Matrix Factorization and Content-based for playlist continuation

Angel Biskupovic, Pontificia Universidad Católica de Chile

***Resumen*—Automatic playlist continuacion (APC) is a contemporary problem in music consumption. SInce making playlist takes more time for the user, playlist continuation is a revolutionary way to explore music. Because of this importance, the 2018 ACM Recommendation Systems Challenge focused on evaluating automatic playlist continuation systems, using the Million Playlist dataset provided by Spotify. In this article, I present a two-stage model for calculating recommendations for a subset of playlist in the dataset. The first stage focuses on a quick selection of candidates, using the BPR method, in this way, there are 5000 song selected. The second stage consists of the evaluation of these 5000 songs, for this an embedding is used that containss the metada delivered in the dataset, and when comparing these 5000 song with the songs that have been heard in the playlist, the 10 that will be in the final recommendation.**

I. Introducción

Con el paso de los años, los sistemas de recomendación aplicados al dominio de la música han ido tomando más relevancia en la academia y en la industria. Existen diversos servicios de streaming (Spotify, Apple Music, Pandora, entre otros) que ponen a disposición catálogos de millones de canciones ya que es de su interés implementar sistemas recomendadores capaces de elevar la fidelidad de los usuarios y de atraer nuevos usuarios a sus plataformas. 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 (APC).

La tarea de continuación automática de la lista de reproducción (APC) consiste en agregar una o más pistas a una lista de reproducción de una manera que se ajuste a las mismas características de destino de la lista de reproducción original. Esto tiene ventajas tanto para escuchar como para crear listas de reproducción: los usuarios pueden disfrutar escuchando sesiones continuas más allá del final de una lista de reproducción de duración finita, al mismo tiempo que les resulta más fácil crear listas de reproducción más largas y atractivas sin necesidad de tener una gran familiaridad musical. Una gran parte de la tarea de APC es inferir con precisión el propósito previsto de una lista de reproducción determinada. Esto es un desafío no solo por la amplia gama de estos propósitos previstos (cuando incluso existen), sino también por la diversidad en las características subyacentes o características que podrían ser necesarias para inferir esos propósitos. La tarea de APC también puede beneficiarse potencialmente de la elaboración de perfiles de usuario, por ejemplo, haciendo uso de listas de reproducción anteriores y el historial de escucha a largo plazo del usuario.

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, donde se utiliza especificamente Bayesian Personalized Ranking. La segunda etapa consiste en un algoritmo basado contenido, donde se extrae un embedding de palabras de cada canción, utilizando metadatos como nombre de artista, nombre de canción, track uri, etc, reduciendo drasticamente su tiempo de ejecución.

II. Modelo Implementado

Se propone un modelo que consiste en dos etapas:

- **Weight Regulated Matrix Factorization**: El objetivo de esta etapa, es reducir el espacio de búsqueda de manera rápida y eficiente, de manera de obtener un set de canciones candidatas de cada playlist que pasarán a la segunda etapa.
- **Content-Based** : Una vez obtenidas las canciones candidatas, se procede a evaluar estas con un método basado en contenido, para esto se obtiene un embedding de texto, sacado de los metadatos entregados, y se mide la similaridad de las canciones candidatas con las canciones que están en la playlist, y de esto se selccionan las 10 mejores.

II-A. Primera Etapa: Weight Regulated Matrix Factorization

Se busca obtener las representaciones latentes de cada lista de reproducción y de cada canción. El problema de predecir $\hat{x_{ui}}$ puede verse como una matriz $X : UXI$. Con la factorización matricial, la matriz objetivo X se aproxima

por el producto de la matriz de dos matrices de rango bajo $W :\| U \| xk$ y $H :\| I \| xk$

$$\hat{X} := WH^t \tag{1}$$

donde k es la dimensionalidad / rango de la aproximación. Cada fila wu en W puede verse como un vector de características que describe a un usuario u y, de manera similar, cada fila hi de H describe un elemento i. Por lo tanto, la fórmula de predicción también se puede escribir como:

$$\hat{x_{ui}} =< w_{uf} * h_{if} >= \sum_{i=1}^{k} w_{uf} h_{if} \tag{2}$$

Los parámetros del modelo para la factorización matricial son θ = (W, H). Los parámetros del modelo también pueden verse como variables latentes, modelando el gusto no observado de un usuario y las propiedades no observadas de un artículo.

II-B. Segunda Etapa: Based-Content

Para la segunda etapa, se utilizan los metadatos entregados tales como: nombre de artista, nombre de canción, albúm, track uri. Con esta información, se arma un embedding de texto para cada una de las canciones. El modelo utilizado para esta tarea se llama “bags of word”, que consiste en mapear cada palabra de un vocabulario como un vector codificado. La primera etapa del modelo consiste en la detección y descripción de los puntos de interés, donde el tipo de detector utilizado es cuadrícula regular, que es uno de los más simples y eficaces que podemos utilizar a la hora de detectar características. La imagen se divide en partes iguales y estas partes son los puntos de interés. La única limitación que hay, es que utiliza poca información de la imagen. Lo siguiente, una vez extraídos los puntos de interés de la imagen, con el descriptor se debe elegir la parte más importante del vector de características. Un buen descriptor debe tener la habilidad de controlar la intensidad, la rotación, la escala y las variaciones de la misma dimensión cuando el orden de los diferentes vectores no importa. Algunos descriptores locales e individuales se extraen con el descriptor Scale Invariant Feature Transform (SIFT). Para finalizar, se encuentra el histograma de frecuencia de las palabras,hay que crear el histograma para saber qué es el objeto que se quiere identificar. Para cada imagen que se quiere identificar, previamente se habrá utilizado un algoritmo de aprendizaje con el que se habrán guardado las características más importantes de las imágenes de entrenamiento en una bolsa. Una vez se ha terminado la clasificación de las imágenes de entrenamiento, se realiza el cálculo del histograma para cada imagen que se quiere reconocer. En el eje ”X.es”tarán las características extraídas de las imágenes de entrenamiento y en el eje ”Y.el” número de veces que aparece cada característica en la imagen a analizar. La siguiente figura muestra la transformación de texto en vector:

Figura 1: Texto transformado a embedding

Una vez obtenidos lo embedding de cada canción, se ve la similitud que existe entre las cancones candidatas y las canciones de la playlist a la que se va a recomendar. Para esto se utiliza la similaridad coseno que viene definida por:

$$sim(V_i, V_j) = cos(V_i, V_j) = \frac{V_i * V_j}{\|V_i\|\|V_j\|} \tag{3}$$

Una vez obtenida la similaridad entre las canciones, se va a utilizar como score la siguiente ecuación:

$$score = max\{sim(V_i^X, V_j^X)\} \tag{4}$$

Una vez obtenido el score, estos se ordenan de manera de seleciconar los top-K que serán los que finalmente serán recomendados.

III. Métricas de evaluación

Para le evaluación del rendimiento del modelo, se utilizaron las métricas que serán descritas a continuación. Se denota como $\mathcal{G}$ el conjunto de canciones relevantes para la playlist, en términos de cantidad.

III-A. NDCG

Esta métrica mide qué tantas canciones relevantes fueron recomendadas teniendo en cuenta el orden de recomendación. Mientras más canciones sean relevantes, mejor.

$$nDCG_p = \frac{DCG_p}{IDCGp} \tag{5}$$

Para cada playlist, se compara el valor de DCG con el DCG ideal $IDCG_p$. Lo siguiente es definir los items relevantes, esto es:

$$rel_t = \begin{cases} 1 \; si \; t \; es \; relevante \\ 0 \; si \; no \; es \end{cases} \tag{6}$$

Las cantidades vienen definidas por:

$$DCG = rel_t + \sum_{i=2}^{R} \frac{rel_i}{log_2 i + 1} \tag{7}$$

$$IDCG = 1 + \sum_{i=2}^{G} \frac{1}{log_2 i + 1} \tag{8}$$

III-B. R-precision

Mide la cantidad de canciones que fueron relevantes para la playlist. Al igual que el caso anterior, mientras mas canciones relevantes hayan, mejor R.precision se obtiene.

$$R - precision = \frac{\| G \cap R_{t:\|G\|} \|}{\| G \|} \tag{9}$$

III-C. MAP

El objetivo es eliminar el error en los primeros elementos en lugar de mucho más tarde en la lista. Para ello, necesitamos una métrica que pondera los errores en consecuencia. El objetivo es ponderar mucho los errores en la parte superior de la lista. Luego, disminuya gradualmente la importancia de los errores a medida que bajamos los elementos inferiores de una lista.

La métrica de predicción promedio (AP) intenta aproximarse a esta escala móvil de ponderación. Utiliza una combinación de la precisión en las sublistas sucesivas, combinada con el cambio de recuerdo en estas sublistas. El cálculo es el siguiente:

$$AveP = \frac{\sum_{k=1}^{n}(P(k)xrel(k))}{Numero_de_canciones_relevantes} \tag{10}$$

IV. Experimentos

Primero se mostrarán las características que tiene el dataset del Challenge 2018. Este contiene un total de 1000000 de playlist credas por usuarios de spotify, cada uno tiene sus respectivas canciones y metadatos. La siguiente tabla muestra una relación e información relevante del dataset:

Datos	Cantidad
Listas de reproducción	1.000.000
Canciones	2.262.292
Albumes	734.684
Artistas	295.860
Cantidad promedio por lista	66.34
Numero total de canciones	66.346.428

Hay que recalcar otras caracteristicas de este dataset, existe un mínimo global de cantidad de seguidores y gente que escucha esta lista, por lo tanto, no hay playlist sin seguidores ni que no son escuchadas. Las listas de reproducción tienen entre 5 y 250 canciones.

Para esto se creo una matriz que contiene la información relevante, esta matriz es de orden 1.000.000 x 2.262.292 cuyas entradas vienen definidas por:

$$r_{pt} = \begin{cases} 1 \ si \ cancion \ esta \ en \ playlist \\ 0 \ si \ no \ es \end{cases} \tag{11}$$

IV-A. Baselines

Como baselines para comparar, fueron utilizados los siguientes métodos:

- ALS con gradiente conjugado: Se utilizó este método que consiste en obtener una matriz playlist x canciones, y de está manera obtener las interacciones del usuario (playlist) con los items (canciones). Es un método basado en implicit feedback.
- BPR: Este método es el empleado en el modelo final, al igual que el caso anterior, es un modelo basado en implicit feedback, en este caso se seleccionan directamente las 10 canciones a recomendar.

IV-B. Desarrollo

Lo primero fue dividir el dataset en entrenamiento y validación (o test), para esto se dividio en 800.000 playlist para el set de entrenamiento y 200.000 para el set de test, en esta primera etapa aún se encuentran las 2.262.292 de canciones, la figura 2 muestra la distribución implementada.

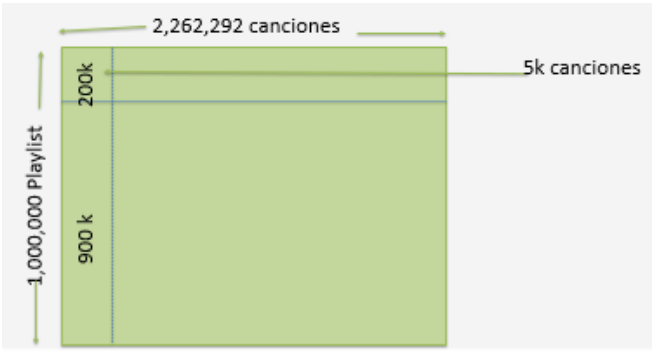

Figura 2: Separación dataset

Una vez dividido el dataset, se procedio a hacer la evaluación con BPR, para esto se tuvo que seleccionar los mejores parámetros de este algoritmo, para esto se vario la cantidad de factores latentes entre [25,50,100,150,200,250], lo siguiente fue variar el learning rate entre [0.001,0.002,0,01,0,02] y se realizó el mismo procedimiento con la regularización variando entre los mismo valores que en el learning rate. La siguiente tabla muestra los valores utilizados para el modelo:

Parámetro	Valor
Factores	250
Learning Rate	0.01
regularización	0.02

Lo siguiente fue seleccionar el tamaño del primer filtro, para esto se probó con diferentes valores [1000,2500,5000,10000] obteniendo el mejor resultado con 5000.

También en paralelo, se obtuvo el embedding de cada una de las canciones, se probó con diferentes metadatos, finalmente la entrada de texto tuvo la forma [artista, nombre cancion, track uri] que obtuvo mejores resultados, lo siguiente fue seleccionar el tamaño del vector, para esto se probó con tamaños de vectores de [50,100,200], seleccionando un vector de tamaño 100. Para esta parte del proceso, fue utilizado la libreria word2vec de python.

Una vez obtenidos los embedding, lo que se hace es comparar los embedding del filtro de 5000 canciones con los embedding de las canciones que se encuentran en la playlist a recomendar. Como se mencionó anteriormente se obtiene el score de cada una de estas 5000 canciones y se selccionan las 10 con score más alto, que serán finalmente las canciones recomendadas. Una vez realizado todo este proceso, se procede a evaluar las recomendaciones.

Para el procesamiento de los datos fue necesario utilizar un PC con una RAM de 30 GB, la etapa de procesamiento

de datos y de separación de dataset tomó alrededor de 2 horas. La obtención de los embedding de las canciones tomó alrededor de 1.5 horas, y la evaluación del modelo completo (BPR+Word2vec) tomo alrededor de 3 horas. Por lo tanto el procesamiento y obtención de resultados duro un total de aproximadamente 7 horas.

IV-C. Resultados obtenidos

Como se mencionó anteriormente, para le evaluación del modelo se utilizaron tres métricas de desempeño, y se comparó el modelo con dos modelos basados en implicit feedback. Los parámetros utilizados en estos modelos se pueden observar en la siguiente tabla:

Modelo	Factores	Learning rate	Regularización
ALS	25	n/a	0.02
BPR	250	0.01	0.02

Finalmente los resultados obtenidos se pueden observar en la tabla que se muestra a continuación:

Método	NDCG	R-precision	MAP
ALS	0.2425	0.1436	0.03296
BPR	0.2423	0.1415	0.03298
BPR + content-based	0.3348	0.1754	0.03891

En los resultados mostrados, se puede observar que la combinación del método BPR con content-based obtiene un mejor resultado que BPR por si solo, lo que demuestra su efectivad, y además la cantidad de tiempo que se reduce al comparar 5000 canciones en vez de 2.262.292 si se hubiese utilizado solo content-based, ya que al ser solo 5000 canciones ya requeria de bastante tiempo, con más de 2 millones el tiempo podría haber aumentado considerablemente.

V. Conclusión

Lo primero que se debe destacar de este método propuesto, es la capacidad de reducir el espacio de búsqueda de manera muy rápida y eficiente mediante el uso del método BPR, que son técnicas basadas en feedback implícito, que tienen una respuesta rápida (a comparación de otro métodos) a dataset muy grandes. Además, esto permiten aplicar otro tipo de técnicas como filtrado colaborativo o basadas en contenido que requieren un computo mucho mayor y pueden no funcionar en un dataset muy grande.

Se utilizó un técnica basada en contenido para la segunda etapa, que permitió utilizar los metadatos entregados en el set de datos, y encontrar ciertas similaridades entre características de las canciones, como artista o nombre canción, teniendo un buen resultado.

Los resultados obtenidos finalmente muestran que se pudo mejorar la calidad de la recomendación utilizando un tiempo menor. Si se compara el NDCG con el del ganador del challenge estos obtuvieron un valor de 0.3766, lo que no es una gran diferencia con el método propuesto acá. Como trabajo futuro, para otras personas que cursen esta asignatura, se podría utilizar un ensamble de técnicas como basada en contenido en conjunto con User-KNN, o se puede considerar integrar información temporal que se intento en este proyecto, pero por temas de tiempo no se logró.

VI. Bibliografia

Referencias

[1] Muñoz, Thomas; Palma, Rodolfo., *Matrix Factorization User KNN Ensemble Approach for Automatic Playlist Continuation.*

[2] Cedric De Boom, Rohan Agrawal, Samantha Hansen, Esh Kumar, Romain Yon, Ching-Wei Chen, Thomas Demeester, and Bart Dhoedt, *Large-scale user modeling with recurrent neural networks for music discovery on multiple time scales.*

[3] Volkovs, Maksims; Rai, Himanshu; Cheng, Zhaoyue; Wu, Ga; Lu, Yichao; Sanner, Scott., *Two- stage model for automatic playlist continuation at scale.*

[4] Zamani, Hamed; Schedl, Markus; Lamere, Paul; Chen, Ching Wei., *An analysis of approaches taken in the ACM Recsys challenge 2018 for automatic music playlist continuation*