Sistema Recomendandor de Juegos para la plataforma STEAM

José Guzmán
Pontificia Universidad Católica
San Joaquín, Santiago
jaguzman6@uc.cl

Germán Cheuque
Pontificia Universidad Católica
San Joaquín, Santiago
gacheuque@uc.cl

ABSTRACT

El mundo de los Videojuegos se ha diversificado mucho durante los últimos anõs, haciendo que aumente drásticamente la cantidad de usuarios pertenecientes a comunidades dedicadas a esta área de entretención, números que se traducen en ganacias de la industria. Este contexto hace que sea relevante implementar sistemas recomendadores dentro de esta área, que cumplan con los requerimientos de los distintos usuarios y que tomen en cuenta las características propias de la industria, como la gran cantidad de videojuegos que salen cada año. En este trabajo se probó el potencial de varios modelos basados en Máquinas de Factorización (FM), redes neuronales profundas (DeepNN) y uno derivado de la mezcla de ambos (DeepFM), escogidos por su potencial de recibir múltiples variables de entrada. Uno de estas variables corresponde al sentimiento derivado a partir de reviews realizadas por los usuarios. Se evaluaron tomando en cuenta la precisión de recomendación y la diversidad/novedad de la lista recomendada. Todos los algoritmos probados superan al modelo ALS usado como baseline. El algoritmo con mejor desempeño resulta ser DeepNN demostrando que las interacciones de orden superior son más importantes en esta tarea de recomendación. El sentimiento actuó como ruido debido a la baja densidad de reviews existente en el conjunto de datos. En este sentido, el algoritmo DeepNN logra mejoras de 0.04% en MAP@10 y de un 5.9% en Novelty al no considerar sentimiento. Se propone que los algoritmos probados se benefician del embedding en donde se agrupan distintos parámetros para darles igual importancia.

KEYWORDS

Sistemas Recomendadores, Factorization Machines, Deep Neural Networks, Deep Factorization Machines, Novelty, Diversity

ACM Reference Format:
José Guzmán and Germán Cheuque. 2018. Sistema Recomendandor de Juegos para la plataforma STEAM. In *Proceedings of ACM Conference (RecSys).* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCCIÓN

De acuerdo al European Mobile Game Market, en 2016 alrededor de 2.5 billones de personas dedicaban parte de su tiempo a jugar videojuegos [1], convirtiendose así en uno de los pasatiempos más comunes en el globo, que atrae cada día a nuevos adeptos ofreciendo nuevas experiencias con cada título que sale al mercado.

Hace un par de décadas muy pocas personas hubieran apostado que invertir en videojuegos pudiera ser una buena idea. Contrario a dicha opinión, el alto número de adeptos que los videojuegos han ido captando hace que hoy las grandes empresas dedicadas a esta industria estén dentro de algunas de las mejores valoradas economicamente en el mundo. A nivel global, en 2017 la industria de videojuegos creció un 10.7% [4], logrando ganancias de $116 billones de dólares. Ya este año, de acuerdo al Newzoo's Global Games Market Report [8] los jugadores del mundo han gastado alrededor de $138 billones de dólares, cifra que ya es mayor a las ganancias del año anterior y que sólo puede crecer en lo que resta de éste. Las principales tiendas de aplicaciones para smartphones también dan cuenta de una consistencia en estos números, ya que, para el 2018 Google Play y la App Store de iPhone registran un crecimiento de 10.3% y 14.1% [12] respectivamente a igual fecha del año anterior.

Una de las razones de estos grandes número es la diversificación que la industria ha tenido últimamente, en cuanto a las múltiples plataformas de juego, a la categoría o género de los videouegos, a la introducción del factor *social* y también al posicionar una nueva idea de deporte competitivo ligada al uso de la tecnología (e-sports). Esta adaptabilidad de la industria hace que una gran gama de ítemes con diferentes atributos y posibilidades, llegue a los usuarios y despierte en ellos el interés de hacerse parte de la comunidad. Por su parte la plataforma Steam, empresa muy reconocida dedicada a la distribución digital de videojuegos, ha sabido aprovechar este fenómeno de diversificación para incrementar sus ganancias, ofreciendo a sus usuarios una infinidad de opciones de compra y muchas otras funcionalidades dentro de la plataforma. Dicha compañía cifra en 10 millones el número de personas que ingresan a sus servidores cada hora a jugar videojuegos, interactuar socialmente con otros jugadores, comprar nuevos productos de entretenimiento, entre otras.

Sin embargo, el hecho de poseer una oferta tan variada y gigantesca de productos representa un problema para los jugadores al momento de elegir un nuevo item que les pueda gustar, por lo que se hace nescenario crear mecanismos que los ayuden a encontrar los productos que se ajusten a sus intereses. [14].

Por esta razón, y por los precedentes del crecimiento de la industria presentados, se hace relevante la existencia de buenos sistemas recomendadores que puedan dar al usuario una rápida muestra de los nuevos lanzamientos y aquellos que realmente puedan ser de su gusto. Lograr desarrollar un algoritmo que cumpla esta función podría generar numerosos beneficios económicos para la industria, una mejor experiencia para el usuario e inclusive poder conocer la tendencia de preferencias, cosa que sirva a los desarrolladores para trabajar sobre nuevas ideas y temáticas con un éxito asegurado. Además de los beneficios obvios que el desarrollo de un sistema

RecSys, Diciembre 2018, Santiago, Chile

óptimo puede generar, una poderosa razón que motiva la idea de trabajar sobre esta temática es poder modelar el comportamiento de los usuarios y aprender de su comportamiento, idea relevante que puede ser replicable en numerosos campos.

Es por ello que el principal objetivo de este proyecto es el diseño de un sistema de recomendación que sea capaz de realizar recomendaciones personalizadas y acertadas de videojuegos disponibles en la plataforma Steam a sus diferentes usuarios. Además, de acuerdo a registros de Steam de 2014, cerca de un 37% del total de 781 millones de videojuegos presentes en su plataforma no habían sido jugados en ninguna oportunidad [9], lo cual nos invita a que nuestro sistema recomendador privilegie la novedad de manera significativa con respecto a la precisión.

Para lograr estos objetivos e implementar un sistema que sea robusto a dichos aspectos, experimentaremos con diferentes opciones que se basen en varios paradigmas de los sistemas recomendadores. Dado que el objetivo principal está dirigido a la recomendación, nuestros modelos estarán dedicados a la tarea de ranking. Las opciones estudiadas, aumentando en complejidad, fueron las siguientes:

- Sistema basado en Colaborative Filtering que utilice el algoritmo ALS para la factorización de variables latentes. Este modelo será simple al considerar un número limitado de features en la construcción de las variables latentes.
- Sistema que utilice el modelo de Factorization Machines y tome en consideración un número importante de features tanto del usuario como de los ítemes para la construcción de las variables latentes.
- Sistema basado en FM y un modelo de Deep Learning, denominado DeepFM, donde se incluya el análisis de texto de diferentes reviews y opiniones para extraer sentimiento que será usado como una dimensión más del análisis de recomendación.

Así, este trabajo estará compuesto por una sección donde se describan todos los componentes necesarios para la realización de las pruebas de recomendación, seguido esto se presentará la explicación de la metodología utilizada en el proceso de experimentación, luego se mostraran y analizarán los resultados obtenidos, y finalmente se presentarán las conclusiones extraidas del proceso experimental.

2 ESTADO DEL ARTE

Numerosos son los estudios que tocan el tema de la recomendación, no obstante, son pocos los estudios que han visto en la industria de videojuegos, las plataformas de compra de videojuegos y comunidades de esta índole un sitio interesante para probar el poder de recomendación de los algoritmos existentes. A pesar de aquello, existen varios estudios relevantes cuya aplicación podría ser útil al contexto presentado.

En Bertens et al. (2018)[2] los autores prueban el poder de dos sistemas recomendadores bajo la idea de predecir el próximo item más probable de comprar por usuario. El estudio se acota principalmente a estudiar el comportamiento de jugadores japoneses en el juego *Age Of Ishtaria* considerando su historial de progreso diario, tiempo de juego y compras. Uno de los modelos que prueban es denominado Extremely Randomized Trees (ERT), el cual es una versión aleatorizada de los árboles de decisión. Este método posee la ventaja de ser computacionalmente eficiente debido a la alta paralelización de sus componentes, además de ser óptimos a la hora de prevenir el overfitting con el coste de caer en sesgos cuando la aleatorización es elevada sobre niveles óptimos. Por otro lado, utilizaron además un algoritmo basado en Redes Neuronales profundas y dado que los datos de entrenamiento poseen la particularidad de ser secuenciales, prefirieron el uso de redes recurrentes, sin embargo, sin dar mayores detalles de esta arquitectura. Ambos modelos presentan resultados muy similares, siendo ERT quien presenta ligeras mejoras pero mostrando un mejor desempeño en cuanto a tiempos de entrenamiento y escalabilidad.

Quadrana et al. (2018) proponen un sistema recomendador basado en la atención a la sequencia de eventos como una de las mejores aproximaciones a resolver la recomendación en contextos como el propuesto. Su publicación consiste un resumen de múltiples modelos desarrollados en el área, tomando en consideración diferentes tareas y metas que se quieren lograr. Entre la discusión destacan el potencial de los modelos basados en Redes Neuronales. A pesar de que la discusión de su modelo es relevante al contexto de la industria de videojuegos, los autores destacan más su implementación a la tarea de atención en sesión. Una aplicación particular de este tipo de aplicaciones es lo que hacen los autores en Wan et al. (2018) donde estudian cadenas monótonas de comportamiento que se traducen en una secuencia de eventos que dan cuenta de la preferencia creciente que un usuario puede tener por un producto, yendo desde una caracterización implícita hacia una cada vez más explícita. La novedad presentada por el equipo es una nueva manera de enfretar el problema de recomendación, el cual prueban sobre numerosos dataset incluído uno de juegos en STEAM. Sus mejoras versus Most Popular varían entre el 1% y 28% inclusive.

Dado el contexto de redes e interacción que existe en las plataformas de videojuegos es que estudios de sistemas recomendadores basados en grafos resultan también relevantes. En esta área el estudio de Shams et al. (2016) [13] resulta uno de los más recientes. En él su método denominado GRank promete modelar correctamente las prioridades de los usuarios y discriminar de mejor manera las conexiones relevantes entre los nodos.

Dado el contexto de los estudios más recientes nuestras propuestas estarán ligadas a aprender sistemas recomendadores basados en deep learning, que puedan explotar interacciones entre usuarios e ítemes y que mezclen información tanto ítemes como usuarios, considerando información implícita como el tiempo dedicado a jugar un videojuego e información explícita como la acción de recomendar un juego o no.

3 RECOMENDADOR DE JUEGOS EN STEAM

3.1 Bases de datos

Para llevar a cabo la implementación de los sistemas a evaluar y probar su poder predictivo trabajamos sobre tres bases de datos

particulares. La primera presenta el registro de compra de videojuegos. La segunda base de datos contiene la opinión que diferentes usuarios de la plataforma Steam poseen sobre ítemes disponibles en ella. La tercera nos brinda información detallada de las características de los videojuegos.[7]

Los diferentes dataset se unen con el fin de crear tuplas de datos asociadas a cada par ítem/usuario. La tabla 1 presenta una síntesis de las diferentes features disponibles.

Feature	Tipo dato	Descripción
user_id	Str	Identificador particular de un usuario
item_id	Str	Identificador particular de un item
count	Int	Número de juegos que posee un usuario
playtime	Int	Tiempo que el ítem ha sido jugado por el usuario
RecCount	Int	Veces que un ítem ha sido recomendado
Metacritic	Int	Valoración del juego, de acuerdo a crítica especializada.
GenreIs |G| = 13	[Bool]	Verdadero cuando el item pertenece a uno de 13 géneros diferentes de juego.
Category |C| = 8	[Bool]	Verdadero cuando el item pertenece a una de 8 categorías de juego, ej: multijugador
Plataform |P| = 3	[Bool]	Verdadero si el item es soportado por uno de los 3 sistemas operativos indicados.
recommend	[Bool]	Verdadero si el usuario recomendó positivamente un item.
review	Str	Texto que contiene la opinión del usuario

Table 1: Tabla resumen de features disponibles.

Cada tupla está compuesta por 32 características particulares, en ella se encuentra información implicita y abundante como el tiempo de juego, e información más explícita, pero escaza contenida en los datos de recomendación y review. Como se observa, no existe una medida de *rating* generalizada que nos ayude a evidenciar de en qué medida un usuario gusta o prefiere de un ítem, es por ello que nuestros análisis posteriores deben basarse en medidas de implicit feedback o que combinen más de un tipo de información para realizar recomendaciones. En este sentido, especial atención pondremos al tiempo de juego como una medida relevante.

El set de datos de registros de compra contiene un total de 5,153,209 tuplas. En ella se agrupa un total de 70,912 usuarios diferentes y 10,978 ítemes entre los cuales elegir. Estos números dan cuenta de una densidad de 0.66% de registros observados.

Dada la baja densidad del dataset se preveen dificultades para el entrenamiento y correcto aprendizaje de las relaciones entre ítemes y usuarios. Es por esto que proponemos realizar cortes con el fin de obtener una densidad final representativa de al menos el 10% del set de datos sin perder el orden de magnitud del número de registros. Para llevar a cabo este objetivo es que fijamos un corte para ítemes al considerar únicamente aquellos con un mínimo de 200 compras; por su parte consideraremos usuarios que al menos hayan realizado un número de 100 compras. Algunas características resumen de la nueva base de datos se encuentran en la tabla 2.

Feature	Cantidad
Número de registros	2,149,858
Número de usuarios diferentes	8,183
Número de ítemes diferentes	2,872
Número de reviews	9,823
Densidad del dataset	9.14%
Compras promedio por usuario	262.72
Compras promedio de ítemes	748.55

Table 2: Tabla resumen de la base de datos después de los cortes.

Como se aprecia en la tabla, los objetivos perseguidos se consiguen al obtener una densidad cercana al límite deseado reduciendo el tamaño del set de datos aproximadamente a la mitad. Junto al análisis de estos números, también es importante verificar de qué manera el número de ítemes o usuarios encontrados de manera más frecuente aportan sesgo a los registros. Debido a ello es que se generan las figuras 1 y 2.

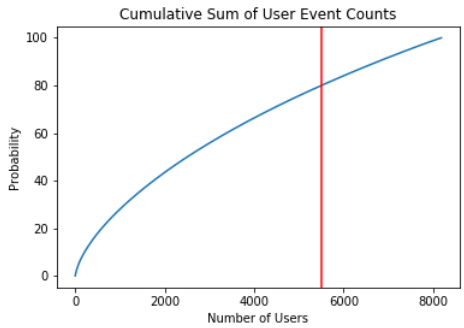

Figure 1: Suma cumulativa del aporte de diferentes usuarios al número de registros. 67% de los usuarios explican el 80% de registros.

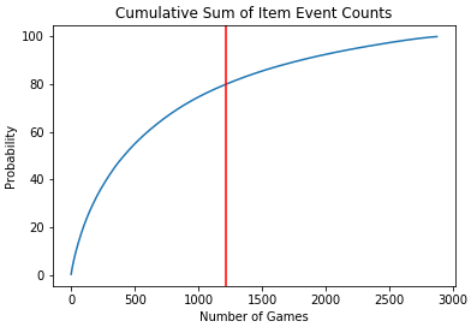

Figure 2: Suma cumulativa del aporte de diferentes ítemes al número de registros. 40% de los ítemes explican el 80 % de registros.

La figura 1 nos dice que el sesgo es bastante despreciable desde el punto de vista de usuarios, donde el 67% de ellos da cuenta del 80% del total de registros. Una situación diferente nos presenta la figura 2 en donde sí vemos que un gran número de registros (80%) se explica por una cantidad reducida (40%) de ítemes. Esta situación nos llama a estar alertas frente la capacidad de nuestros sistemas recomendadores de recomendar cosas diferentes.

Como se comentó anteriormente, la medida de implicit feedback que será utilizada para medir la preferencia que un usuario le da un ítem será el tiempo que dicho usuario dedica a jugarlo. Un límite de 5 horas se escogió como el límite para expresar dicha preferencia como positiva. Si bien valores superiores podrían haber sido introducidos para hacer aún más restrictiva y segura la medida de preferencia, se determinó que tal medida es suficiente al considerar que existen numerosos juegos cuya duración base no sobrepasa dicha cantidad de horas. La existencia de este dato podría ayudar a refinar de manera individual dicho límite. Las figuras 3 y 4 analizan cómo es la distribución de tiempo invertido de juego para los usuarios más activos y juegos más comprados respectivamente.

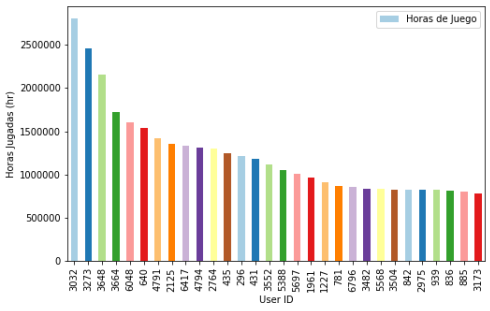

Figure 3: Distribución del tiempo de juego de los usuarios más activos.

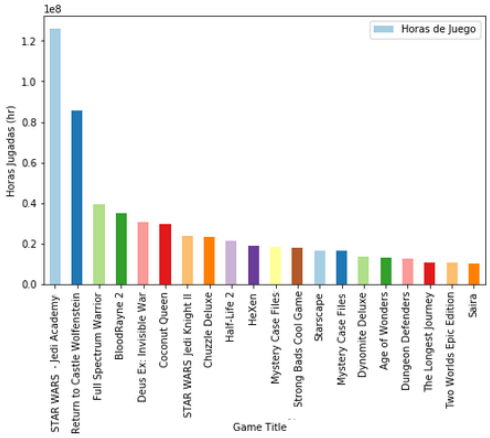

Figure 4: Distribución del tiempo de juego de los videojuegos más jugados.

Como se observa en las gráfica 3 que en general los tiempos de juego de cada usuario activo no varía significativamente en órdenes de magnitud. Por su parte la variación es más notoria para los juegos más jugados particularmente entre el juego en primer lugar respecto al resto de ellos. Cuando observamos los promedios de tiempo de juego encontramos otras características informativas de las distribuciones. Por un lado el tiempo promedio que un usuario invierte jugando es igual a 167,317.9 horas, mientras que el tiempo promedio que un ítem ha sido jugado alcanza las 476,727.85 horas. Nuevamente nos encontramos frente a una situación donde los usuarios presentan una distribución menos heterogénea y cierto número de ítemes constituyen la mayor influencia esta vez desde el punto de vista de preferencia. A pesar de estos número que pueden sugerir la baja capacidad de ofrecer listas de recomendación diversas, al transformar el tiempo de juego al indicativo de preferencia bajo el límite de 5 horas observamos que un total de 979,380 registros, que representan el 46% del total del set de datos corresponde a ítemes relevantes para la recomendación, implicando que el resto son juegos con muy bajo número de horas de juego cosa que explica la diferencia entre la gráfica 4 y el promedio de juego del total de ítemes.
Para trabajar con estos datos y generar entrenamientos significativos de los algoritmos presentados en las secciones posteriores, se usará la técnica de K-folders, con el fin de hacer particiones aleatorias que den mayor robustez al entrenamiento.

3.2 Modelos de sistemas recomendadores

3.2.1 *ALS*..

Dentro de los diversos métodos de factorización matricial, ALS destaca como un algoritmo capaz de trabajar usando feedback implícito y que es bastante fácil de implementar.

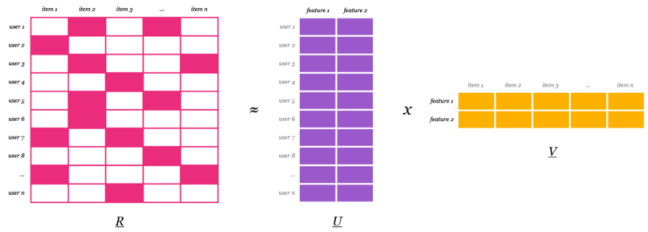

Figure 5: Representación del método de factorización matricial.

Como muestra la figura 5, la idea de usar el método de factorización es encontrar una nueva forma de factorización que represente a ítemes (y_i) y usuarios (x_u) de modo que la valoración particular que un usuario tiene por un ítem esté representada por el producto de dichos factores latentes.
Una innovación que este método introduce es analizar de manera separada la medida de implicit feedback mediante la introducción de una medida de preferencia (*preference*) que indica precisamente si el usuario ha consumido o no un determinado ítem,

$$p_{ui} = \begin{cases} 1 & \text{if} r_{ui} > 0 \\ 0 & \text{if} r_{ui} = 0 \end{cases} \quad (1)$$

La segunda medida indica la confianza (*confidence*) que se tiene en la medida que otorga preferencia.

$$c_{ui} = 1 + \alpha * r_{ui} \tag{1}$$

Donde α es un factor de escala lineal que da mayor importancia a los ítemes relevantes. El valor $\alpha = 40$ es usualmente usado, el mismo encontrado en el paper original que presenta los mejores resultados. Bajo este esquema la búsqueda de los factores latentes de usuarios x_u e ítemes y_i se encuentran bajo una redefinición de la función de pérdida usada en el entrenamiento.

$$min_{y*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(\sum_u ||x_u||^2 + \sum_i ||y_i||^2), \tag{2}$$

ecuación en donde se verifica la optimización por Mínimos cuadrados partiendo con una inicialización aleatoria de los factores. De esta manera las posteriores actualizaciones de los factores latentes estarán dadas por,

$$x_u = (V^T V + V^T(C^u - I)V + \lambda I)^{-1} V^T C^u p(u) \tag{3}$$

$$y_i = (U^T U + U^T(C^i - I)U + \lambda I)^{-1} U^T C^i p(i) \tag{4}$$

Algunos parámetros relevantes como el número de factores latentes a considerar se obtienen mediante análisis en varias iteraciones.

3.2.2 *Factorization Machines (FM)* .

Las Máquinas de Factorización o Factorizatión Machines, son un tipo de sistemas recomendadores basado en factores latentes, que puede tener âĂIJnâĂİ dimensiones (orden-n) de dichos factores []. En nuestro caso se definió n=2, debido a que ha sido ampliamente utilizado de estaforma ya que entrega buenos resultados manteniendo un tiempo de procesamiento razonable. Con esta configuración, la salida es una predicción resultante de la interacción lineal de las entradas (orden-1) más la de los factores latentes, definidos por las relaciones entre pares de entradas (orden-2), tal y como se observa en la ecuación 5.

$$y(x) := w_0 + \sum_{i=0}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < v_i, v_j > x_i x_j \tag{5}$$

Decidimos escoger este modelo por ser un tipo de sistema que puede escalar facilmente, ha demostrado un gran desempeño al trabajar con bases de datos dispersas, y que además soporta datos de entrada numéricos de cualquier tipo. Todas estás son características que contribuyen en el proceso de recomendación en una empresa como la de los videojuegos, donde la variedad de productosy el crecimiento son factores críticos del proceso. En la figura 6 se observa un ejemplo de arquitectura de un modelo de máquina de factorización y en la 7 un ejemplo de entradas y salidas que puede recibir.

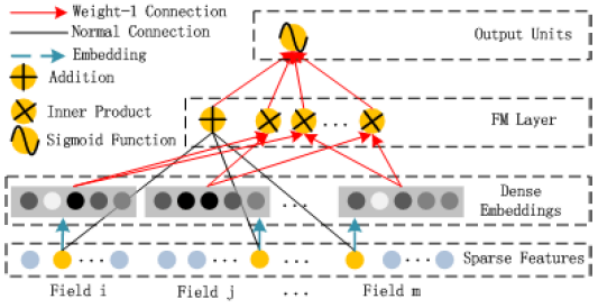

Figure 6: Ejemplo de arquitectura de una Máquina de Factorización.

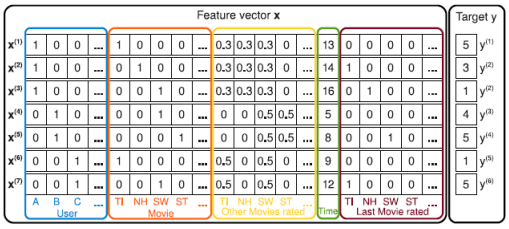

Figure 7: Ejemplo de características de entradas que pueude recibir una Máquina de factorización. Se observa que puede trabajar con variables de tipo buleanas, enteras y flotantes en paralelo.

3.2.3 *DeepFM*.

DeepFM es un modelo más nuevo que busca aprovechar aprovechar la versatilidad de los modelos de Factorization Machines para modelar interacciones de bajo orden entre las variables de entrada, con la capacidad modelar interacciones más profundas que tienen las Deep Neural Networks (DNN) [6]. Para ello se implementa una DNN en paraelo a la FM y el resultado de ambas se une en un último nodo utilizando una Sigmoide, haciendo que la salida sea una predicción resultante de la interacción de varios ordenes de las entradas que representa la probabilidad de si un usuario pertenece o no a una cierta clase definida con las etiquetas de entrenamiento. En nuetro caso, dicha clase se refería a si el usuario le gustaba el juego o no.

Se decidió clasificar las entradas según su tipo (usuario, item, categoría, plataforma, etc.) y utilizar un emmbeding de las entradas que permitiera darle el mismo peso a cada grupo, para que el efecto de las variables "sparse" no interfiriera de forma significativa contra el efecto de las otras variables. Dicho embeding consistía en generar vavrias matrices que permitieran transformar las N_i variables de cada clase de entradas en un número K fijo definido como parámetro del modelo. Esta idea fue tomada del mismo paper [6] de donde se obtuvo la información del modelo, que además nos aportó gran parte de su implementación, la cual, en su mayoría, se encuentra hecha utilizando la librería de tensorflow.

La figura 8 muestra una representación de todos los componentes del modelos y cómo se encuentran conectados.

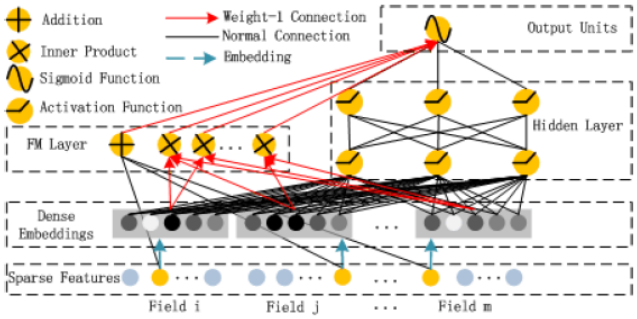

Figure 8: Distribución del tiempo de juego de los videojuegos más jugados.

3.3 Análisis de sentimiento

Uno de los datos que nos parece más interesante dentro del set de datos es aquel correspondiente a las reviews. Una review es un trozo de texto que guarda la opinión que un usuario posee acerca del item particular. No obstante, un trozo de texto como tal no resulta conveniente para ser usado en los modelos con los que trabajaremos, es por ello que proponemos el análisis de sentimiento como una técnica relevante para traducir dicho texto a una medida numérica representante de qué tan negativa o positiva es la idea representada. Su introducción podría aportar un grado más de confianza a la valoración expícita que representa la recomendación, pudiendo ser capaces de imputar cuando sean medidas inconsistentes. Para lograr este objetivo usamos tres herramientas diferentes.

En primer lugar usamos el algoritmo de libre uso Tweetment. Este método se encuentra como una librería de python capaz de hacer clasificación de sentimiento en etiquetas positivo o negativo, representados por 1 o 0, respectivamente. Este algoritmo está entrenado sobre tweets logrando de acuerdo a su publicación original un F1-score de 69.02% en clasificación binaria de mensajes.[11]

Otro algoritmo probado corresponde a SentiWordNet. Este fue desarrollado como una aplicación de Opinion Mining capaz de clasificar frases y texto en un rango continuo entre [-1, 1] indicativo de qué tan negativo o positivo es la idea expresada en él. SentiWordNet basa su análisis en la utilización de un diccionario pre-entrenado que sintetiza relaciones de n-gramas otorgándoles puntaje de su positividad, negatividad u objetividad, sin ser excluyentes.[5]

Finalmente se probó un modelo basado en Redes Neuronales y un embedding usando Word2Vec, dicho algoritmo intentaba usar atención sobre el texto, en especial sobre el ítem en cuestión. No obstante, en el desarrollo de este algoritmo nos enfrentamos a dos dificultades: La clasificación nos permite entregar una etiqueta simplemente positiva o negativa y no continua; a su vez, la implementación de la arquitectura logró un F1-score muy inferior al algoritmo similar Tweetment.[10]

De los algoritmos descritos se decidió trabajar finalmente con SentiWordNet por la ventaja de otorgar una etiqueta con valor continuo que resulta ser más flexible en el contexto de comentarios.

4 METODOLOGÍA DE EXPERIMENTACIÓN Y EVALUACIÓN

Los sucesivos experimentos realizados consistieron en el entrenamiento y evaluación de los diferentes modelos presentados. Para realizar estas evaluaciones y evitar medir de manera sesgada el desempeño de estos modelos se procedió a realizar cortes aleatorios en diversos K-folders y luego aplicando la técnica de validación cruzada al promediar dichos scores. Para entrenar ALS se usaron 5 folders mientras que en el caso de FM y DeepFM usamos 3. El set de entrenamiento también se obtuvo de manera aleatoria, no obstante, dado que realizamos cortes en que un usuario tiene al menos 100 ítemes comprados verificamos que al menos 10 ítemes por cada usuario quedara dentro del set de entrenamiento. De esta manera intentamos predecir la preferencia que se tuvo sobre ellos y verificar qué recomendaciones realizan nuestros modelos.

Para medir el nivel de funcionalidad de los algoritmos introducimos diversas métricas para evaluar su desempeño. En primer lugar tomamos como una medida relevante el tiempo de entrenamiento de los algoritmos, parámetro importante en darnos una idea de la escabilidad de los modelos. Además de ello medimos el valor de MAP a la décima recomendacion. Esta métrica es la abreviación de Mean Average Precision, una medida que promedia sobre todo los usuarios la precisión media de una lista de recomendación cada vez que encontramos un elemento relevante, en este caso compuesta de 10 elementos. O sea, si consideramos el Average Precision como,

$$AP = \frac{\sum_k P@k * rel(k)}{\#elementos\_relevantes} \tag{6}$$

el mean average precision queda dado por,

$$MAP = \frac{\sum_{u=1}^{n} AP(u)}{\#usuarios} \tag{7}$$

De igual manera medimos NDCG a la décima recomendación. Esta medida trata de estandarizar la ganancia y utilidad de una lista de recomendación a través de la consideración de la relevancia de cada elemento e introduciendo una medida de descuento logarítmica por su posición en la lista de recomendación. La métrica se calcula como una división entre el Discounted Cumulative Gain a la posición k:

$$DCG_k = \sum_{i=1}^{k} \frac{rel(i)}{log_2(1+i)} \tag{8}$$

y su valor ideal a la posición k, también llamado Ideal DCG, de esta manera la métrica se define como,

$$NDCG_k = \frac{DCG_k}{IDCG_k} \tag{9}$$

Finalmente nos interesa medir métricas de cuán diversas y nuevas son las recomendaciones realizadas por nuestros sistemas. Dado el contexto en que muchísimos videojuegos nuevos aparecen año

a año en escena estas métricas resultan ser muy relevantes. Para medir dichas cantidades usaremos relaciones basadas en la similaridad entre ítemes en una lista de recomendación. Estas métricas se definen a continuación, [3]

$$Novelty(R|u) = \sum_{n,j \in u} disc(n)p(rel|i_n, u)p(rel|j_n, u)d(i, j) \quad (10)$$

$$Diversity(R|u) = 2 \sum_{k<n} disc(n)disc(k)p(rel|i_n, u)d(i, k) \quad (11)$$

Las definiciones introducen un factor de discount, el cual particularmente se eligió uno logarítmico, igual al usado en NDCG. El término d(i, k) hace referencia a la distancia entre los pares de ítemes, medido como una distancia coseno en este trabajo. Finalmente los términos $p(rel|i_n, u)$ hablan de la probabilidad que un usuario u prefiera el ítem i_n entendido a su vez como una relevancia; esta medida se obtiene directamente como el output calculado por los sistemas de recomendación.

5 ANÁLISIS DE PARÁMETROS

Con el fin de obtener las mejores configuraciones bajo las cuales los modelos funcionan de la mejor manera, se procedió a hacer un análisis de los diferentes parámetros afinables en nuestros modelos.

En el caso de ALS lo parámetros que se pueden variar tienen que ver con el número de factores latentes usados para la factorización matricial, el valor del factor λ correspondiente al parámetro de regularización en las ecuaciones 3 y 4. Finalmente el número de épocas de entrenamiento o iteraciones. El análisis de sensibilidad se presenta en las siguientes figuras.

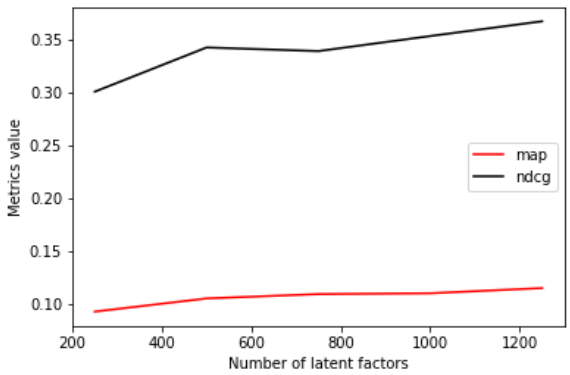

Figure 9: Entrenamiento sobre distinto número de factores latentes.

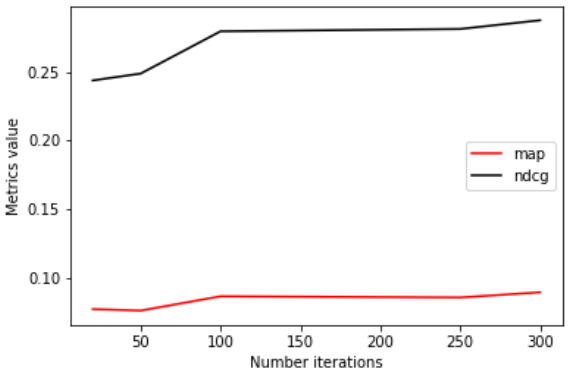

Figure 10: Número de iteraciones del algoritmo

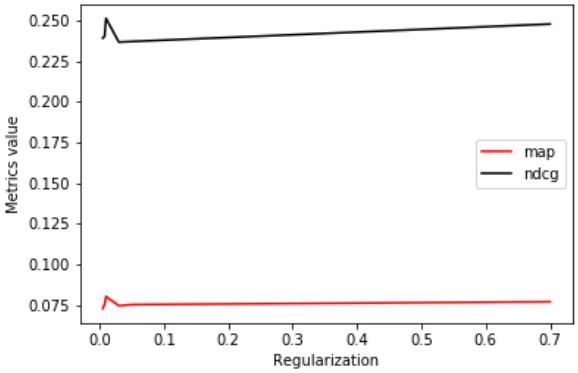

Figure 11: Factor de Regularización usado.

Probamos combinaciones de los mejores parámetros encontrados, usando un parámetro de regularización de 0.01, un número de factores latentes igual a 500 en favor del tiempo de entrenamiento, aunque se observa que un número mayor a 800 vuelve a ser favorable; finalmente un máximo de iteraciones de 300.

Por su parte, los modelos de Deep Learning como DeepFM y DNN fueron entrenados cambiando dos parámetros relevantes. El primero influye directamente en la arquitectura de la red y corresponde al número de neuronas de capa. El segundo parámetro corresponde al tamaño del batch, o sea, al número de mediciones o datos que ingresan a la vez a la red. Este parámetro puede ser muy relevante, pudiéndo influir en el overfitting o la velocidad de aprendizaje del modelo. El entrenamiento produce gráficas cuyo score es medido en términos de la norma Gini, que representa una buena métrica para clasificación binaria y entrega un mayor espectro de resolución que ROC-AUC. Un ejemplo de dicho entrenamiento se observa a continuación:

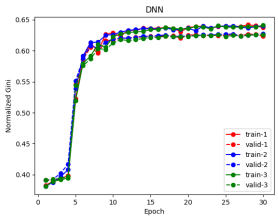

Figure 12: Curva de entrenamiento DNN.

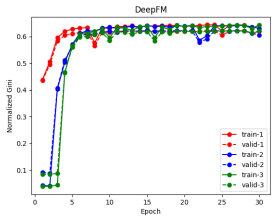

Figure 13: Curva de entrenamiento DeepFM.

Mediante el análisis de estas curvas y las métricas definidas se realizó el respectivo análisis de sensibilidad resumido en la tabla 3.

Config	Tiempo (seg)	MAP @10	NDCG @10	Novelty	Diversity
B_{1024} HL_{32x32}	2027,52	0,8911	0,9437	0,1667	0,4245
B_{512}	2186,77	0,8940	0,9456	0,1845	0,4967
B_{256}	2692,96	0,8919	0,9440	0,1847	0,4996
HL_{16x16}	1815,70	0,8939	0,9454	0,1815	0,4775
HL_{8x8}	**1708,57**	0,8941	0,9455	0,1837	0,4869
B_{512} HL_{8x8}	2180,6	**0,8945**	**0,9458**	**0,1857**	**0,5149**

Table 3: Tabla de análisis de sensibilidad de parámetros sobre el modelo DeepFM, variando la cantidad de neuronas en cada capa ($H_{i,j}$) y el tamanño del batch (B_n), ambas del componente DNN del modelo. Se tomo como base un modelo con un tamaño de batch de 1024 y con 32 neuronas por capa

Lo primero que se observa es que al disminuir el tamaño del batch de 1024 a 512 se obtiene una mejora sobre todas las métricas. Sin embargo, al disminuir un poco más este parámetro a 256, los resultados disminuyen un poco, a excepción de la métrica de Novelty que aumenta un porcentaje pequeño, pero que no es significativo con respecto las pérdidas. Esto se puede deber a que el reducir el tamaño del batch puede estar generando un sobreentrenamiento con el set de training que afecta los resultados de la etapade prueba.

También se puede ver que al disminuir la cantidad de neuronas por capa de 32 a 8, se obtienen mejoras resultados sobre todas las métricas. Esto puede deberse a que para la cantidad de *Features* y registros trabajados, con menos neuronas se realiza un mejor embeding de la información, y por lo tanto se obtienen interacciones más ricas que con un número mayor de neuronas.

Finalmente, se realizó la prueba con un modelo que tuviera las dos configuraciones de parámetros que arrojaron los mejores resultados, y se obtuvo sistema que superó a los anteriores en todas las métricas estudiadas, tal y como se esperaba.

6 RESULTADOS

En la tabla 4 se muestran los resultados obtenidos al evaluar todos los modelos de sistemas recomendadores estudiados en este trabajo de investigación:

Config	Tiempo (seg)	MAP @10	NDCG @10	Novelty	Diversity
ALS	1421,8	0,107	0,332	-	-
FM	1450,86	0,8929	0,9448	0,1758	0,4500
DNN	1889,58	0,8968	0,9473	0,1860	0,4845
DeepFM	2027,52	0,8911	0,9437	0,1667	0,4245
FM (SS)	**1416,04**	0,8936	0,9452	0,1798	0,4597
DNN (SS)	1866,35	**0,8972**	**0,9474**	**0,1970**	**0,5335**
DeepFM (SS)	1984,17	0,8916	0,9439	0,1906	0,5333

Table 4: Tabla de resultados de pruebas de todos los modelos estudiados, incluyendo el análisis de los resultados con y sin (SS) la variable de sentimiento.

Lo primero que se observa es que, a pesar de que el modelo base ALS es el más rápido de entrenar, todos los modelos estudiados lo superan en el resto de las métricas, como era de esperarse.

Como un segundo punto importante se tiene que, al contrario de lo propuesto, el modelo de DNN supera en todos los aspectos tanto a FM como al modelo DeepFM, que buscaba aprovechar las ventajas de los modelos anteriores. Esto lo que nos indica es que las interacciones de mayor orden aportadas por el modelo DeepNN brindan mayor información que las de bajo orden aportadas por el modelo FM, por lo cual la salida de este último pueden estar generando ruido al resultado final.

Finalmente, se observa que, contrario a lo que esperabamos, al quitar la variable generada por el proceso de análisis de sentimiento sobre los reviews, los resultados mejoran para todas los modelos y métricas estudiadasa. Sin embargo, el hecho de que la relación de registros con reviews sea tan pequeã en con paración a los ergistros sin este dato, nos hace pensar que los resultados no reflejan el verdadero efecto que puede tener este análisis sobre bases de datos que estén más balanceadas. La perdida de rendimiento puede atribuirse a la baja densidad de reviews, que hace que la variable de sentimiento, más que agregar información relevante al entrenamiento, se traduzca en un efecto de ruido que genera los resultados observados.

7 CONCLUSIONES

En este trabajo se experimentaron con diferentes opciones de sistemas recomendadores dentro del contexto de recomendación de videojuegos, utilizando a la empresa STEAM como plataforma de prueba. Se trabajó con tres bases de datos que contenían información de las compras de los usuarios, de las horas dedicadas a cada item, de su interacción social (críticas) y de las características propias de los videojuegos. Encontrando de este modo distintos campos en que se reúnen. Se escogió el modeo ALS como modelo de base para la comparación y se realizaron pruebas con los modelos de Máquinas de Factorización (FM), DeepNN y de DeepFM, siendo este último, un modelo que utiliza una red neuronal profunda (DeepNN)

que funciona en paralelo a una capa compuesta de una FM, para introducir interacciones de mayor orden entre las entradas, con el objetivo de que esto pudiese mejorar el rendimiento de nuestra predicción en cuanto a los factores de novedad, diversidad y presición.

Todos los modelos estudiados lograron superar al modelo base ALS. Por su parte, DeepNN destacó sobre el resto. A pesar de ser un modelo más simple que DeepFM, logró explotar de mejor manera las relaciones item-usuario, y a pesar de que demora más que FM y DeepFM en alcanzar resultados competitivos, logra resultados muy parejos sobre diferentes dataset. Además, obtiene mejores resultados con las métricas evaluadas, lo cual implica que las interacciones de mayor orden aportadas por el modelo DeepNN brindan mayor información que las de bajo orden aportadas por el modelo FM.
Se propone que el modelo DeepNN se beneficia del embedding de datos, cosa que le permite explorar diferentes conjuntos de ellos, otorgándoles igual importancia.
Para obtener la información más explícita se trabajó sobre las reviews usando distintos métodos de análisis de sentimiento, encontrando una medida continua que pudiera abtraer aún más la opinión del usuario por el videojuego, cosa que pudiera mejorar los resultados obtenidos. Concluímos que la baja densidad de las reviews con respecto al total de registros no aporta a una mejora de los resultados, traduciéndose más bien en un efecto de ruido, al hacer que las métricas reduzcan su valor tanto en MAP@10 como en NDCG@10 e inclusive en las medidas de Novelty y Diversity.

Como trabajos futuros se piensa realizar pruebas de análisis de parámetros sobre el modelo DNN que arrojó los mejores resultados para ver si se pueden superar. También se tiene planeado utilizar otro set de datos donde la relación entre los registros con y sin reviews sea más cercana para poder probar si verdaderamente el análisis de sentimiento puede ser una herramienta útil para la recomendación de videojuegos. Finalmente se podría verificar al cambiar el número de campos de datos con que se alimentan las redes DNN, si efectivamente la red se beneficia del embedding y de otorgarle la misma importancia a ciertos conjuntos de datos en especial.

REFERENCES

[1] [n. d.]. Global Games Market Revenues 2018 | Per Region & Segment. https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/. Accessed: 2018-11-20.

[2] Paul Bertens, Anna Guitart, Pei Pei Chen, and África Periáñez. 2018. A Machine-Learning Item Recommendation System for Video Games. *ArXiv e-prints*, Article arXiv:1806.04900 (June 2018), arXiv:1806.04900 pages. arXiv:stat.ML/1806.04900

[3] Pablo Castells, SaÃžl Vargas, and Jun Wang. 2011. Novelty and Diversity Metrics for Recommender Systems: Choice, Discovery and Relevance. *Proceedings of International Workshop on Diversity in Document Retrieval (DDR)* (01 2011).

[4] Takahashi D. [n. d.]. Newzoo: Game industry growing faster than expected, up 10.7% to $116 billion 2017. https://venturebeat.com/2017/11/28/newzoo-game-industry-growing-faster-than-expected-up-10-7-to-116-billion-2017/. Accessed: 2018-11-20.

[5] Andrea Esuli and Fabrizio Sebastiani. 2006. SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LRECâĂŹ06*. 417–422.

[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. *CoRR* abs/1703.04247 (2017). arXiv:1703.04247 http://arxiv.org/abs/1703.04247

[7] Kelly C. Hicks E. [n. d.]. COMP 7150 | Data Science Project Report. Accessed: 2018-11-20.

[8] Ell K. [n. d.]. Video game industry is booming with continued revenue. https://www.cnbc.com/2018/07/18/video-game-industry-is-booming-with-continued-revenue.html. Accessed: 2018-11-20.

[9] Orland K. [n. d.]. Introducing Steam Gauge: Ars reveals SteamâĂŹs most popular games. https://arstechnica.com/gaming/2014/04/introducing-steam-gauge-ars-reveals-steams-most-popular-games/. Accessed: 2018-11-20.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *CoRR* abs/1310.4546 (2013). arXiv:1310.4546 http://arxiv.org/abs/1310.4546

[11] Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. *ArXiv e-prints*, Article arXiv:1308.6242 (Aug. 2013), arXiv:1308.6242 pages. arXiv:cs.CL/1308.6242

[12] Randy N. [n. d.]. Randy Nelson. https://sensortower.com/blog/app-revenue-and-downloads-1h-2018. Accessed: 2018-11-20.

[13] Bita Shams and Saman Haratizadeh. 2016. Graph-based Collaborative Ranking. *CoRR* abs/1604.03147 (2016). arXiv:1604.03147 http://arxiv.org/abs/1604.03147

[14] Rafet Sifa, Christian Bauckhage, and Anders Drachen. 2014. Archetypal game recommender systems. *CEUR Workshop Proceedings* 1226 (01 2014), 45–56.