elementos que van al inicio de la lista

GC

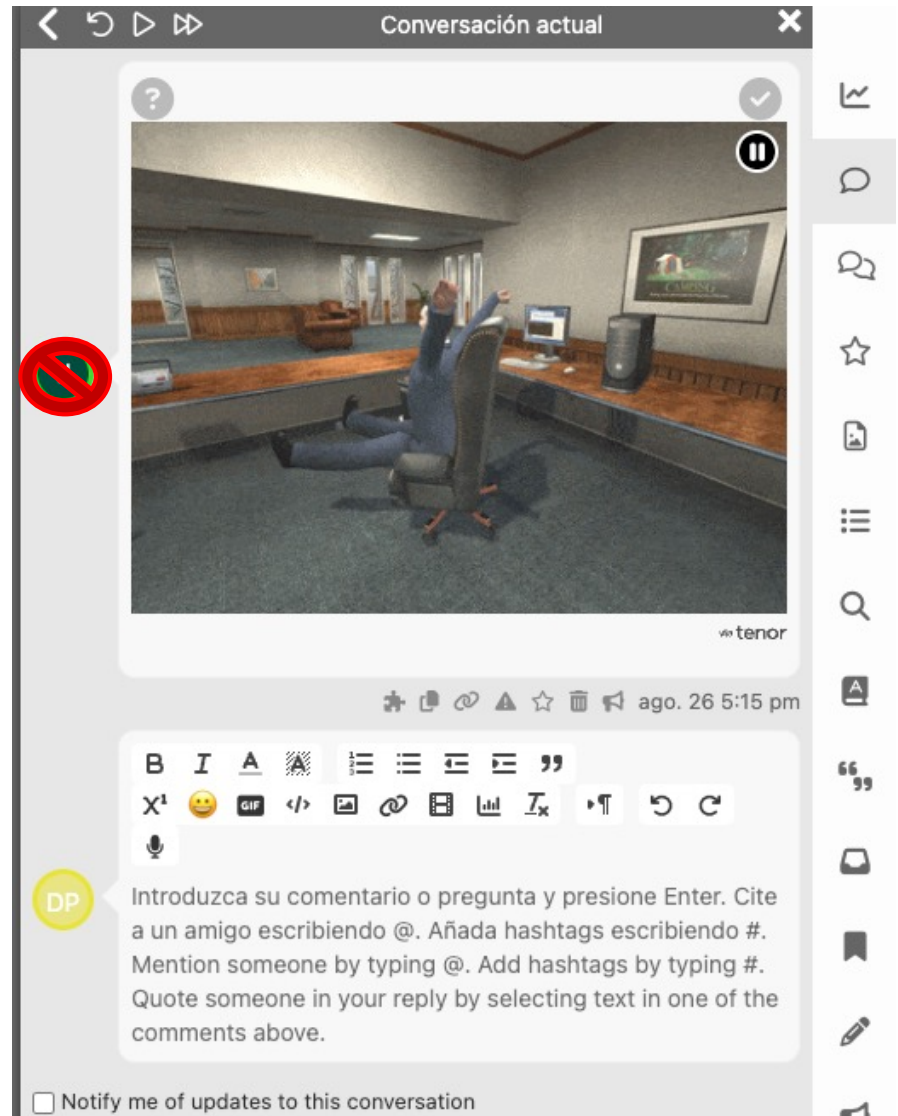
Estoy de acuerdo con Paula e incluso creo que puede ser más útil para listas largas que para listas cortas, ya que de esta manera, manipulando bien los parámetros se puede obtener una lista de recomendaciones mejor limitada sin necesidad de fijar un n de recomendaciones.

Comentarios en Perusal

A similar metric to this is the *half life utility metric* [9]. Half life utility uses a faster exponential discounting function. The half life utility is as follows:

$$HalfLife(u) = \sum_{i \in Rec} \frac{\max(r_{ui} - d, 0)}{2^{(k_i - 1)/(\alpha - 1)}} \quad (35)$$

Half life utility has two parameters. The first is d which is a score that should represent the neutral rating value. A recommendation for an item with score d should neither help nor hurt the user, while any item rated above d should be good recommendations. d should also be used as the “default” rating value for



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.

Rendimiento de una lista: Kendall-Tau

Se compara el resultado de ranking como lista, respecto a una lista que representa el "ground truth". En el contexto RecSys, se ha usado una modificación llamada AP correlation:

$$\tau_{ap} = \frac{2}{N-1} \cdot \left[\sum_{i \in I} \frac{C(i)}{\text{index}(i) - 1} \right] - 1$$

N es el numero de items rankeados en la lista, $C(i)$ el numero de items reankeados bajo $\text{index}(i)$ de forma correcta. Valores de *APcorrelation* van entre +1 to -1. Un problema que tiene es que asume un orden total, con un orden parcial de los elementos no es útil.

Diversity (Ziegler)

Esta métrica se calcula sobre una lista de recomendaciones. Se compara la similaridad entre los pares de elementos recomendados, obteniendo la **Intra-list Similarity**

$$ILS(P_{w_i}) = \frac{\sum_{b_k \in P_{w_i}} \sum_{b_c \in P_{w_i}, b_k \neq b_c} c_o(b_k, b_c)}{2}$$

Valores altos de ILS denotan menor diversidad en la lista. Basado en esta métrica, los autores proponen un algoritmo de diversificación. Los resultados de un estudio off-line y online muestran que la satisfacción del usuario va más allá de la precisión de la recomendación, incluyendo la diversidad percibida de las recomendaciones.

Ref: Ziegler, C. N., McNee, S. M., Konstan, J. A., & Lausen, G. (2005, May). Improving recommendation lists through topic diversification. In Proceedings of the 14th international conference on World Wide Web (pp. 22-32). ACM.

Diversidad (Lathia) en el tiempo

Lathia compara diversidad y novedad a lo largo del tiempo. La razón $L2/L1$ corresponde a la fracción de elementos de $L2$ que no están en la lista $L1$.

$$diversity(L1, L2, N) = \frac{|\frac{L2}{L1}|}{N}$$

Por otro lado, "novelty" compara la última lista recomendada $L2$ con respecto al conjunto de todos los ítems recomendados a la fecha A_t .

$$novelty(L2, N) = \frac{|\frac{L2}{A_t}|}{N}$$

Ref: Lathia, N., Hailes, S., Capra, L., & Amatriain, X. (2010, July). Temporal diversity in recommender systems. In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval (pp. 210-217). ACM.

Mean Percentage Ranking (Implicit Feedback)

$$MPR = \frac{\sum_{ui} r_{ui}^t \cdot \overline{rank_{ui}}}{\sum_{ui} r_{ui}^t}$$

Donde r_{ui} indica si el usuario u consumio el item i y $\overline{rank_{ui}}$ denota el percentile-ranking de i dentro de una lista ordenada. De esta forma, $\overline{rank_{ui}} = 0\%$ significa que i está al tope de la lista.

Ref: Hu, Y., Koren, Y., & Volinsky, C. (2008, December). Collaborative filtering for implicit feedback datasets. In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on (pp. 263-272). IEEE.

10 grandes problemas en sistemas de recomendación - 2020

Denis Parra

Profesor Asociado, Depto de Ciencia de la Computación

Escuela de Ingeniería

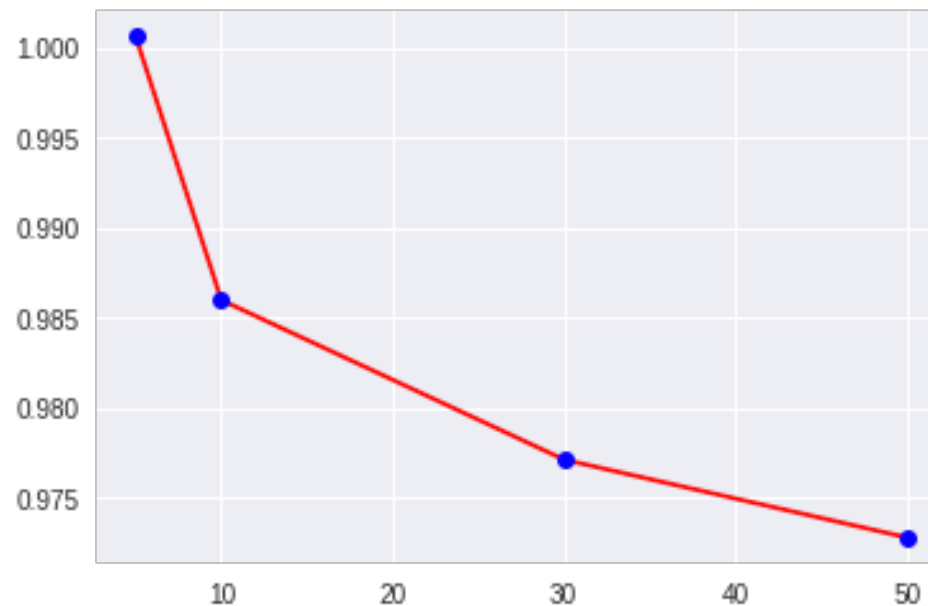
Pontificia Universidad Católica de Chile & IMFD

Problema #5

Métricas

Netflix Prize

- Joe Konstan << terminado el Netflix prize, se vio que mejoras en décimas o centésimas del error de predicción no necesariamente se relacionaban con una mejor experiencia del usuario >>



Academia vs. industria vs. utilidad

Métricas de error y ranking

- RMSE, MAE
- Top-N ranking metrics

Business KPIs

- CTR
- Conversion rates
- Sales increase
- Engagement

Utilidad para el usuario

- Diversidad
- Novedad
- **Serendipia**
- Objetivos de vida

Comentarios

- [Tao Ye] ¿Cómo vincular fácilmente los objetivos comerciales (por ejemplo, más usuarios que regresan, duración de las sesiones) con las métricas de un recsys?
- [Tao Ye] ¿debería uno siempre optimizar todo lo que pueda? ¿Qué margen de ética debe establecer una empresa para equilibrar la ética con los KPI del negocio?

Comentarios

- [Pablo Castells] Veo que los desafíos de la evaluación off-line (procedimientos, dimensiones, sesgos, conjuntos de datos y recursos, etc.) siguen siendo relevantes y también clave para ayudar a que la investigación académica sea relevante para la industria.
- [Pablo Castells] También relacionado con esto, la perspectiva de recomendación **como un proceso cíclico** pueden ser una dirección importante que necesita más trabajo y reflexión (¿relacionado con arreglar Netflix?)

Xiao y Benbasat

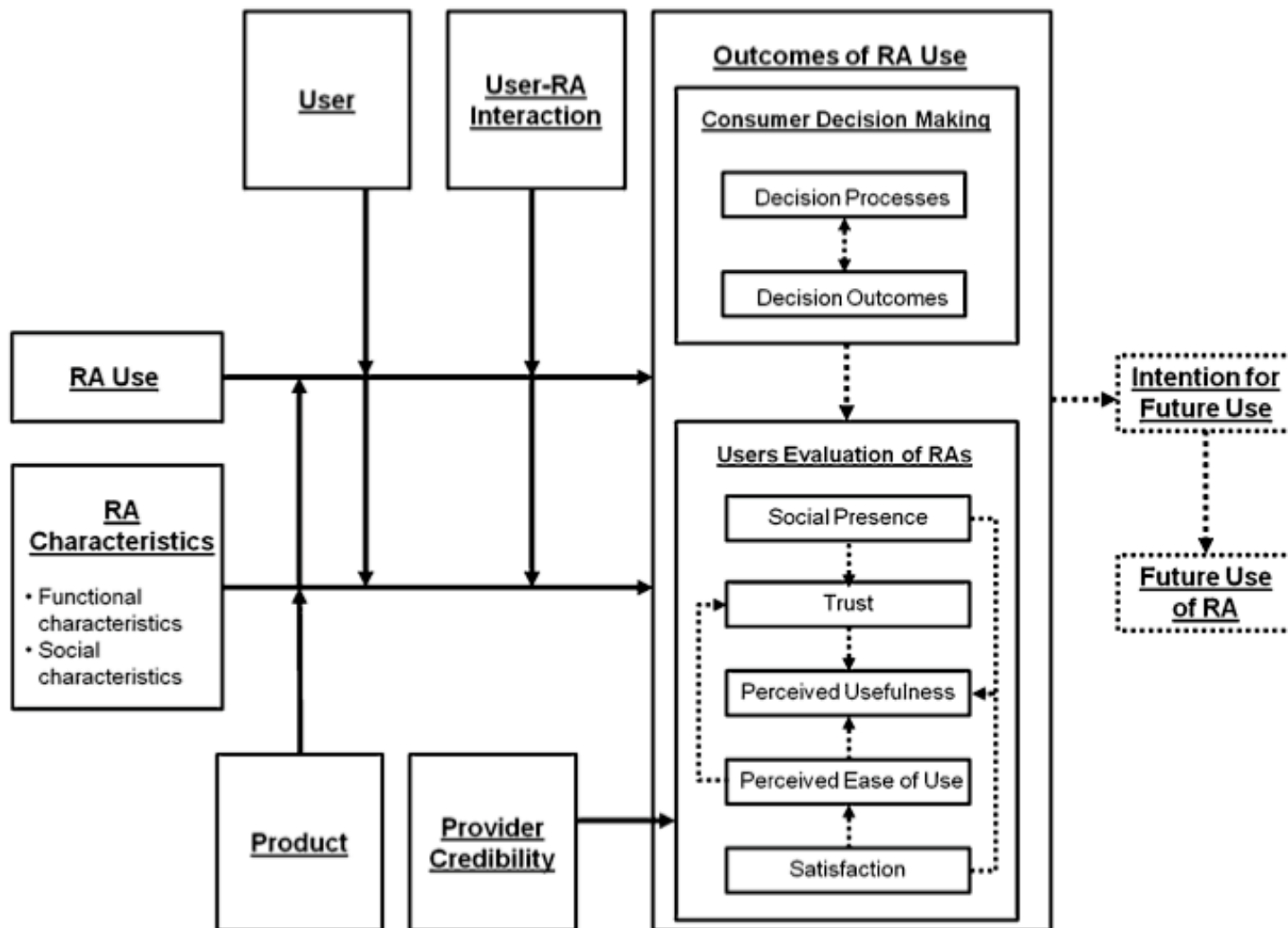


Fig. 2 Updated conceptual model

Framework I - ResQue

- Identifica qué variables definen la experiencia de un usuario con un sistema recomendador

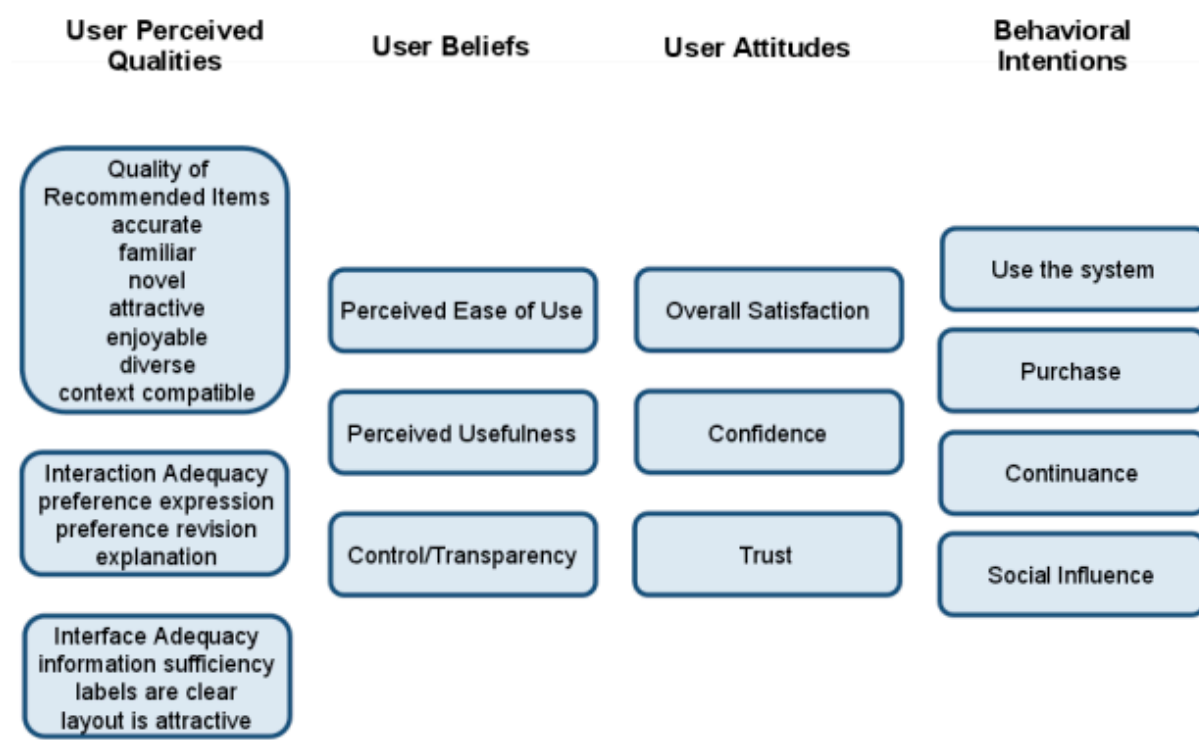


Figure 1: Constructs of an Evaluation Framework on the Perceived Qualities of Recommenders (ResQue).

Framework II

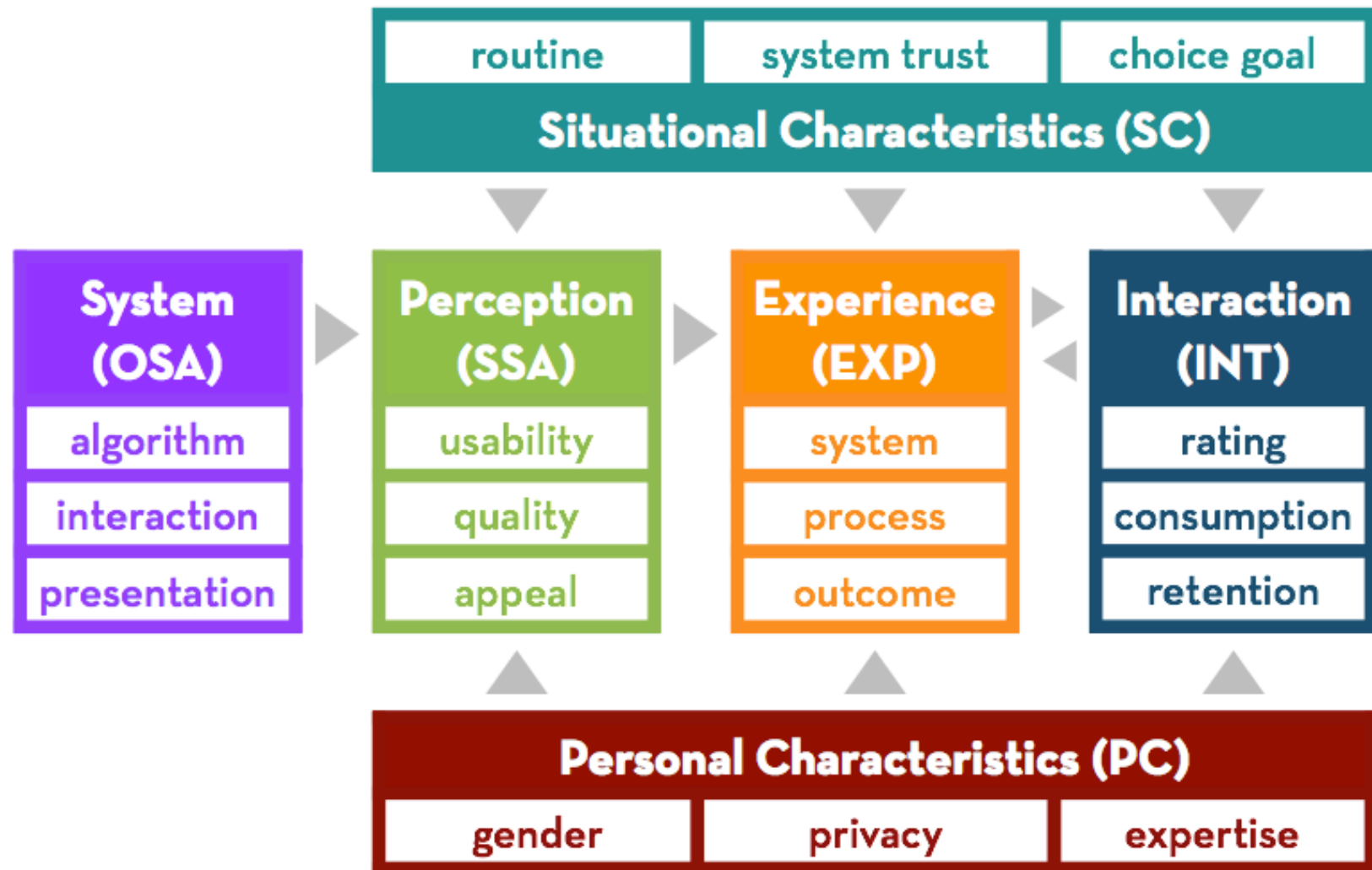


Fig. 1 An updated version of the User-Centric Evaluation Framework [61].

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