Tecnológico de Monterrey

Análisis de grandes volúmenes de datos (Gpo 10)

Equipo 24

Sistemas de Recomendación - Implementación final del sistema de recomendación.

Luis Salomón Flores Ugalde - A00817435
Oscar Israel Lerma Franco - A01380817
Alejandro Guzmán Chávez - A01795398

16/6/2024

En esta entrega es necesario realizar un reporte donde se enlisten los siguientes aspectos:

- **Implementación** final de sistemas de recomendación. Integra la evidencia en GitHub de los algoritmos desarrollados en los avances 4.2 y/o 6.2.
- **Evaluación** integral del desempeño de los modelos utilizando varias métricas. Recuerda integrar la evidencia en el repositorio GitHub del equipo.
- **Documentación** del código base y algoritmos implementados. Entregar en el documento word/pdf en Canvas.

Criterio	Valor
Portada con datos completos de los integrantes	5
Implementación final de sistemas de recomendación. Integra la evidencia en GitHub de los algoritmos desarrollados en los avances 4.2 y/o 6.2.	35
Evaluación integral del desempeño de los modelos utilizando varias métricas. Recuerda integrar la evidencia en el repositorio GitHub del equipo.	35
Documentación del código base y algoritmos implementados. Documento word/pdf.	25
Total	100

Implementación

Mejoras al método SVD

En el transcurso del proyecto, se notó varias deficiencias al algoritmo utilizado por SVD.

Como habíamos propuesto en la entrega anterior, realmente se iba a trabajar sobre el método de coseno para mejorarlo ya que el SVD simplemente no daba resultados con la esparcidas de datos que se tenían al tomar en cuenta solamente los juegos 'recomendados' de los usuarios.

Al notar esto, se decidió hacer un acercamiento diferente y más a fondo en sentido de extracción e interpretación de datos.

En la versión anterior, hicimos caso omiso a una fuente de datos de usuarios de Australia que tomaba en cuenta la librería completa de los jugadores pero no tenía realmente mucha más información más que 'playtime'.

En esta entrega surgió una idea. La cual respondía la pregunta de que es el gusto de un jugador.

Primero tenemos que entender nuestro *dataset*:

- No tenemos 'ratings' como tal, pero tenemos información importante, que es interactividad.
- Un jugador puede tener cientos de juegos en la librería pero se recomienda 1 solo.
- Un jugador puede tener miles de horas en diferentes juegos pero recomendar 1 que haya jugador menos de 10 horas más.
- A un jugador que tiene tiempo reciente jugado vale más que tiempo en historial.
- Independientemente del tiempo de juego, un juego recomendado vale mucho para un jugador.

Para estos puntos llegamos a ciertas áreas de mejora, en específico en la vectorización de 2 elementos: el tiempo de juego; todo el tiempo ("*playtime_forever*"), y 2 semanas ("*playtime_2weeks*").

El primer problema es la alta discrepancia entre los datos ya que pueden tener valores extremos en el dataset, lo que "crea distancia" entre vectores y hace difícil la recomendación. (Lectores recordarán las métricas del modelo anterior de SVD).
Puedes tener 40 juegos con menos de 4 horas y luego tener dos con 30 mil horas (hay casos así)

Para resolver lo anterior, se utilizó el siguiente *pipeline*:

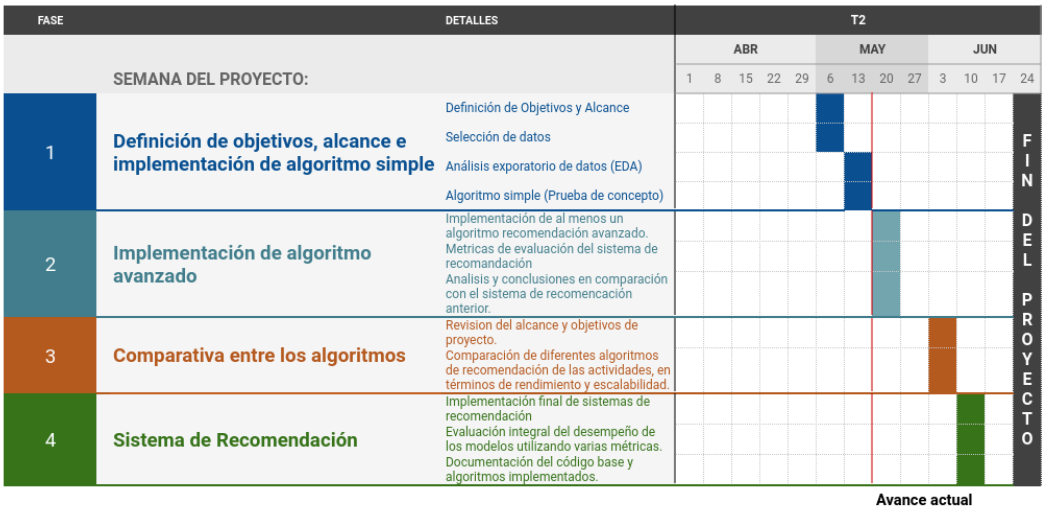
- Normalización en base a logaritmo para *playtime_forever*, y base sigmoid para *playtime_2weeks* con un máximo de 10 (totalmente a nuestra discreción)
- Transformador MinMax en base de rango de (0 a 5) para ambas propiedades.
- Crear la columna *combined_rating* (clasificación combinada), basado en una media ponderada de 60% para *playtime_2weeks* y 40% *playtime_forever.* (Estos valores una vez más son a nuestra idea de que es lo importante)
- Finalmente, este valor de 'combined_rating' se multiplica por el valor recomendado (1+recommend). Esto con el motivo de darle un valor mucho mayor a los items recomendados. Pues sabemos por los datos, es algo muy extraño para un juegor, muy rara vez hacen una recomendación. como el ejemplo que mencionamos anteriormente.

En este último esfuerzo que mencionamos y para volver a explicarlo, es por que se notó que jugadores con gran cantidad de horas no necesariamente recomiendan el juego, y no necesariamente recomiendan ningún juego. Es por esto que se aplica la columna '*weighted_rating',* en la cual duplica el valor del *rating* en caso que recomiende el juego. A tal magnitud creemos que vale tener el 'tag' de 'recommend'.

Estos cambios fueron aplicados en base a experimentación, con base a métricas resultantes y conocimiento "experto" de los desarrolladores.

Cronograma del Avance 4

TÍTULO DEL PROYECTO	Implementación de modelo de recomendación	EMPRESA	INSTITUTO TECNOLÓGICO DE MONTERREY
RESPONSABLE DEL PROYECTO	Equipo 24	FECHA	09/06/24



Evaluación de algoritmos:

Métrica	Método del Coseno	*Nuevo* Método de SVD[1]
Precision@K	0.176965	**0.775621**
Recall@K	**0.647169**	0.407023
MAP@K	0.307603	**0.866197**
NDCG@K	0.411096	**0.516954**
MRR@K	0.388815	**0.979523**
RMSE	*~.89 (añadido en doc)*	**0.650721**
Tiempo Entrenamiento	**2.993277**	8.410636
Tiempo Recomendación	**79.673120**	1699.10935

Entre las métricas de precisión utilizadas:

- **Precisión@K**: Mide el porcentaje de ítems recomendados entre los primeros K elementos relevantes. Cuando se prefiere calidad de recomendaciones a su cantidad.
- **Recall@K**: Mide el porcentaje de elementos relevantes encontrados entre los primeros K recomendados. Evalúa cuántos elementos relevantes son descubiertos por el sistema. Se calcula como el número de ítems relevantes en las primeras K posiciones dividido por el número total de ítems relevantes.
- **MAP@K** (Mean Average Precision at K): Promedio de las precisiones calculadas en cada posición de un ítem relevante hasta K, promediado sobre todos los usuarios o consultas. Considera la posición de los ítems relevantes y su cantidad, dando una medida del rendimiento de la clasificación.
- **NDCG@K** (Normalized Discounted Cumulative Gain at K): Evalúa la clasificación de los ítems recomendados dando más importancia a los aparecen en las posiciones superiores de la lista de recomendaciones.
- **MRR@K** (Mean Reciprocal Rank at K): Es el promedio del recíproco de la posición del primer ítem relevante en la lista de recomendaciones, calculado

[1] Véase Github Notebook.

sobre todas las consultas. Cuando sólo importa la posición del primer ítem relevante.

- **RMSE** (Root Mean Squared Error): proporciona una medida de la magnitud media del error en las valoraciones predichas. Un RMSE más bajo indica una mayor precisión del sistema de recomendación a la hora de predecir las valoraciones de los usuarios.
- **Tiempo de entrenamiento:** tiempo transcurrido de entrenamiento del modelo (en segundos).
- **Tiempo Recomendación:** tiempo transcurrido para recibir la recomendación del modelo, de @K (en segundos). Tenemos que tomar en cuenta que estos datos son el tiempo total de la recomendación de todos los usuarios para K

Resultados y análisis

Con los valores obtenidos anteriormente, observamos que el SVD no solo superó al método de Coseno en todas las métricas excepto *Recall.* Podemos deducir que:

- **Precisión y calidad**: El método SVD supera ampliamente al método Coseno en términos de precisión, MAP, NDCG y MRR. El método SVD ahora proporciona recomendaciones más precisas y mejor clasificadas.
- ***Recall***: El método Coseno tiene un recall más alto, lo que sugiere que recupera más elementos relevantes en general, pero con menos precisión (orden “no relevante”).
- **RMSE**: El método SVD tiene un RMSE más bajo, lo que indica una mayor precisión en la predicción.
- **Eficacia de tiempo**: El método Coseno es *significativamente* más eficiente tanto en tiempo de entrenamiento como en tiempo de generación de recomendaciones. El método SVD, aunque es más preciso, requiere una cantidad impresionante de tiempo de procesamiento.

En nuestros resultados o recomendaciones por usuario, las más similares en contenido son las de SVD e implicitly mientras que las recomendaciones hechas por el método de coseno son completamente diferentes. Aunque en estilo son similares. Esto lo atribuirías a que está hecho con el contexto y es por esto que es el ‘estilo’ de juego o contenido el cual es similar. Aunque sí es muy diferente. Esto tampoco quiere decir que es ‘malo’ solo diferente.

Resultados con *Feedback* implícito

Se trabajó y se terminó el algoritmo de *feedback* implícito sobre nuestro dataset, sin embargo se decidió no continuar con el esfuerzo al enfocarse en el algoritmo SVD por la precisión observada. Por ello no se agregó las pruebas requeridas de métricas, una muestra del output se muestra en la siguiente sección.

Observamos que correr las recomendaciones de *Implicitly* tomó: 9290.773648023605 segundos. Esto es 9 veces más que lo que tomó el nuevo método mejorado de SVD.

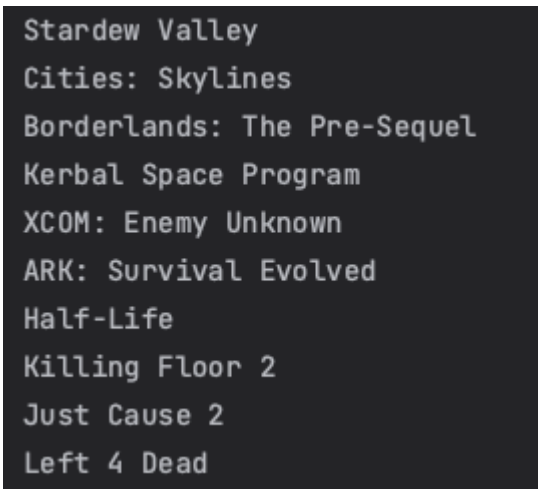
En la siguiente entrega se busca obtener las métricas finalizadas para agregarse en nuestra tabla comparativa.

Muestra **SVD @10** usuario ‘evcentric':

10 rows × 16 columns pd.DataFrame ↗ CSV

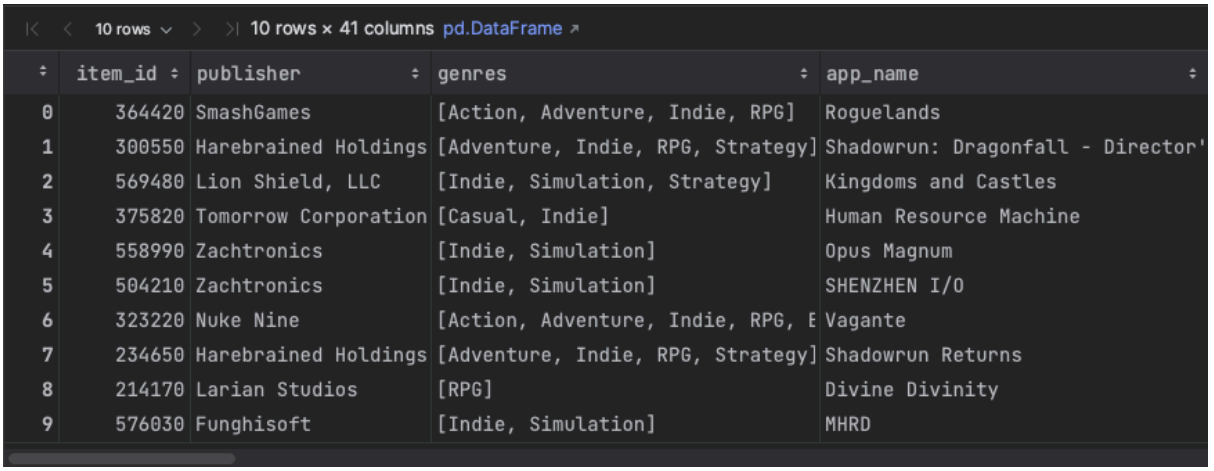
	item_id	app_name	genres	predicted_rating
0	230410	Warframe	[Action, Free to Play]	4.038160
1	211820	Starbound	[Action, Adventure, Casual, Indie, RPG]	3.566726
2	49520	Borderlands 2	[Action, RPG]	1.358669
3	377160	Fallout 4	[RPG]	1.341019
4	22380	Fallout: New Vegas	[Action, RPG]	1.336249
5	8980	Borderlands	[Action, RPG]	1.325703
6	218230	PlanetSide 2	[Action, Free to Play, Massively Multiplayer	1.284814
7	212680	FTL: Faster Than Light	[Indie, Simulation, Strategy]	1.254712
8	220	Half-Life 2	[Action]	1.069524
9	730	Counter-Strike: Global Offensiv	[Action]	1.061462

Muestra **Implicity@10** ‘evcentric':



```
Stardew Valley
Cities: Skylines
Borderlands: The Pre-Sequel
Kerbal Space Program
XCOM: Enemy Unknown
ARK: Survival Evolved
Half-Life
Killing Floor 2
Just Cause 2
Left 4 Dead
```

Muestra **Cosine@10** ‘evcentric':



10 rows × 41 columns pd.DataFrame

	item_id	publisher	genres	app_name
0	364420	SmashGames	[Action, Adventure, Indie, RPG]	Roguelands
1	300550	Harebrained Holdings	[Adventure, Indie, RPG, Strategy]	Shadowrun: Dragonfall - Director'
2	569480	Lion Shield, LLC	[Indie, Simulation, Strategy]	Kingdoms and Castles
3	375820	Tomorrow Corporation	[Casual, Indie]	Human Resource Machine
4	558990	Zachtronics	[Indie, Simulation]	Opus Magnum
5	504210	Zachtronics	[Indie, Simulation]	SHENZHEN I/O
6	323220	Nuke Nine	[Action, Adventure, Indie, RPG, E	Vagante
7	234650	Harebrained Holdings	[Adventure, Indie, RPG, Strategy]	Shadowrun Returns
8	214170	Larian Studios	[RPG]	Divine Divinity
9	576030	Funghisoft	[Indie, Simulation]	MHRD

Tabla comparativa

Aspecto	*Nuevo* SVD	Coseno
Principio	Descomposición matricial de usuarios-item	Similitud del coseno entre vectores
Ventajas	Maneja datos escasos, reducción dimensional	Simplicidad, escalabilidad
Desventajas	*Muy* Costoso computacionalmente, re-calculación	No captura relaciones complejas
Casos de uso	Plataformas de streaming, e-commerce	Recomendaciones de artículos, filtrado colaborativo simple

Documentación

Notebook directo:
https://github.com/VF1Gimure/MNA_GVD24/blob/main/Equipo_user_items_SVD_PLUS.ipynb
Notebook de *Feedback Implícito*:
https://github.com/VF1Gimure/MNA_GVD24/blob/main/Implicity_WIP_2.ipynb

Carpeta del proyecto: https://github.com/VF1Gimure/MNA_GVD24/tree/main

Bibliografía

1. Punia, S. K., Kumar, M., Stephan, T., Deverajan, G. G., & Patan, R. (2021). Performance analysis of machine learning algorithms for big data classification: Ml and ai-based algorithms for big data analysis. International Journal of E-Health and Medical Communications (IJEHMC), 12(4), 60-75.

2. Kang W., McAuley J. (*ICDM*, 2018) **Self-attentive sequential recommendation.** UC San Diego pdf

3. Zhang, Q., Lu, J., & Jin, Y. (2021). Artificial intelligence in recommender systems. Complex & Intelligent Systems, 7(1), 439-457. https://link.springer.com/content/pdf/10.1007/s40747-020-00212-w.pdfLinks