Prueba de diferentes métodos en el Spotify Million Playlist Dataset Challenge

Nicolás Guzmán, Diego Jiménez, Valentina Núñez, Alonso Zamorano

Resumen: El presente trabajo aborda el desafío ”Spotify Million Playlist Dataset Challenge” con el objetivo de realizar un análisis exhaustivo de diversos modelos, de manera de encontrar cuales producen mejores resultados, y conseguir información relevante respecto a las caracteristicas del dataset. Al momento de realizar las pruebas, destacó el rendimiento superior del modelo GRU4Rec en comparación con otros enfoques, indicando una estructura secuencial en las playlists. Se destaca la importancia de gestionar eficientemente el escalado del modelo para conjuntos de datos más grandes y se señala la necesidad de explorar estrategias para integrar información de contenido en el modelo.

1. INTRODUCCIÓN

Spotify es un servicio de transmisión en línea que da acceso a escuchar música, pódcasts y videos digitales (“¿Qué es Spotify? - Spotify.” s.f.). Dentro de spotify los usuarios pueden reproducir una gran variedad de canciones, álbumes, y también pueden crear sus propias playlists. Los usuarios de spotify han creado más de 4.000 millones de listas de reproducción, las cuales pueden estar agrupadas por alguna categoría como género, artista, año, ocasión, entre otros.

Para facilitar la creación de listas de reproducción y así ayudar a las personas a encontrar la música que les gusta, se les entregan recomendaciones a los usuarios sobre qué canciones agregar a su playlist. Por esto es que spotify lanzó el desafío “Spotify Million Playlist Dataset Challenge” cuyo objetivo es desarrollar un sistema que permita considerar las características de la lista de reproducción y entregue una lista de recomendaciones que se puedan agregar a la playlist en cuestión (Chen et al., 2018).

Abordar este desafío es relevante porque para las empresas como spotify, es valioso que puedan realizar recomendaciones certeras a sus usuarios para mantenerlos felices, aumentando su retención y engagement. Además, el desarrollo de un sistema de este estilo puede extrapolarse a otros ámbitos en el futuro, como lo es la recomendación de ítems en e-commerce, sitios de viajes, etc. Por lo mismo, la creación de una metodología novedosa para la recomendación de ítems dado una lista de estos puede ser muy beneficiosa en múltiples rubros.

2. ESTADO DEL ARTE

El ACM RecSys Challenge 2018 en sí ya ha terminado, recibiendo en total 1467 entregas de código. Para medir el rendimiento se ocuparon las métricas de R-Precision, con un valor más alto de 0.2241, además de NDCG, con un valor más alto de 0.3946, y un click más bajo de 1.7818 (Zamani et al., 2018). Los valores más altos de R-Precision y NDCG fueron obtenidos por el mismo equipo, mientras que el click más bajo lo obtuvo otro equipo.

En el tiempo que se realizó este desafío, todavía no se utilizaban tecnologías que ahora son más comunes, como los Transformers. Estos aparecieron en 2017 como mecanismos basados en la atención para hacer traducción, prescindiendo de recurrencia y circunvoluciones. Hoy en día se usan para tareas como recomendar noticias, hasta recomendaciones socio-secuenciales que consideran influencias sociales. Así, su uso en sistemas recomendadores se ha vuelto común, especialmente en lo que respecta al procesamiento del lenguaje. Por ejemplo, NVIDIA creó "Transformers4Rec"(2021), un modelo que combina NLP con RecSys para recomendación secuencial y de sesión.

De esta manera, se busca utilizar estas nuevas tecnologías en este desafió, para encontrar nuevos métodos de recomendación que puedan ser más adecuados para resolver el problema.

3. DATASET

El dataset proporcionado por Spotify para el desafío Spotify Million Playlist Dataset Challenge 2018 consta de un conjunto de 1 millón de listas de reproducción públicas de Spotify, que cuentan con más de 2 millones de canciones únicas de casi, 300000 artistas. Las listas de reproducción fueron creadas por usuarios de Spotify entre el 2010 y el 2017.

Estas están conformadas por el título de la lista, las canciones de la lista y metadata de la lista. Las canciones a su vez cuentan con información del nombre de la canción, nombre del artista, nombre del álbum, la duración de la canción, y la posición en la que está dentro de la lista. El dataset fue creado utilizando información pública de personas de Estados Unidos mayores de 13 años de edad. Cada playlist contiene al menos 5 pistas y no más de 250 pistas, con al menos 3 artistas únicos y 2 álbumes únicos.

Por temas de capacidad de cómputo no se pudo utilizar el dataset completo, por lo que solamente se tomó una fracción de 5000 playlists con 106997 canciones y 24792 artistas. Esta fracción del dataset utilizado se divide en dos dataset, el de entrenamiento que corresponde al 85 % de los datos que contiene un total de 4250 playlists y el de testeo que corresponde al 15 % de los datos que contiene un total de 750 playlists.

4. METODOLOGÍA

Para procesar los datos, se obtienen todas las canciones únicas existentes en todas las playlists del dataset, y se les asigna un id único de manera aleatoria. Luego se hace lo mismo con los id de las playlists, y se genera una lista para cada playlist con los id de sus canciones en orden. De esta manera, se obtiene un dataframe con todas las canciones y otro dataframe con todas las playlists, con una lista de canciones para cada playlist.

Para modelos que necesitan un set de entrenamiento secuencial, se genera secuencias iterativamente con las canciones de cada playlist, que se encuentran ordenadas, obteniendo de esta manera múltiples sets de secuencias y valores a predecir por cada playlist.

Al momento de evaluar los modelos, se extrae la última canción de las playlists del dataset de testear, y se realiza una recomendación de un set de canciones (tamaño depende de K de NDCG) con los modelos para verificar si logran predecir esta canción que fue extraída. Las métricas de evaluación usadas son NDCG@5, NDCG@10, NDCG@20, NDCG@50, NDCG@100 y R-precision.

Como línea base se corren los modelos de Most Popular y Random para tener un punto inicial de comparación de los valores obtenidos de NDCG y R-precision de los modelos probados. Dentro de los modelos utilizados se tiene:

4.1. Transformers

Se generan embeddings de las canciones, usando el modelo pre-entrenado de "sentence-transformer"llamado *all-mpnet-base-v2* que está basado en el *mpnet_base* de Microsoft. Cada canción fue trabajada como una concatenación del nombre de la canción, nombre del artista, álbum y duración separados por espacios. Una vez creados los embeddings de cada canción, se crea un embedding de la playlist a la que se quiere recomendar con el promedio de los embeddings de las canciones que contiene. Se ocupan dos tipos de similaridad:

Similaridad playlist: busca las playlists más similares con similaridad coseno, y luego busca la canciones que más se repiten en estas playlists encontradas.
Similaridad canción: Se compara el embedding promedio de la playlist con el embedding de las canciones y busca las más similares.

4.2. *GRU4Rec*

Este modelo corresponde a una RNN que se utiliza por lo general para recomendaciones secuenciales y de sesion. Esta se origina del paper 'Session-based recommendations with recurrent neural networks' (Hidasi et al., 2016). Se probó con 2 y 3 capas recursivas con diferentes valores de unidades, llegando al mejor valor de 256 unidades por cada capa. Finalmente, se utilizó la función de pérdida de *categorical crossentropy*.

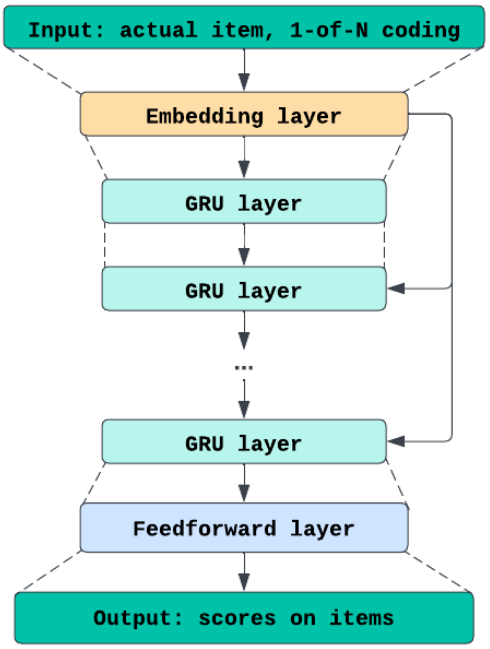

Figura 1: *Arquitectura de GRU4Rec*

4.3. *LSTM*

Una LSTM o Long Short-Term Memory es una red neuronal recurrente que se especializa en procesar y predecir secuencias de datos. A diferencia de las redes neuronales tradicionales, la LSTM incorpora conexiones que la retroalimentan, permitiéndole procesar secuencias enteras de datos y no solo datos individuales, lo que hace que sea altamente eficaz para comprender y predecir patrones de datos secuenciales (Saxena, 2023) . Funciona de forma similar al método de GRU4Rec, reemplazando las capas de GRU por LSTM.

4.4. *Factorización matricial*

La factorización matricial es aquel proceso en el que se transforma una matriz en un producto de matrices de menor tamaño. Su uso está ligado a la reducción de dimensionalidad, reducción de cómputo en operaciones matriciales, entre otros. En este caso se utiliza la matriz de interacción entre playlists y canciones ocupando ALS para realizar la factorización.

4.5. *CNN (1D)*

Una CNN o red neuronal convolucional es un tipo de red neuronal que utiliza kernels para extraer información y patrones en secuencias de 1, 2 y hasta 3 dimensiones.

En este caso, se utiliza para buscar patrones dentro de la secuencia de un embedding promedio de una playlist (esto es, el promedio de los embeddings de sus canciones). Se ocupan 4 capas convolucionales con 512, 256, 128 y 64 filtros respectivamente. Además se alternan capas de max pooling para disminuir el ruido en las salidas de cada capa. Por último, se predice un embedding de la misma dimensión que la entrada para luego "buscar" las canciones más parecidas (distancia coseno) al embedding predicho.

4.6. *KNN*

El k-nearest neightbors algorithm es un modelo de clasificación de aprendizaje no paramétrico supervisado que usa la proximidad para hacer clasificaciones y predicciones sobre la agrupación de datos de un dato individual ("What is the k-nearest neighbors algorithm?", s.f.). Se ocupa la matriz de interacción entre las canciones (ítems) y las playlists (usuarios) como vector binario, usando similaridad coseno como métrica de distancia.

5. RESULTADOS Y ANÁLISIS DE PARÁMETROS

5.1. *Clustering*

El clustering se ocupa para separar en grupos un conjunto de datos no etiquetados, clasificando datos similares en un mismo conjunto o cluster. Se pensaba utilizar este método para disminuir la cantidad de neuronas en la capa densa final de las redes neuronales, que deben tener el tamaño de la cantidad de canciones (106997) si no existe una agregación preexistente como el clustering. Sin embargo, este método no entregó buenos resultados, lo que se puede apreciar en la figura 2 dónde no hay clusters bien definidos, sino que se encuentran mezclados, lo que implica que no hay una semejanza específica entre las canciones

(Kaushik, 2023). Además, las canciones dentro de los clusters no presentaron una semejanza notoria, estaban más bien "repartidos al azar".

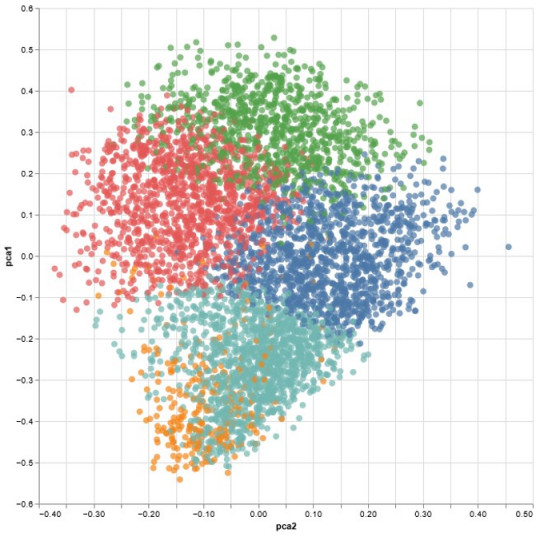

Figura 2: *Cluster de canciones en PCA de 2 dimensiones*

También se le hizo el análisis de Silhoutte para intentar encontrar un número óptimo de clusters. El Silhoutte o coeficiente de silueta es una medida de que tan similar es un dato dentro de un cluster en comparación con otros clusters. El valor del coeficiente de silhoutte se mueve entre -1 y 1, siendo 1 la mejor puntación que indica que los datos son similares dentro de su cluster y están lejos de otros clusters, 0 indica que los clusters están superpuestos (Banerji, 2023). En este caso, se puede apreciar en la figura 3 que los valores de silhouette score son muy cercanos a 0 y van disminuyendo con el aumento del valor de K. Esto significa que la agrupación de datos es más bien aleatoria o que los datos son muy parecidos entre sí. Tomando esto en cuenta, se llegó a la decisión de no utilizar este método.

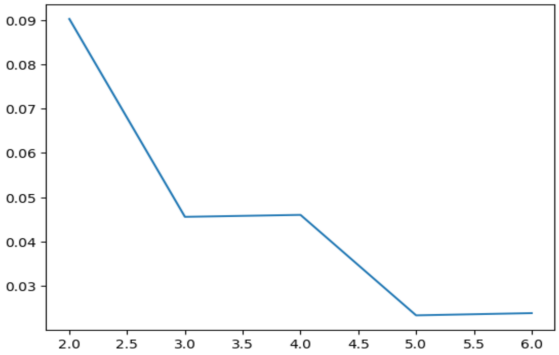

Figura 3: *Silhouette score para distintos K en Kmeans*

5.2. Resultados

Luego de haber entrenado los modelos con el set de entrenamiento, ya sea con las playlists entregadas directamente o modificadas como secuencias para GRU, se realizó la evaluación con el set de testeo respecto a NDCG@5, NDCG@10, NDCG@20, NDCG@50, NDCG@100 y R-precision. Para cada modelo se escoge la variación de parámetros que obtienen los mejores resultados, correspondiendo al promedio de los valores de NDCG@K. Estos resultados se pueden apreciar en la tabla 1 en la página 5.

Playlist corresponde a la similaridad de playlists con los embeddings de transformer, y *Song* corresponde a la similaridad entre el embedding de playlist y las canciones. *GRU* corresponde al modelo GRU4Rec con embeddings generados automáticamente por la capa Embedding de tensorflow, y *GRU+T* corresponde al modelo GRU4Rec con los embeddings personalizados obtenidos con Transformer.

Como se puede observar en la tabla 1, el modelo que consiguió el mejor resultado corresponde a *GRU* con 1 capa de 256 unidades, con 5 épocas de entrenamiento. Esta presenta valores significativamente más altos que el resto de modelos, en una magnitud cercana al triple, como se puede comprobar al observar su valor de NDCG@5 (0.012239), respecto al siguiente más alto (0.004841).

El modelo con el 2° mejor resultado corresponde a *GRU+T*. La utilización de estos embeddings personalizados del Transformer llevó a peores resultados que respecto a los embeddings generados automáticamente con los id de canciones en la secuencia. Esto nos puede indicar que la integración directa de estos embeddings al modelo GRU4Rec no corresponde a la mejor solución, y que si se quisiera utilizar contenido de las canciones o playlists para disminuir los proble-

Tabla 1: *Mejores resultados para cada modelo de NDCG y R Precision*

Model	Parameters	NDCG@5	NDCG@10	NDCG@20	NDCG@50	NDCG@100	RPrecision
Popular		0.001200	0.001600	0.001900	0.003700	0.008500	0.000000
Random		0.000000	0.000000	0.000000	0.000000	0.001000	0.000000
KNN	Neighbors: 3	0.003300	0.003800	0.006500	0.012600	0.016600	0.001300
MF	Fac: 5, Reg: 0.01, Iter: 10	0.001900	0.001900	0.002300	0.004900	0.007900	0.001300
Playlist	n_playlists: 2	0.003900	0.004300	0.006300	0.011000	0.015500	0.001300
Song		0.003300	0.004200	0.005500	0.006400	0.007700	0.002700
GRU	Layers: 1	0.012239	0.014346	0.018676	0.024123	0.028692	0.008000
GRU+T	Layers: 2	0.004841	0.005761	0.007671	0.010689	0.014576	0.004000

mas provocados por utilizar solamente la interacción de ítems y usuarios (como es el caso del cold start, por ejemplo), sería mejor a través de la integración de modelos, en vez de simplemente entregar el contenido como embedding.

Los modelos en 3° y 4° lugar corresponden a KNN con 3 vecinos y la similaridad coseno entre playlists respecto a los embeddings del Transformer (*Playlist*). Estos, a pesar de que KNN corresponde a collaborative filtering y *Playlist* corresponde a content-based filtering, consiguieron valores similares para NDCG y R-Precision, con el promedio de KNN siendo levemente superior. Esto nos dice que los atributos intrínsecos de las canciones entregan información útil respecto a las posibles interacciones que tienen con ciertas playlists, por lo que su utilización es viable.

Respecto a los valores obtenidos con LSTM, estos fueron prácticamente iguales a los obtenidos con GRU4Rec al momento de entrenar, por lo que se decidió no incluirlos en la tabla, asumiendo que estos, al ser ambos recomendadores secuenciales, capturan la misma información de la playlist de forma similar, y que por lo mismo, las posibles razones de como funcionan se pueden explicar con solo uno de los modelos.

Por otro lado, al momento de correr CNN, no se obtuvieron buenos resultados, llegando a valores de 0 para todos los valores de NDCG@K y R-Precision, por lo que no se ponen estos resultados en la tabla.

5.3. Análisis de Parámetros

Respecto al comportamiento de los modelos al cambiar los parámetros, se hizo un análisis para los 4 primeros lugares. En ciertos casos, como para *GRU* con 3 capas, el modelo no presentaba buenos resultados al momento de ser entrenados, incluso empeorando época a época. Estos no se terminaron de entrenar debido al tiempo que significaría su proceso completo y como empeoraban época a época no tenía sentido entrenarlos, por lo que no aparecen en los gráficos.

5.3.1. GRU y GRU+T

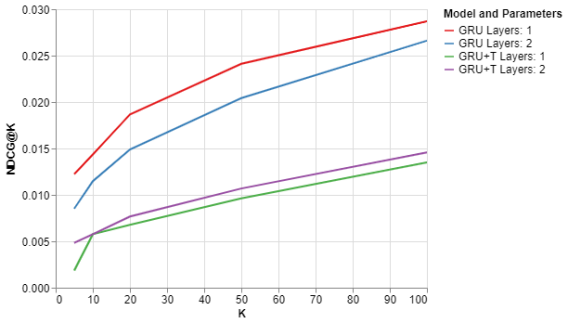

Figura 4: *Valores de NDCG para GRU y GRU+T con distinta cantidad de capas*

En la figura 4 se muestra las diferencias entre los resultados al cambiar la cantidad de capas de GRU, además del uso de embeddings personalizados o no, para GRU4Rec. En este gráfico, se puede ver que los valores de la GRU sin embeddings personalizados son mejores que su contraparte, para toda posible cantidad de capas. GRU sola es aproximadamente 2x mejor que el modelo con embeddings personalizados, indicando que estos embeddings empeoran el modelo significativamente. Esto puede ser debido a que se utiliza un Transformer preentrenado y no especializado en esta tarea, por lo que los embeddings de canciones no se encuentran tan separados entre sí, y pueden presentar ruido. De esta manera, se introduce un cierto nivel de confusión al utilizarlo, causando la disminución de calidad del modelo. Tomando esto en consideración, el realizar un fine-tuning previo del Transformer podría entregar mejores resultados.

Por otra parte, el mejor modelo para *GRU* solo se obtiene con 1 capa, mientras que *GRU+T* obtiene su mejor valor con 2 capas. Esto puede ser causado por el mayor nivel de confusión que significa el uso de los embeddings personalizados. Al haber una menor diferenciación entre los embeddings, y posiblemente más ruido, se necesita una mayor cantidad de capas para extraer la información latente necesaria para recomendar, por lo que para *GRU+T*, el modelo con 2 capas tiene mejores resultados.

Respecto a la cantidad de capas, se puede ver que es un número reducido, solamente 1 o 2. Esto puede explicarse debido a que el dataset utilizado es relativamente pequeño, por lo que no es necesario un nivel de profundidad alto. Esto significa que si se intentara usar este modelo con el dataset completo, seguramente sea necesario aumentar la cantidad de capas, lo que también aumentaría el poder de cómputo y memoria necesario.

5.3.2. KNN

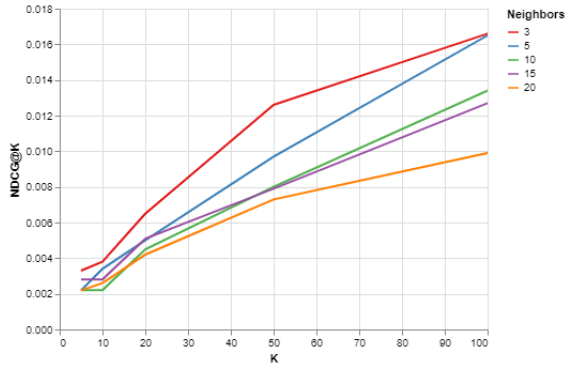

Figura 5: *Valores de NDCG para KNN con distinta cantidad de neigbhors*

La figura 5 muestra como cambia el resultado del modelo KNN respecto a la cantidad de vecinos que se utilizan. Como se puede observar, al disminuir la cantidad de vecinos, aumenta el valor de NDCG en la mayoría de los casos. El modelo con solamente 3 vecinos presenta el valor más alto de NDCG@K para todos los valores de K, y el resto de valores de vecinos se entrecruzan parcialmente con valores más bajos de K, pero vuelven al orden para valores más altos de K.

Esto podría significar que las playlists presentan nichos especialmente pequeños y especializados, de manera que un número más alto de vecinos generaliza demasiado las canciones recomendadas. Tomando esto en cuenta, un número menor de playlists corresponde a la mejor opción para este modelo.

Aun así, esto podría verse alterado al aumentar la cantidad de playlists utilizadas, es decir, al aumentar el tamaño del dataset. Al haber una mayor cantidad de datos, cada nicho estaría más poblado y se podría extraer información de más vecinos. Esto es importante de considerar al momento de escalar este tipo de modelos para una mayor cantidad de datos.

5.3.3. Playlist

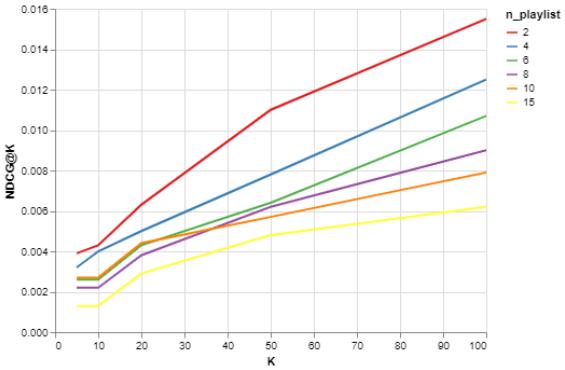

Figura 6: *Valores de NDCG para similaridad de playlist con Transformers con distinta cantidad de playlists*

La figura 6 muestra el resultado de utilizar la similaridad coseno entre playlists con los embeddings de canciones obtenidos con el Transformer. Debido a que con este modelo se extrae las canciones más repetidas en las playlists vecinas, el aumentar la cantidad de estas alterara igualmente la calidad del modelo.

Como se puede observar, el modelo disminuye sus valores de NDCG al aumentar la cantidad de playlists vecinas utilizadas. Así, el modelo con mejores resultados corresponde al que utiliza solamente 2 playlists, mientras que el peor corresponde al de 15 playlists.

Al igual que en el caso de KNN, existe un cierto nivel de entrecruzado para los modelos con valores de K menores, pero al aumentar estos valores la disminución respecto a la cantidad de vecinos se hace más explícita.

Esto puede ser causado por lo mismo que altera los valores de KNN respecto a la cantidad de vecinos. Al haber nichos muy específicos de playlists, el aumentar la cantidad de estas utilizadas para obtener canciones significa una sobre-generalización, causando que disminuya la calidad de las recomendaciones. Por lo mismo, un modelo con menos cantidad de playlists es

mejor en este caso.

6. CONCLUSIONES

Una vez realizada la investigación y análisis, se puede afirmar que GRU fue el modelo con mejores resultados, seguido por GRU con embeddings personalizados (embeddings del Transformer), KNN y la similaridad entre playlists respecto a los embeddings del Transformer. El estudio tenía el objetivo de poder comparar diferentes tipos de modelos para sacar conclusiones respecto del problema y de las necesidades de este. Dado que el modelo con mayor rendimiento fue el GRU4Rec, se puede inferir que existe un tipo de información secuencial del que está aprendiendo la red neuronal, por lo que esta obtiene los mejores resultados. Así, los modelos secuenciales podrían ser más favorables al momento de intentar hacer recomendación de continuación de playlists en este caso.

Por otra parte, podemos concluir que la limitación del dataset perjudicó en los resultados del modelo. El dataset utilizado para evitar problemas de cómputo consideraba un 0.5 % de las playlists del dataset original, además de un 5.3 % de las canciones. La eliminación permitió ejecutar el modelo secuencial sin mayores problemas, sin embargo, la pérdida de este volumen de data genera una perdida de generalidad del modelo y, por tanto, del rendimiento. Así, se puede afirmar que este tipo de solución al problema tiene un costo de cómputo elevado por la alta cantidad de información que procesa, por lo que se debería buscar una forma de escalar este modelo para el procesamiento de más datos de manera eficiente, de manera que se pueda utilizar el dataset completo de playlists.

Finalmente, se encontró que los modelos que utilizaban los embeddings obtenidos con el Transformer llegaban a resultados similares a otros métodos, como KNN, que corresponde a collaborative filtering. A partir de esto, se puede concluir que los datos intrínsecos de las canciones presentan información valiosa que logra indicar ciertas interacciones entre playlists y canciones. De esta manera, su posible inclusión en sistemas más complejos de recomendación podría mejorar los resultados, y reducir problemas existentes en modelos dependientes de las interacciones, como el problema de cold start.

7. TRABAJO FUTURO

Se ha obtenido una prueba de que el uso de GRU obtiene buenos resultados respecto a la problemática de Playlist Continuation al compararlo con otros métodos. Se puede buscar mejorar el sistema recomendador planteado, variando la estructura de la red neuronal, o realizando modelos compuestos con otras soluciones. El utilizar información de contenido más explícitamente en el modelo puede aumentar la calidad de los resultados, además de reducir problemas de cold start, por lo que la integración con un modelo content-based, como el realizado con Transformers, podría conseguir buenos resultados.

Otro punto a considerar es la limitación del dataset propuesta por nosotros. En principio, la reducción del tamaño de los datos permitió ejecutar sin problemas de cómputo todos los entrenamientos y testeos. Sin embargo, no se puede ignorar que se eliminó gran parte del dataset original para poder realizar este experimento. Por lo tanto, con una mayor capacidad de cómputo, se podría considerar utilizar una mayor cantidad de datos, de manera que sea factible utilizar un modelo secuencial. Con este tipo de dispositivos se deberían conseguir valores más realistas e incluso se podrían conseguir mejores resultados con base en los cambios hechos.

Por último, se podría realizar mejoras de los embeddings creados para las canciones y playlist. Se debería investigar nuevos modelos de Transformer que podrían obtener mejores resultados con este dataset, y se debería realizar un fine-tuning de estos modelos a las concatenaciones obtenidas de las canciones para conseguir mejores resultados. Se podría alterar la información que es entregada en la concatenación, e incluso se podría recopilar información adicional que capture otros aspectos intrínsecos de cada canción, como su ritmo, género, ánimo, u otros aspectos que ya son utilizados por Spotify para sus recomendaciones. Una mejor representación del embedding permitiría generar relaciones más reales entre elementos y, por lo tanto, se conseguirían mejores resultados a la hora de recomendar.

REFERENCIAS

¿Qué es Spotify? - Spotify. (s.f.). https : / / support . spotify.com/es/article/what-is-spotify/

Banerji, A. (2023). K-Mean: Getting the Optimal Number of Clusters. https : / / www .

analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/

Chen, C., Lamere, P., Schedl, M., & Zamani, H. (2018). Recsys challenge 2018: automatic music playlist continuation. https://dl.acm.org/doi/abs/10.1145/3240323.3240342

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). Session-based Recommendations with Recurrent Neural Networks.

Kaushik, S. (2023). *Agrupación - Introducción, diferentes métodos y aplicaciones*. https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/#What_Is_Clustering?

Saxena, S. (2023). *What is LSTM? Introduction to Long Short-Term Memory*. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

What is the k-nearest neighbors algorithm? (s.f.). https://www.ibm.com/topics/knn

Zamani, H., Schedl, M., Lamere, P., & Chen, C. (2018). An Analysis of Approaches Taken in the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation. *CoRR*, *abs/1810.01520*. http://arxiv.org/abs/1810.01520