Recomendación de música usando un sistema híbrido

tipo cascada, con algoritmos de filtrado colaborativo y basado en contenido

Víctor Gálvez Yanjarí
Pontificia Universidad Católica de Chile
Santiago, Chile
vagalvez@uc.cl

Camilo Ruiz-Tagle Molina
Pontificia Universidad Católica de Chile
Santiago, Chile
cjruiztagle@uc.cl

ABSTRACT

In the world of online music commerce, the recommendation of songs is an indispensable feature for streaming platforms and for buying music in digital format (Amazon, Google Play). For the recommendation of songs, cascade strategies have been used, whose first level is a content-based recommender system and the second level is a collaborative filtering system. The same architecture has been used for the recommendation of TV programs. However, this has not been done for the recommendation of songs. Our proposal seeks to recommend songs using a hydrid implementation, whose first level is a collaborative filtering system and the second level is a content-based system.

CCS CONCEPTS

• **Information systems** → Content analysis and feature selection; Similarity measures.

KEYWORDS

collaborative filtering, content-based, hybrid recommender system

1 ESTADO DEL ARTE

Los sistemas recomendadores son técnicas y herramientas de *software* que proporcionan sugerencias de ítemes para ser utilizados por un usuario [10]. Esta definición puede ser contextualizada y aplicada en los más diversos ámbitos de la industria y del conocimiento. Concretamente, uno de los contextos donde los sistemas recomendadores generan valor y son de gran utilidad, es en la recomendación de música, específicamente de canciones o ítemes a usuarios con la necesidad de una recomendación basada en su interés. Esto ha motivado a grandes empresas de la música, como Spotify, a generar *challenges* para mejorar sus recomendaciones automáticas de *playlists* [11], lo que muestra lo indispensable de buenas recomendaciones para la efectividad de plataformas de streaming y de compra de música en formato digital (tales como Amazon, Google Play). Para esto, usualmente se cuenta con metadata relacionada con las características de la música (título, álbum, categoría, etc.), pero también es posible tener información clave para el desarrollo de sistemas recomendadores como *ratings* y comentarios (*reviews*). Resulta de interés combinar ambas entradas (*ratings* y *reviews*) para generar una buena recomendación, lo que da lugar a la utilización de sistemas híbridos de recomendación o mixturas de algoritmos con enfoques complementarios cuya interacción puede ser beneficiosa para la calidad de la recomendación.

IIC3633 - Sistemas Recomendadores, Diciembre 2018, Santiago, Chile

Para este tipo de entradas, existen sistemas híbridos que ocupan algoritmos basados en contenido (para considerar los *reviews*) y los combinan con algoritmos de filtrado colaborativo. Para abordar este problema, en la literatura existen implementaciones tipo cascada, en los que ocupan primero un sistema recomendador y luego refinan los resultados con un segundo sistema [2]. También, se observan implementaciones donde primero ocupan un algoritmo basado en contenido y luego uno de filtrado colaborativo [2, 4, 5], lo que constituye un preproceso de los datos para luego generar recomendaciones. Esto se ha hecho para recomendaciones de ítemes de música [4] y también para recomendar programas de TV [1].

Alternativamente, se ha abordado este tipo de implementación usando primero un algoritmo de filtrado colaborativo y luego uno basado en contenido. Esta es la propuesta explicada en este artículo, la que consiste en generar recomendaciones usando información de *ratings* y de *reviews*, mediante una estrategia de sistema híbrido cuyo primer nivel será un algoritmo de filtrado colaborativo y en el segundo nivel es un algoritmo basado en contenido. Adicionalmente, evaluamos las diversas implementaciones mediante una métrica de penalización que considera la posición de los ítemes en función de interés del usuario.

2 SOLUCIÓN

La solución propuesta consiste en la generación de un sistema híbrido que ocupa primero un algoritmo de filtrado colaborativo y luego uno basado en contenido. La ventaja de este enfoque es que se evita emplear el algoritmo de segundo nivel (basado en contenido) en elementos que ya están bien diferenciados por el primer nivel (de filtrado colaborativo) o que tienen una calificación muy baja. Es importante destacar que las clasificaciones proporcionadas por el recomendador del primer nivel sólo pueden refinarse con la implementación de la estrategia propuesta.

Concretamente, proponemos una arquitectura híbrida para un sistema recomendador de música digital, basado en calificaciones y opiniones de otros usuarios sobre un conjunto de ítemes, ocupando datos de Amazon. La idea consiste en tomar el conjunto de registros para procesarlos con un algoritmo de filtrado colaborativo y con él obtener un subconjunto de ítemes recomendados para cada usuario. Luego, para cada uno de los items recomendados, se genera un listado adicional de ítemes obtenidos mediante una similitud obtenida con un algoritmo basado en contenido. De este modo, se incrementa la experiencia de recomendación mediante un post procesamiento basado en contenido. Esta implementación propuesta, será aplicada en la secuencia ya descrita, la que se muestra también en la Figura 1.

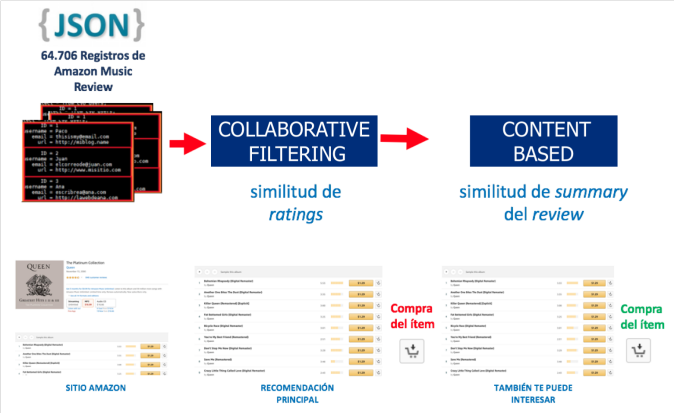

Figura 1: Esquema de solución propuesta.

3 DATASET

Para evaluar la efectividad del esquema propuesto, se ocupó el dataset de Amazon Music Review [8], el que contiene 64.706 registros realizados por 5.541 usuarios sobre 3.568 ítemes. Los atributos que contiene el dataset son:

- *reviewerID*
- *asin* (id de la canción)
- *summary* (resumen del *review*)
- *overall* (*rating* entregado por el *reviewer*)
- *reviewerName* (nombre del *reviewer*)
- *helpful* (indica cuán útil es el comentario, por ejemplo 2/3)
- *reviewText* (texto del *review*)
- *unixReviewTime* (formato unix time del tiempo del *review*)
- *reviewTime* (tiempo del *review* en raw)

Los atributos utilizados como entrada fueron *reviewerID*, *asin*, *summary* y *overall*. Para verificar la utilidad del *dataset* a utilizar, se presentan a continuación las Figuras 2 y 3.

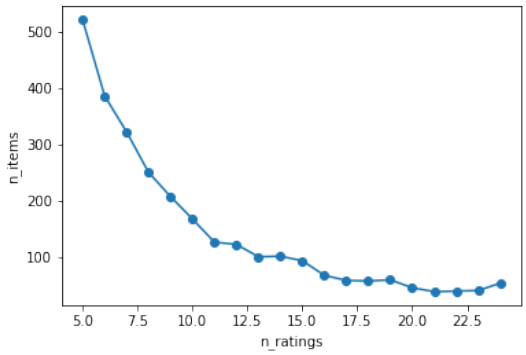

Figura 2: Gráficos nº items vs nº ratings.

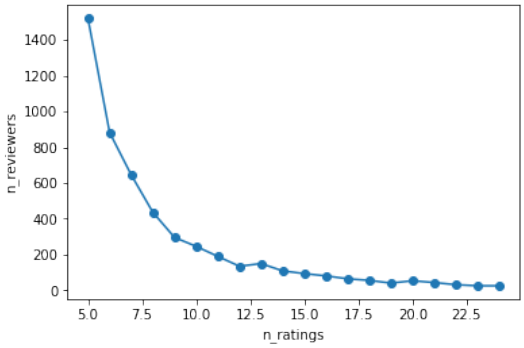

Figura 3: Gráficos nº reviewers vs nº ratings.

De la Figura 2 se puede ver que poco más de 500 ítemes tienen 5 *ratings*. El número de ítemes disminuye a medida que aumenta el número de *ratings* (es decir, los ítemes que tienen pocos *ratings* son muchos, y los ítemes que tienen muchos *ratings* son pocos). Situación similar ocurre con los *reviewers* (Figura 3), donde se puede ver que los *reviewers* que entregan muchos ratings son pocos, y los que entregan pocos *ratings* son muchos. Cabe destacar que por construcción del dataset, se cuenta con a lo menos 5 *ratings* por item y por *reviewer*.

Por lo tanto, respecto de la calidad de los datos, podemos indicar que existen buenas condiciones para realizar el experimento propuesto.

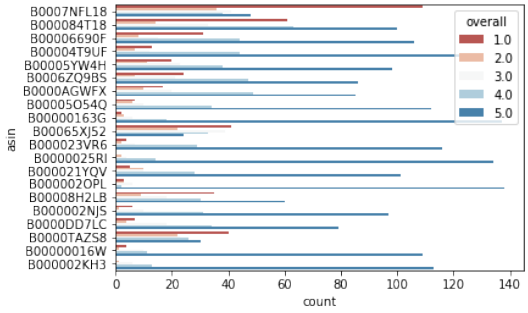

Figura 4: Distribución de ratings por ítem.

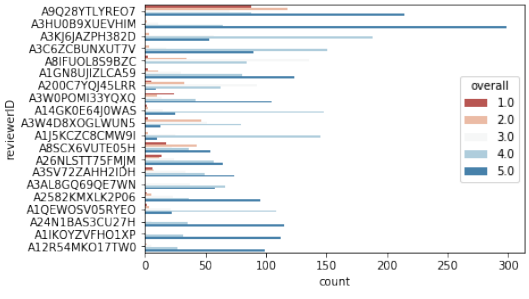

Figura 5: Distribución de ratings por reviewer.

De las Figuras 4 y 5, se puede ver que la frecuencia de los *ratings* con mayores calificaciones positivas es claramente dominante: Es decir, los *ratings* por ítem están más concentrados en aquellos con valores de *rating* = 4 y 5, situación similar a la que ocurre con los *ratings* por *reviewer*.

4 METODOLOGÍA

4.1 Separación del dataset

El dataset original se separó en dos: uno para entrenar los modelos (*training*) y otro para testearlos (*testing*). La separación se hizo ocupando una relación 80-20: en el dataset para *testing*, está el 20% de los ítemes de aquellos usuarios que evaluaron más de 5 ítemes, y en el dataset para *training* está el 80% restante de aquellos *reviewers* que evaluaron más de 5 ítemes, además de los ítemes de los *reviewers* que evaluaron 5 ítemes. Cabe indicar que se probaron diversas proporciones de datos para entrenamiento y test, antes de elegir la que nos entregara mejores resultados, como la relación indicada.

4.2 Métricas

Existen dos enfoques complementarios para evaluar la calidad de las recomendaciones, considerando la recomendación como un producto final de utilidad para el usuario:

- Medir el desempeño de cada uno de los algoritmos recomendadores que forman parte del ensamblaje, y de este modo buscar la mejor combinación según las métricas definidas para ello.
- Utilizar una métrica que evalúe efectivamente la *interacción* de las dos recomendaciones, y considere qué tan interesante resulte esta para el usuario. Este enfoque implica la construcción de una métrica que penalice o pondere el desempeño de las recomendaciones generadas por el algoritmo basado en contenido respecto de la posición de cada ítem recomendado según el algoritmo de filtrado colaborativo

4.2.1 Métricas para la evaluación de algoritmos recomendadores. Para evaluar la calidad de las recomendaciones de los algoritmos de filtrado colaborativo y basados en contenido se usó el MAP (*Mean Average Precision*) y el nDCG (normalized Discounted Cummulative Gain), los que se calculan de la siguiente manera:

$$AP_u = \frac{\sum_{i=1}^{k} Precision@i \cdot relevante(i)}{|relevantes|} \tag{1}$$

$$MAP = \frac{\sum_{u=1}^{n} AP_u}{n} \tag{2}$$

$$DCG = \frac{\sum_{i=1}^{k} 2^{relevante(i)} - 1}{log_2(1+i)} \tag{3}$$

$$nDCG = \frac{DCG}{iDCG} \tag{4}$$

donde k es el largo de la lista de ítemes recomendación, n es el número de *reviewers*, iDCG (*inverse Document Cummulative Gain*) es el cálculo de DCG pero usando la lista de recomendación ordenada al revés y $relevante(i)$ indica si el ítem de la ubicación i en la lista de recomendación es relevante o no.

4.2.2 Métrica para la evaluación del ensamblaje de algoritmos recomendadores. Para evaluar la solución propuesta, construimos una métrica que penalice o pondere el desempeño de las recomendaciones generadas por el algoritmo basado en contenido respecto de la posición de cada ítem recomendado según el algoritmo de filtrado colaborativo. La formulación de la métrica es la siguiente:

$$X = \alpha X_{CF} + (1-\alpha) X_{CB} \tag{5}$$

$$X_{CB} = w_1 X_{CB,1} + ... + w_k X_{CB,k}, w_1 \geq ... \geq w_k \tag{6}$$

donde X corresponde a la métrica usada (MAP o nDCG), α al peso que se entrega a la recomendación del algoritmo de filtrado colaborativo y w_k al peso de la recomendación entregada por el algoritmo basado en contenido y que es la asociada al ítem k de la lista de recomendación obtenida con el algoritmo de filtrado colaborativo.

4.3 Algoritmos recomendadores

Esta sección contiene la definición formal de los algoritmos recomendadores utilizados en este trabajo.

4.3.1 SVD (Singular Value Decomposition). Se descompone la matriz item usuario en 3 matrices: $A = U \cdot S \cdot V$, donde U es la descomposición de vectores de la matriz original, y $S \cdot V$ corresponde a los nuevos ratings [7].

4.3.2 SlopeOne. Se predice un vector de ratings w usando un vector conocido de ratings v, a través de la minimización de la siguiente ecuación:

$$min_b \sum_{i=1}^{n} (v_i + b - w_i)^2 \tag{7}$$

Se usa una ecuación de pendiente uno, $f(x) = x + b$, por lo que en la ecuacion (7) se tiene que $v = x$, $w = f(x) = v + b$ [6].

4.3.3 ALS (Alternating Least Squares). Técnica para resolver modelos de factores latentes a través de la resolución de una matriz de factorización. Una matriz de factorización mapea usuarios e ítemes a un espacio de factores latentes, cuyas interacciones ítem usuario se modelan como producto punto [3]. Una matriz de factorización se resuelve a través del siguiente problema de optimización:

$$min_{q,p} \sum_{(u,i)} (r_{ui} - {q_i}^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{8}$$

donde r_{ui} es el *rating* que el usuario u entrega al ítem i (dato conocido). Se busca determinar q_i y p_u (desconocidos), donde q_i mide el grado en que el item i posee estos factores (positivo o negativo) y p_u mide el grado de interés del usuario en los factores. ALS fija q_i y p_u de manera alternada entre iteraciones, para encontrar los vectores. Cuando todos los p_u están fijos, se recalculan los q_i resolviendo el problema de mínimos cuadrados. Luego, se fijan los q_i y se recalculan los p_u, y así sucesivamente.

4.3.4 tf-idf (term frequency - inverse document frequency). Medida numérica que expresa cuán relevante es una palabra para un documento en una colección. Esta medida corresponde a la multiplicación de 2 términos: El primero es la frecuencia de la palabra t en el documento d ($f_{t,d}$), y la segunda es el logaritmo del número

de documentos n_d dividido por la frecuencia de la palabra t de un documento d en todos los documentos ($df_{t,d}$) [9].

$$tf - idf = f_{t,d} log\left(\frac{n_d}{df_{t,d}}\right) \tag{9}$$

4.3.5 LDA (Latent Dirichlet Allocation). Permite descubrir tópicos en las oraciones que se la dan de *input*. Representa los documentos como un mix de tópicos y determina la probabilidad de que cada palabra influya en la explicación de un tópico.

4.3.6 NMF (Non-negative matrix factorization). Se factoriza una matriz (V) en dos matrices (W y H), donde ninguna de las tres matrices tiene elementos negativos. Genera factores con dimensiones muy reducidas en comparación a la matriz original. Produce un cluster de características: Cada columna de $W \cdot H$ es una combinación lineal de los vectores columna de W (matriz de características) y la transposición de los vectores de H (matriz de coeficientes). El número de columnas (y por lo tanto de características) de W y filas de H se determina manualmente. La factorización de matrices se hace a través de la resolución del siguiente problema de optimización:

$$\min_{W,H} ||\mathrm{V} - \mathrm{W} \cdot \mathrm{H}||_{\mathrm{F}} \tag{10}$$

$$\text{s.a.} W \geq 0 \tag{11}$$

$$X \geq 0 \tag{12}$$

4.4 Especificación de Baselines

Entenderemos por *baseline* al mejor valor obtenido luego de aplicar una métrica sobre un conjunto de algoritmos de referencia. Esto permitirá determinar posteriormente si nuestra propuesta logra un mejor o menor desempeño que dicho valor. Concretamente, aplicaremos las métricas:

- definidas en 4.2.1 sobre cada uno de los algoritmos de filtrado colaborativo: SVD, SlopeOne y ALS.
- definidas en 4.2.1 sobre cada uno de los algoritmos basados en contenido: *tf-idf*, LDA y NMF.
- definidas en 4.2.2 sobre cada una de las implementaciones hídridas descritas en la etapa "Implementación del Híbrido Cascada" de la sección 4.5.

El *baseline* ocupado es el compuesto por el esquema híbrido que usa el mejor algoritmo de filtrado colaborativo y el mejor algoritmo basado en contenido.

4.5 Etapas del experimento

El experimento realizado consideró las siguientes etapas en el proceso de ensamblaje de interacción entre el algoritmo de filtrado colaborativo y el basado en contenido:

(1) *Entrenamiento de algoritmos de filtrado colaborativo*: Se probaron tres algoritmos: SlopeOne, SVD (*Singular Value Decomposition*) y ALS (*Alternating Least Squares*). Para SlopeOne y SVD se ocupó la librería `pyreclab`, mientras que para ALS se ocupó la librería `implicit`. Adicionalmente, se midió el tiempo de ejecución y memoria ocupada con la librería `psutil`.
(2) *Evaluación de la recomendación generada con el algoritmo de filtrado colaborativo*: Se evaluaron las recomendaciones generadas calculando MAP@10 y nDCG@10. El cálculo de MAP@10 y nDCG@10 de SVD y SlopeOne se hizo ocupando `pyreclab` y *relevance threshold = 4*, pues los ratings son muy positivos. Para el caso de ALS, se usó la implementación vista en clases, la que considera que los ítemes relevantes para un *reviewer* son todos los que están en el dataset de *testing*.
(3) *Preparación de los datos de entrada para el algoritmo content-based*: Para este proceso se ocupó como contenido el *summary* de los *reviews*, sobre los cuales realizamos cada una de las actividades que permiten tokenizar el texto: eliminación de las letras capitales y signos de puntuación, para luego separar el texto sin signos de puntuación en tokens, y obtener los tokens stem (raices de cada palabra). Finalmente, se remueven los *stopwords* para crear el diccionario y las *bags of words*. Se ocuparon las librerías `nltk` y `sklearn` para realizar lo señalado.
(4) *Entrenamiento del algoritmo basado en contenido*: los datos generados en la etapa interior sirven de entrada para los algoritmos tf-idf (*term-frequency, inverse document frequency*), LDA (*Latent Dirichlet Allocation*) y NMF (*Non-Negative Matrix Factorization*). Los algoritmos se entrenaron ocupando la librería `gensim`. Para NMF se usó *n components = 10* y para LDA se usó *topic number = 10*. Adicionalmente, se midió el tiempo de ejecución y memoria ocupada con la librería `psutil`.
(5) *Evaluación de la recomendación generada con el algoritmo basado en contenido*: Se evaluaron las recomendaciones generadas calculando MAP@10 y nDCG@10. El cálculo se hizo con la implementación vista en clases, la que considera que los ítemes relevantes para un *reviewer* son todos los que están en el dataset de *testing*.
(6) *Evaluación de la recomendación generada con el esquema propuesto*: se calcula para la recomendación en 2 pasos MAP@10 y nDCG@10 de la forma propuesta en las ecuaciones (5) y (6).
(7) *Implementación del híbrido cascada*: para cada usuario se obtiene una lista de ítemes recomendados (primer nivel, filtrado colaborativo), y para cada ítem de esta lista se recomiendan los ítemes similares a cada uno de ellos, usando el *summary* (segundo nivel, basado en contenido). Consideramos como estrategia la generación de tres implementaciones híbridas que contienen el mejor algoritmo colaborativo en el primer nivel, y cada uno de los tres algoritmos indicadoslas para el segundo nivel.

5 RESULTADOS

5.1 Usando pesos w_k iguales

En esta sección se muestran los resultados obtenidos ocupando $\alpha = 0.5$ y $w_1 = \ldots = w_k = 1/k$, obteniendo los resultados de las tablas 1, 2 y 3.

Tabla 1: Resultados algoritmos de filtrado colaborativo.

Algoritmo	SVD	SlopeOne	ALS
MAP@10	0.00122	0.01516	0.17987
nDCG@10	0.00132	0.01297	0.21308
Tpo. train	5.63	0.46	2.65
Tpo. recom	10.82	16.64	4.94

Tabla 2: Resultados algoritmos basados en contenido.

Algoritmo	tf-idf	LDA	NMF
MAP@10	0.13773	0.08805	0.07617
nDCG@10	0.22079	0.15142	0.13501
Tpo. train	2.52	50.1	5.74
Tpo. recom	344.68	304.66	280.34

Tabla 3: Resultados solución propuesta usando pesos w_k iguales.

Algoritmo	ALS + tf-idf	ALS + LDA	ALS + NMF
MAP@10	0.0992	0.09482	0.09637
nDCG@10	0.12118	0.11491	0.11785

5.2 Usando pesos w_k distintos

En esta sección se muestran los resultados obtenidos ocupando *relevance threshold = 4* en los algoritmos de filtrado colaborativo, *n components = 10* y *topic number = 10* en los algoritmos basados en contenido, $\alpha = 0.5$ y $w_1 = 0.35, w_2 = 0.2, w_3 = 0.1, w_4 = \ldots = w_{10} = 0.05$, obteniendo los resultados de la tabla 4.

Tabla 4: Resultados solución propuesta usando pesos w_k distintos.

Algoritmo	ALS + tf-idf	ALS + LDA	ALS + NMF
MAP@10	0.10038	0.09526	0.09618
nDCG@10	0.12284	0.11536	0.1178

Al observar los resultados indicados en las tablas 1, 2, 3 y 4 se puede constatar que:

- El mejor algoritmo de filtrado colaborativo es ALS, con un nDCG@10 de 21.3%, y un MAP@10 de 17.9%.
- El mejor algoritmo basado en contenido es tf-idf, con un nDCG@10 de 22.1%, y un MAP@10 de un 13.77%.
- El mejor algoritmo basado en contenido, según el consumo de recursos de máquina para tiempo de entrenamiento: tf-idf con 2.52 segundos.
- El mejor esquema híbrido, considerando pesos w_k iguales, es ALS en primer nivel y tf-idf en segundo nivel, con un nDCG@10 de 11.12% y MAP@10 de 10%. De manera similar, el mejor esquema híbrido, considerando pesos w_k distintos, es ALS en primer nivel y tf-idf en segundo nivel, con un nDCG@10 de 12.28% y MAP@10 de 10%.

6 ANÁLISIS DE PARÁMETROS

Para todo el análisis se usó $\alpha = 0.5$ y $w_1 = 0.35, w_2 = 0.2, w_3 = 0.1, w_4 = \ldots = w_{10} = 0.05$.

Usando un *relevance threshold = 3* en los algoritmos de filtrado colaborativo, *n components = 10* y *topic number = 10* en los algoritmos basados en contenido, para la evaluación de la solución propuesta, se obtienen los resultados de la tabla 5.

Tabla 5: Resultados solución propuesta usando *relevance threshold = 3, n components = 10* y *topic number = 10*

Algoritmo	ALS + tf-idf	ALS + LDA	ALS + NMF
MAP@10	0.10031	0.09584	0.09628
nDCG@10	0.12447	0.11783	0.11947

Usando un *relevance threshold = 4* en los algoritmos de filtrado colaborativo, *n components = 7* y *topic number = 7* en los algoritmos basados en contenido, para la evaluación de la solución propuesta, se obtienen los resultados de la tabla 6.

Tabla 6: Resultados solución propuesta usando *relevance threshold = 4, n components = 7* y *topic number = 7.*

Algoritmo	ALS + tf-idf	ALS + LDA	ALS + NMF
MAP@10	0.10031	0.09569	0.09475
nDCG@10	0.12447	0.11724	0.11643

Usando un *relevance threshold = 4* en los algoritmos de filtrado colaborativo, *n components = 13* y *topic number = 13* en los algoritmos basados en contenido, para la evaluación de la solución propuesta, se obtienen los resultados de la tabla 7.

Tabla 7: Resultados solución propuesta usando *relevance threshold = 4, n components = 13* y *topic number = 13.*

Algoritmo	ALS + tf-idf	ALS + LDA	ALS + NMF
MAP@10	0.10031	0.09566	0.09537
nDCG@10	0.12447	0.11735	0.11656

En todas las tablas anteriores, se aprecia que frente al cambio de parámetros por los valores señalados la combinación ALS + tf-idf sigue siendo la mejor.

7 CONCLUSIONES

Evaluamos la interacción de recomendaciones generadas por dos algoritmos de distinto tipo, obteniendo que el mejor esquema de sistema híbrido es el que está compuesto por los algoritmos que tienen mejor desempeño en el primer nivel y en el segundo nivel (en este caso, ALS y tf-idf). Sin embargo, la métrica del esquema híbrido tiene un valor menor que el del mejor algoritmo ocupado en cada nivel. Pudimos ver que el uso de ponderadores distintos es más preciso, ratificándose nuestra intuición de que esta métrica es más adecuada al incorporar los pesos que castigan o premian la posición de los items relacionados, según el interés del usuario. Como trabajo futuro, se propone incorporar el uso de la utilidad del *review* en la recomendación, y añadir al diseño el *feedback* producido por el usuario para adaptar el comportamiento de la recomendación en 2 pasos. Adicionalmente, se pueden incorporar al análisis métricas para medir la diversidad y la novedad de una lista de ítemes recomendados. Es importante destacar que los resultados están condicionados por la calidad del dataset utilizado, por lo que se propone usar el esquema con un dataset donde la presencia de ratings con niveles inferiores sea mayor, y así evaluar el comportamiento de la métrica en otras condiciones. El dataset usado y la distribución de datos para entrenamiento y test disminuyó el efecto de un eventual *coldstart*, pero no logró mejorar el aprendizaje respecto de calificaciones inferiores.

REFERENCES

[1] Ana Belén Barragáns-Martíneza, Enrique Costa-Montenegro, Juan Burguillo, Marta Rey-López, Fernando Mikic-Fonte, and Ana Peleteiro. 2010. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *Information Sciences* 180, 22 (Noviembre 2010), 4290–4311. https://doi.org/10.1016/j.ins.2010.07.024

[2] Robin Burke. 2002. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction* 12, 4 (Noviembre 2002), 331–370. https://doi.org/10.1023/A:1021240730564

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (Agosto 2009), 30–37. https://doi.org/10.1109/MC.2009.263

[4] Aristomenis Lampropoulos, Paraskevi Lampropoulou, and George A. Tsihrintzis. 2012. A Cascade-Hybrid Music Recommender System for mobile services based on musical genre classification and personality diagnosis. *Multimedia Tools and Applications* 59, 1 (Julio 2012), 241–258. https://doi.org/10.1007/s11042-011-0742-0

[5] Aristomenis Lampropoulos, Dionysios Sotiropoulos, and George Tsihrintzis. 2012. Evaluation of a Cascade Hybrid Recommendation as a Combination of One-Class Classification and Collaborative Filtering. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*. IEEE, Atenas, Grecia, 674–681. https://doi.org/10.1109/ICTAI.2012.96

[6] Daniel Lemire and Anna Maclachlan. 2007. Slope One Predictors for Online Rating-Based Collaborative Filtering. (Febrero 2007). https://arxiv.org/pdf/cs/0702144.pdf

[7] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press. https://nlp.stanford.edu/IR-book/

[8] Julian McAuley. 2018. Amazon Product Data. University of California, San Diego. Retrieved 2 de Diciembre de 2018 from http://jmcauley.ucsd.edu/data/amazon/

[9] Michael Pazzani and Daniel Billsus. 2002. Content-Based Recommendation Systems. *The Adaptive Web* 4321 (Noviembre 2002), 325–341. https://doi.org/10.1007/978-3-540-72079-9_10

[10] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to Recommender Systems Handbook. Retrieved 2 de Diciembre de 2018 from http://www.inf.unibz.it/~ricci/papers/intro-rec-sys-handbook.pdf

[11] Spotify, Amherst The University of Massachusetts, and Johannes Kepler University Linz. 2018. Spotify RecSys Challenge 2018. Retrieved 3 de Diciembre de 2018 from https://recsys-challenge.spotify.com/details