Matrix Factorization–User KNN Ensemble Approach for Automatic Playlist Continuation

Thomas Muñoz
Pontificia Universidad Católica de Chile
Santiago, Chile
tfmunoz@uc.cl

Rodolfo Palma
Pontificia Universidad Católica de Chile
Santiago, Chile
rdpalma@uc.cl

ABSTRACT

Automatic playlist continuation is a contemporary problem in music consumption. Since playlists elaboration is more time consuming for the user, playlist continuation is an uprising way of exploring music. Due to this importance, the ACM Recommender Systems Challenge 2018 was focused on evaluating automatic playlists continuation systems, using the Million Playlist Dataset provided by Spotify. In this paper we present a two-stage model towards computing recommendations for a subset of playlists of the dataset. We first focus on a fast retrieval stage that aims to reduce the candidate songs to be searched in the second stage, which reranks these tracks. The final recommendations are an ensemble of this two systems. This simple ensemble system would have reached the 30th place in the challenge main track out of over 100 teams.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Recommender Systems; Automatic Playlist Continuation; Collaborative Filtering; Matrix Factorization; User KNN

1. INTRODUCCIÓN

La relevancia de sistemas recomendadores aplicados sobre el dominio de la música ha experimentado un incremento tanto en la academia como en la industria en los últimos años [1]. Servicios de *streaming* como Spotify, Pandora y Apple Music ponen a disposición catálogos de millones de canciones y es de su interés implementar sistemas recomendadores capaces de elevar el *engagement* de sus usuarios. Es por lo anterior, que en conjunto con la conferencia *ACM RecSys* del 2018, Spotify lanzó un *challenge* dedicado a la tarea específica de continuar automáticamente listas de reproducción.

A diferencia de otros dominios sobre los cuales se pueden aplicar sistemas recomendadores, la música presenta características especiales que vale la pena mencionar. El tamaño de los catálogos de música comercial son del orden de magnitud de las decenas de millones de ítems, mientras que en otros dominios, como por ejemplo las películas, los tamaños de los catálogos son del rango de decenas de miles de ítems. Dado lo anterior, la escalabilidad en sistemas recomendadores de música es un atributo mucho más relevante que en sistemas recomendadores aplicados a dominios más clásicos, como las películas.

Al tomar una relevancia importante la escalabilidad de los sistemas, este trabajo propone un sistema que considera este atributo de calidad mediante la implementación de dos etapas. La primera, basada en factorización de matrices, permite reducir rápida y eficientemente el tamaño del espacio de búsqueda de canciones relevantes. La segunda etapa consiste en un algoritmo de vecinos cercanos, cuyo tiempo de ejecución se ve drásticamente reducido por el filtrado de la primera etapa.

Luego de ejecutar diversos experimentos, se puede observar que el modelo propuesto mejora significativamente métricas, como NDCG, en comparación al *baseline most popular*. Además, se considera que el modelo es competitivo, debido a que si se hubiese enviado al *ACM RecSys Challenge 2018* hubiese obtenido el lugar Nº30 en el *track* principal.

2. *AUTOMATIC PLAYLIST CONTINUATION*

A continuación se definirá la tarea sobre la cuál se aplicará el sistema recomendador. En primer lugar, se define una lista de reproducción como una secuencia ordenada de canciones con el propósito de ser escuchadas en conjunto por algún usuario. La tarea de continuar automáticamente una lista de reproducción (APC, por su sigla en inglés) consiste en ir agregando una o más canciones a una lista de reproducción siguiendo el propósito original de ésta.

Esta tarea está muy relacionada con la generación automática de listas de reproducción (APG, por su sigla en inglés). APC se diferencia de APG en que esta última tarea trabaja sobre listas de reproducción vacías, mientras que la primera puede aprovechar información de las canciones pertenecientes a las listas de reproducción para hacer recomendaciones. Más allá de esa diferencia, la tarea APG se ha caracterizado con conceptos, como *características objetivo* y *conocimiento contextual*, que son útiles para definir APC.

Una forma alternativa de ver la tarea de APC, incorporando nociones de APG, consiste en recomendar una secuencia de canciones que satisfagan *características objetivo* de una lista de reproducción, dado algún *conocimiento contextual* del catálogo de canciones disponible para recomendar. Es requisito del sistema recomendador, identificar correctamente dichas características objetivo y conocimiento contextual. Usualmente, esto se ha realizado mediante técnicas de aprendizaje de máquina.

3. *MILLION PLAYLIST DATASET*

Los experimentos se realizarán sobre el conjunto de datos del *RecSys Challenge* de 2018 de Spotify, llamado *Million Playlist Dataset*, el cual contiene un conjunto de 1.000.000 de listas de reproducción creadas por usuarios de Spotify e incluye títulos de listas de reproducción, listas de reproducción con sus respectivas canciones y otros metadatos. Para tener una idea general de la relación entre listas de reproducción y canciones, se presenta información relevante del *dataset* en el cuadro 1.

Algunas otras características importantes del conjunto de datos se enumeran a continuación:

Dato	Cantidad
Listas de reproducción	1.000.000
Canciones	2.262.292
Álbumes	734.684
Artistas	295.860
Cantidad promedio de canciones por lista	66,34

Cuadro 1: Descripción de datos

- Cualquier lista de reproducción incluida en el *dataset* tiene un título compartido por un número significativo de otras listas de reproducción en el *dataset*.
- Las listas de reproducción contienen al menos 3 artistas únicos y 2 álbumes únicos por lo que no hay listas de un solo álbum o de un solo artista.
- Existe un mínimo global de cantidad de seguidores y escuchadores, es decir, no hay listas de reproducción que no son escuchadas ni seguidas.
- Las listas de reproducción tienen entre 5 y 250 canciones.

Para la implementación, se redujo la información a una matriz de pertenencia R de dimensiones 1.000.000×2.262.292, cuyas entradas r_{pt} son de la forma:

$$r_{pt} = \begin{cases} 1, & \text{si la canción } t \text{ está en la playlist } p \\ 0, & \text{si no} \end{cases}$$

4. MODELO

Se propone un sistema recomendador de dos etapas.

1. **Weight Regulated Matrix Factorization (MF)** [3]: cuyo objetivo principal es reducir rápida y eficientemente el espacio de búsqueda de canciones candidatas para cada *playlist*, y rankear este subconjunto.
2. **User KNN** [2]: cuyo objetivo es *rankear* el subconjunto de canciones entregado por MF.

Finalmente, se combinan los puntajes de ambos métodos y se recomienda una lista ordenada de canciones R para continuar la *playlist*.

4.1. *Weight Regulated Matrix Factorization*

Se busca obtener representaciones latentes de cada lista de reproducción y cada canción. Para esto, se definen las variables:

- x_p es el vector latente que representa a la *playlist* p.
- y_t es el vector latente que representa a la canción t.
- S_{pt}^{mf} es el *score* que tiene la canción t para la lista p.

El objetivo es resolver el siguiente problema de optimización:

$$\min_{x,y} \sum_{p,t} c_{pt}(r_{pt} - x_p^T y_t)^2 + \lambda \left(\sum_p \|x_p\|^2 + \sum_t \|y_t\|^2 \right)$$

Donde c_{pt} es la confianza en alguna observación y se define como $c_{pt} = 1 + \alpha r_{pt}$, que representa la confianza o importancia que se le da a los pares p, t observados en el *dataset*. Esta importancia es cuantificada por α.

Luego, se calcula $S_{pt}^{mf} = x_p^T y_t$ para seleccionar un subconjunto de canciones relevantes para la playlist p, que son las que tienen los mayores *scores*.

4.2. *User KNN*

El objetivo de esta etapa es, para cada *playlist* p, asignar un *score* S_{pt}^{knn} a cada una de las canciones t entregadas por la etapa anterior.

Este puntaje se obtiene comparando qué tanto se parece la lista de reproducción p a las listas p' que contienen a la canción t. En concreto, el *score* de una canción t para una playlists p se obtiene como

$$S_{pt}^{knn} = \sum_{p' \in P(t)} sim(r_p, r'_p) = \sum_{p' \in P(t)} \frac{r_p \cdot r'_p}{\|r_p\| \|r'_p\|}$$

Donde $P(t)$ es el conjunto de listas de reproducción que contienen a la canción t y r_p es la fila p de la matriz de pertenencia R (ver sección 3).

$$p_t = \begin{cases} 1, & \text{si } t \text{ esta en la playlist } p \\ 0, & \text{si no} \end{cases}$$

4.3. *Ensemble*

Finalmente, al unir los métodos descritos anteriormente, para hacer las recomendaciones se realiza un *ensemble* de los *scores* computados por los procesos.

$$S_{pt} = w^{mf} S_{pt}^{mf} + w^{knn} S_{pt}^{knn}$$

Donde w^{mf} y w^{knn} son los pesos que representan la incidencia que tendrá cada *score* en el rankeo final. Estos puntajes se ordenan de manera descendente y se entrega una lista de recomendación para cada lista de reproducción p.

Debido a que los puntajes de los distintos métodos no necesariamente tienen la misma escala y variabilidad, antes de hacer la mezcla se estandarizan los S_{pt}^{mf} y S_{pt}^{knn} de todas las listas de reproducción para que tengan media igual a 0 y desviación estándar igual a 1.

5. MÉTRICAS

Para medir qué tan bien rinde el modelo, se utilizaron las métricas que se describen a continuación. Estas son las utilizadas para evaluar los sistemas implementados en el *RecSys Challenge* de 2018 de Spotify. Se denota G como el conjunto de canciones relevantes para la *playlist* y R la lista de recomendación ordenada.

5.1. *R-precision*

Mide qué tantas canciones relevantes fueron recomendadas para la *playlist*, en términos de cantidad (cardinalidad). Más es mejor.

$$\text{R–precision} = \frac{|G \cap R_{1:|G|}|}{|G|}$$

5.2. NDCG

Mide qué tantas canciones relevantes fueron recomendadas teniendo en cuenta el orden de recomendación. Más es mejor.

$$\text{nDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p}$$

Que para cada *playlist* compara el *Discounted Cumulative Gain* (DCG) con el ideal (IDCG). Sea

$$rel_t = \begin{cases} 1, & \text{si } t \text{ es relevante} \\ 0, & \text{si no} \end{cases}$$

Las cantidades se definen como:

$$DCG = rel_1 + \sum_{i=2}^{|R|} \frac{rel_i}{\log_2 i + 1}$$

$$IDCG = 1 + \sum_{i=2}^{|G|} \frac{1}{\log_2 i + 1}$$

5.3. *clicks*

Dada una lista de reproducción, se define el *click* como la recomendación de 10 canciones para esa lista.

Esta métrica indica cuántos *clicks* son necesarios para encontrar una recomendación relevante. Menos es mejor.

$$\text{clicks} = \left\lfloor \frac{\text{argmin}_i \{R_i : R_i \in G\} - 1}{10} \right\rfloor$$

6. EXPERIMENTOS

A continuación se describirán los experimentos realizados a fin de evaluar el rendimiento del modelo.

Fue necesario particionar y podar la matriz de pertenencia obtenida a partir del *Million Playlist Dataset* para evaluar las recomendaciones hechas por los modelos. Este procedimiento se ilustra en la figura 1.

6.1. *Most popular*

A fin de obtener un *baseline* contra el cual contrastar el rendimiento del modelo, se evaluó el funcionamiento de un sistema recomendador que siempre recomendara continuar una lista de reproducción con las canciones más populares del conjunto de datos. Se define $MP = \{t_1, t_2, \ldots, t_n\}$ como el conjunto de canciones más populares, dónde t_i corresponde a la i-ésima canción con mayor número de pertenencias a listas de reproducción.

Luego, cuando se debe continuar una lista de reproducción p, se le recomienda el conjunto MP.

6.2. *Ensemble*

A continuación se detallará el experimento asociado a la ejecución del modelo *ensemble* propuesto.

Para el entrenamiento del método WRMF de la primera etapa se utilizó la librería `implicit` de Python. En particular, se utilizó el método de optimización denominado *Alternating Least Squares*.

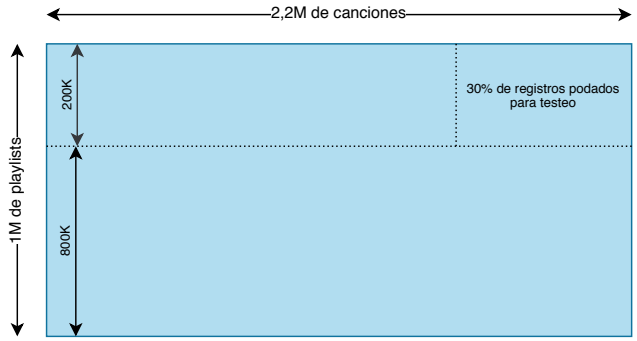

Figura 1: Diagrama con la partición realizada al *Million Playlist Dataset*. A fin de evaluar el rendimiento del modelo, fue necesario podar canciones de algunas listas de reproducción para posteriormente evaluar la relevancia de las recomendaciones realizadas. El procedimiento para realizar la poda fue muestrear de forma uniforme 200.000 listas de reproducción del conjunto de datos, para posteriormente podar de forma uniforme el 30 % de las canciones pertenecientes de cada lista de reproducción muestreada. El conjunto de canciones podadas de la lista de reproducción i-ésima se denota G_i y se considera el conjunto relevante de canciones a recomendar.

Para acelerar el entrenamiento, se entrenó usando una GPU Tesla K80 de forma gratuita sobre la plataforma *Google Colaboratory*.

Los parámetros del modelo utilizados fueron $\alpha = 100$, $\lambda = 0{,}001$ y el número de factores latentes igual a 192. El modelo se entrenó durante aproximadamente 6 horas alcanzando poco más de 1.000 iteraciones de entrenamiento. El entrenamiento fue realizado sobre la totalidad de la matriz podada.

Como se mencionó anteriormente, el objetivo de la primera etapa es rápidamente recuperar un conjunto de canciones candidatas $R^{mf} = \{t_1, t_2, \ldots, t_m\}$ a ser recomendadas sin sacrificar en exceso *recall*. Evidentemente, existirá una correlación entre el tamaño del conjunto de canciones candidatas y el *recall* de este conjunto. Esto se ilustra en la figura 2.

A continuación de seleccionar el conjunto R^{mf}, corresponde pasar a la segunda etapa: *User KNN*. Es en esta etapa donde es relevante el tamaño del filtro usado en la primera etapa, debido a que el tiempo de ejecución de la segunda etapa es directamente proporcional al número de elementos que le llegan.

Finalmente, se calculan los puntajes S_{pt} descritos en la sección de formulación del modelo y se forma el conjunto de recomendaciones $R_p = \{t_1, t_2, \ldots, t_n\}$, donde t_i corresponde a la i-ésima canción con mayor puntaje para la lista de reproducción p. El conjunto R_p se acota a 500 recomendaciones, porque ese es el número de recomendaciones realizadas en el *ACM RecSys Challenge 2018*[1].

Para determinar los valores de los pesos w^{mf} y w^{knn} se realizó una búsqueda en grilla sobre el conjunto $\{0{,}1; 0{,}5; 1; 1{,}5; 2\}$, concluyendo que la combinación de pesos que entrega el mejor rendimiento es $\{w^{mf}, w^{knn}\} = \{2; 0{,}5\}$.

[1]Si se recomienda un número distinto de canciones, las métricas de NDCG y *clicks* dejan de ser comparables con las del tablero del *challenge*.

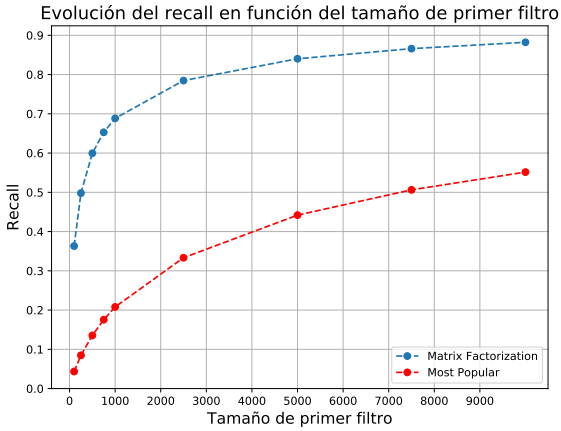

Figura 2: Gráfico de la evolución del *recall* en función del tamaño del primer filtro. Se observa un crecimiento agresivo de la curva asociada al modelo de factorización de matrices. Una decisión de diseño realizada fue seleccionar un tamaño de filtro igual a 5.000 para obtener al menos un 80 % de *recall*. El gráfico también ilustra la superioridad de la factorización de matrices por sobre *most popular* como método para realizar el primer filtro.

Método	R-precision	NDCG	clicks
Most Popular	0.00939	0.05387	28.927
MF	0.133597	0.334477	3.905151
MF + User KNN	0.138598	0.34172	3.66257

Cuadro 2: Resultados

6.3. Resultados

El desempeño de los métodos por sí solos, del *ensemble* y de *most popular* se pueden observar en el cuadro 2.

Se puede apreciar que *most popular* rinde mal en el dominio de *playlists* y canciones. Esto se debe a que hay infinidad de listas que no tienen canciones populares. El mal desempeño se puede aterrizar viendo que en promedio se necesitan 289 canciones recomendadas para encontrar una relevante.

También se ve que el *ensemble* mejora el desempeño con respecto a los métodos por sí solo, y en promedio se necesitan 36 recomendaciones para encontrar una relevante.

Si este modelo se hubiese enviado al *ACM RecSys Challenge 2018*, hubiese obtenido la posición número 30 en el *track* principal. Lo anterior es razonable si se considera el tamaño del conjunto de datos y disponibilidad de recursos computacionales con los que se contaba.

7. CONCLUSIONES Y TRABAJO FUTURO

En primer lugar, se destaca que el enfoque de acotar el espacio de búsqueda mediante técnicas de cómputo más rápido (MF) permite aplicar técnicas tradicionales de filtrado colaborativo a conjuntos de datos muy grandes.

Por otro lado, realizar una mezcla de los métodos de recomendación *User KNN* y factorización de matrices tiene un mejor rendimiento que los métodos por si solos.

Además, cabe destacar que tanto el *ensemble* como los métodos por sí solos superan en rendimiento a *most popular*. Esto se atribuye a que en el dominio de la música, desde el punto de vista de consumo, existen muchas *playlists* para las cuales no es relevante lo más popular. Esto es esperable ya que las listas de reproducción tienden a ser específicas: de ciertos artistas, ciertos géneros, etc.

Es interesante que una mezcla ponderada de puntajes funcione bien y mejore el rendimiento de las recomendaciones, por lo que incorporar otras técnicas a la mezcla puede lograr mejores resultados. Por ejemplo, un sistema que logre capturar la naturaleza secuencial de las listas de canciones es algo deseable, ya que para las listas de reproducción, el orden en que aparecen las canciones tiene cierta connotación en el propósito de la lista.

REFERENCIAS

[1] Schedl, M., Zamani, H., Chen, C., Deldjoo Y. & Elahi, M. *Current Challenges and Visions in Music Recommender Systems Research*, 2018.

[2] Volkovs, M., Rai, H., Cheng, z., Wu, G., Lu Y. & Sanner, S. *Two-stage Model for Automatic Playlist Continuation at Scale*, 2018.

[3] Hu, Y., Koren, Y. & Volinsky, C. *Collaborative Filtering for Implicit Feedback Datasets*, 2008.