

JUEGOTEA

PROYECTO FINAL
GRUPO 1

Javier Sesé García
Leila Ruiz Casanova
Víctor del Pino Castilla

Índice

Índice.....	2
Introducción.....	3
Obtención de datos.....	4
Herramientas de extracción.....	5
Obtención de usuarios	5
Obtención de Juegos y Notas.....	6
Refinado de los juegos	7
Refinado de las aristas	8
Red generada	9
Algoritmo de recomendación	10
Ejecución del sistema de recomendación.....	11
Observaciones finales y conclusiones.....	13
Bibliografía	14

Introducción

En el presente documento se describen las acciones realizadas para la realización de un sistema de recomendación de juegos de mesa.

Los sistemas de recomendación Online permiten que los usuarios obtengan propuestas de productos en base a sus gustos y acciones pasadas, los mejores ejemplos los encontramos en las plataformas de comercio electrónico (Amazon y eBay) y más concretamente en el ámbito del ocio los encontramos que nos recomiendan música (Spotify), series y películas (Netflix, Amazon Video).

Para el nicho de los juegos de mesa actualmente no hay un sistema de recomendación especializado, consultando páginas web genéricas de comercio electrónico y especializadas encontramos diferentes grados de recomendaciones de productos, pero no un sistema concreto de recomendación de juegos. En páginas como [Zacatrus](#) y [Fnac](#), encontramos recomendaciones genéricas de juegos que incluyen otros productos como fundas y las recomendaciones parecen generadas por las características y mecanismos sados en cada juego, en [Juegos de la mesa redonda](#) encontramos juegos relacionados pero visitando la plataforma hemos encontrado páginas en las que no se referenciaba ningún otro producto lo que nos lleva a pensar que son referencias creadas a mano, de calidad pero insuficientes y en páginas como [GoblinTrader](#) y [El Corte Ingles](#) no se ofrecen recomendaciones

Nuestro objetivo principal es desarrollar un sistema que dado uno o varios juegos como entrada ofrezca una salida con diversos juegos relacionados. Para establecer esta relación entre los juegos y ofrecer las mejores recomendaciones disponemos de la información en la plataforma BGG [[Board Game Geek](#)]¹, esta es una red social centrada en los juegos de mesa que entre otra funcionalidad dispone de un sistema de calificación que usaremos como base para establecer las relaciones entre los juegos.

En las siguientes páginas se desarrolla la creación del sistema de recomendación.
Todo el código se puede encontrar en el repositorio <https://github.com/victordpc/SOC>.

Bienvenidos al juego desconectado.

1 <https://www.boardgamegeek.com/>

Obtención de datos

Para obtener los datos hemos utilizado la red social BGG en la que los usuarios tienen el interés común sobre los juegos de mesa, dispone de una ficha sobre la inmensa mayoría de los juegos de mesa que existen en la que se pueden encontrar además de las características de cada uno, imágenes, videos de partidas, estadísticas, expansiones, un foro para cada uno de los juegos, expansiones, se pueden subir ficheros con modificaciones, reglas propias. Como secciones importantes además dispone de un foro, una sección de noticias, un apartado de compra/venta para juegos, blogs de los usuarios, capacidades para que los usuarios generen sus colecciones de juegos y la parte más interesante para este estudio permite que los usuarios valoren cada uno de los elementos disponibles.

Con toda esta información disponible hay que seleccionar los datos relevantes que nos permitan efectuar recomendaciones de calidad.

Para ello en una primera aproximación planteamos generar un grafo bipartito en el que existan nodos representando a los usuarios y a los juegos en el que las aristas estén restringidas a unir un nodo usuario a un nodo juego siempre que ese usuario haya valorado a ese juego con una nota superior a un umbral, de forma que el grafo abarque la totalidad de los juegos y los usuarios disponibles, y se obtenga la recomendación para un juego, en base a los juegos relacionados con los usuarios que están relacionados con el juego dado como entrada.

Debido a la dificultad para obtener la totalidad de los datos y tratarlos por recomendaciones de los profesores se abandona esta idea y se plantea la posibilidad de generar dos grafos no bipartitos uno de juegos y otro de usuarios.

Debido a que nuestro objetivo es generar recomendaciones para juegos de mesa nos centramos en el grafo de juegos abandonando el grafo de usuarios.

Para obtener los datos que necesitamos para este análisis disponemos de dos API V1² y V2³ [*Application Program Interface*] publicadas por la propia BGG y la información que podemos obtener a realizar operaciones usando la técnica Web Scraping⁴.

La estrategia que planteamos para obtener los datos consiste en, obtener los usuarios que han valorado un juego conocido, de esta forma podemos confiar que un gran abanico de tipos de jugadores, y obtener todas las valoraciones realizadas por estos usuarios y en base a estos datos generar la información de los nodos y aristas.

Para obtener los datos hemos utilizado el juego Azul⁵ y establecido la nota umbral para relacionar los juegos en 9

² https://boardgamegeek.com/wiki/page/BGG_XML_API

³ https://boardgamegeek.com/wiki/page/BGG_XML_API2

⁴ https://es.wikipedia.org/wiki/Web_scraping

⁵ <https://boardgamegeek.com/boardgame/230802/azul/ratings?rated=1>

Herramientas de extracción

Se han generado una serie de scripts en lenguaje Python en su versión 3.6⁶ con ayuda de la librería Request⁷, que nos permiten obtener la información necesitada, están pensados para ser ejecutados de forma secuencial obteniendo la información y refinándola según se realizan los siguientes pasos:



1. Obtenemos los nombres de todos los usuarios que han valorado un juego.
2. Obtenemos todos los juegos valorados por cada uno de los usuarios y las notas dadas a cada juego.
3. Limpiamos la información de cada juego obtenida en el paso 2 y generamos una lista de los juegos únicos que tenemos disponibles.
4. Limpiamos la información de cada relación obtenida en el paso 2 y generamos una lista con las aristas que relacionan los juegos y sus pesos asociados

Obtención de usuarios

En un primer intento probamos a realizar un scraping de la web para obtener los datos, pero comprobamos que debido a la forma de construir la web y las respuestas que nos devolvían las peticiones se incrementaba demasiado el nivel de dificultad, por lo que abandonamos esta línea de trabajo para obtener los datos mediante la API.

Para obtener los usuarios de los que obtendremos los datos hemos generado el script [01_usuarios.py](#)⁸ para esto obtenemos de la API V2 mediante peticiones web con la librería Request los usuarios que han valorado el juego Azul, esta API nos devuelve un XML⁹ con la información del juego y lotes con los usuarios y la nota que estos le han asignado de 100 en 100 por lo que tenemos que recorrer todas las páginas para obtener estos usuarios que guardamos en un [fichero](#)¹⁰ para procesarlo después

Para ejecutarlo con la instrucción: `python3 01_usuarios.py`

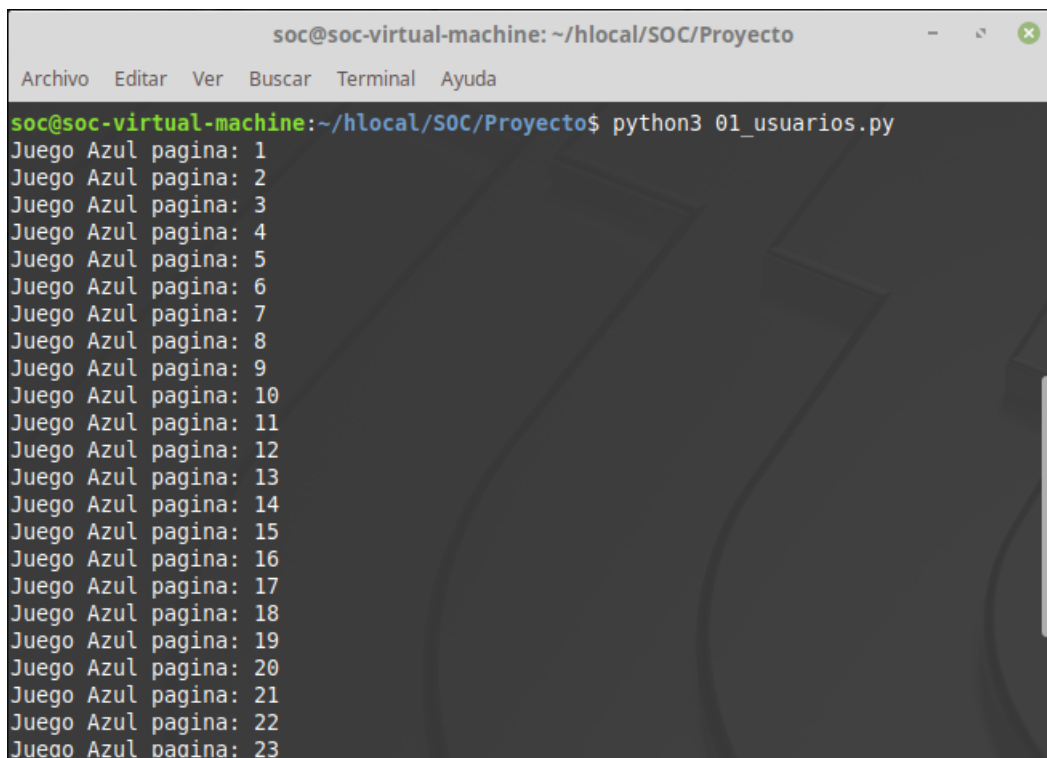
⁶ <https://www.python.org/downloads/release/python-360/>

⁷ <http://docs.python-requests.org/en/master/>

⁸ https://github.com/victordpc/SOC/blob/master/Proyecto/01_webCrawler.py

⁹ <https://api.geekdo.com/xmlapi2/thing?id=230802&comments=1&page=1&pagesize=100>

¹⁰ <https://github.com/victordpc/SOC/blob/master/Proyecto/Files/usuarios.csv>



```
soc@soc-virtual-machine: ~/hlocal/SOC/Proyecto
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
soc@soc-virtual-machine:~/hlocal/SOC/Proyecto$ python3 01_usuarios.py
Juego Azul pagina: 1
Juego Azul pagina: 2
Juego Azul pagina: 3
Juego Azul pagina: 4
Juego Azul pagina: 5
Juego Azul pagina: 6
Juego Azul pagina: 7
Juego Azul pagina: 8
Juego Azul pagina: 9
Juego Azul pagina: 10
Juego Azul pagina: 11
Juego Azul pagina: 12
Juego Azul pagina: 13
Juego Azul pagina: 14
Juego Azul pagina: 15
Juego Azul pagina: 16
Juego Azul pagina: 17
Juego Azul pagina: 18
Juego Azul pagina: 19
Juego Azul pagina: 20
Juego Azul pagina: 21
Juego Azul pagina: 22
Juego Azul pagina: 23
```

Obtención de Juegos y Notas

Este paso se asocia con el script [02_datosJuegos.py](#)¹¹, en el que recorremos todos los usuarios recogidos en el fichero generado en el paso 1, para cada uno de estos datos realizamos una petición con la librería Request, a la API V1 de la que obtendremos una respuesta con el código 202, que nos indica que el servidor ha aceptado la petición y está generando los datos, cada segundo se vuelve a realizar la petición, para obtener los datos generados en el servidor en formato XML¹², recorremos los resultados para obtener los juegos que se han valorado y obtener la nota que se ha establecido.

Según recorremos los resultados, vamos almacenando en un fichero todos los juegos a los que se hace referencia con sus datos asociados, y almacenamos en una lista los juegos valorados por cada uno de los usuarios, que guardamos en un [fichero](#)¹³. Una vez hemos recogido todos los datos del usuario generamos la relación de todos los pares de juegos que le gustan y si superan la nota umbral se almacenan en un [fichero](#)¹⁴.

Para ejecutarlo con la instrucción: `python3 02_datosJuegos.py`

¹¹ https://github.com/victordpc/SOC/blob/master/Proyecto/02_datosJuegos.py

¹² <https://www.boardgamegeek.com/xmlapi/collection/victordpc?rated=1>

¹³ <https://github.com/victordpc/SOC/blob/master/Proyecto/Files/NodosJuegos.csv>

¹⁴ <https://github.com/victordpc/SOC/blob/master/Proyecto/Files/AristasJuegos.csv>

```
soc@soc-virtual-machine: ~/hlocal/SOC/Proyecto
Archivo Editar Ver Buscar Terminal Ayuda
soc@soc-virtual-machine:~/hlocal/SOC/Proyecto$ python3 02_datosJuegos.py
1/2486 -mIDE-
2/2486 1000rpm
3/2486 216stitches
4/2486 2d20
5/2486 2dTones
6/2486 9snick4
7/2486 aaj94
8/2486 Aazmp
9/2486 Aberdeen1977
10/2486 abernath
11/2486 absenthro
12/2486 Abso
13/2486 Abstract Paul
14/2486 abweins
15/2486 AC0990
16/2486 AcemanBR
17/2486 acharland
18/2486 Achire
19/2486 Adamadamadam
20/2486 AdamCarr
21/2486 adamisclassy
22/2486 adamredwoods
23/2486 adamscott
```

Refinado de los juegos

En este paso recogido en el script [03_constructorNodos.py](#)¹⁵, recorre el fichero de nodos generado en el paso 2 y elimina los duplicados y genera un [fichero](#)¹⁶ preparado para importar en Gephi¹⁷ con todos los juegos disponibles

Para ejecutarlo con la instrucción: `python3 03_constructorNodos.py`

```
soc@soc-virtual-machine: ~/hlocal/SOC/Proyecto
Archivo Editar Ver Buscar Terminal Ayuda
soc@soc-virtual-machine:~/hlocal/SOC/Proyecto$ python3 03_constructorNodos.py
Fin
soc@soc-virtual-machine:~/hlocal/SOC/Proyecto$
```

¹⁵ https://github.com/victordpc/SOC/blob/master/Proyecto/03_constructorNodos.py

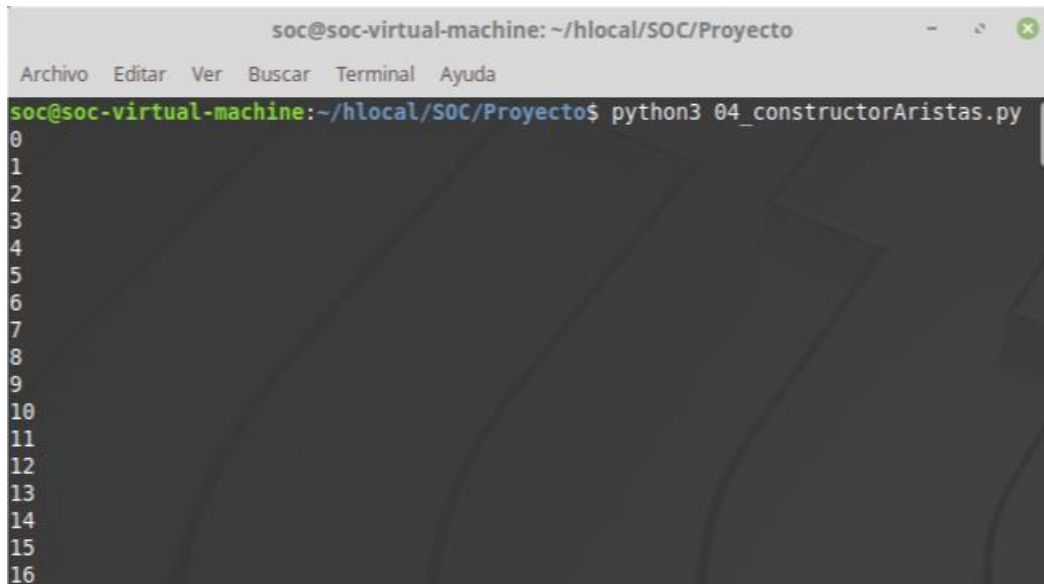
¹⁶ <https://github.com/victordpc/SOC/blob/master/Proyecto/Files/NodosGrafo.csv>

¹⁷ <https://gephi.org/>

Refinado de las aristas

En este paso el script [04_constructorAristas.py](https://github.com/victordpc/SOC/blob/master/Proyecto/04_constructorAristas.py)¹⁸, lee todos los pares de juegos generados en el paso 2 y calcula el peso que tiene la arista en función de las repeticiones que se encuentren de ese par de juegos y genera un [fichero](https://github.com/victordpc/SOC/blob/master/Proyecto/Files/AristasGrafo.csv)¹⁹ preparado para importar en Gephi con todas las aristas para relacionar los juegos disponibles incluyendo el peso para cada una de ellas.

Para ejecutarlo con la instrucción: `python3 04_constructorAristas.py`



```
soc@soc-virtual-machine: ~/hlocal/SOC/Proyecto
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
soc@soc-virtual-machine:~/hlocal/SOC/Proyecto$ python3 04_constructorAristas.py
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

¹⁸ https://github.com/victordpc/SOC/blob/master/Proyecto/04_constructorAristas.py

¹⁹ <https://github.com/victordpc/SOC/blob/master/Proyecto/Files/AristasGrafo.csv>

Red generada

Para generar la red disponemos de 37.275 nodos con 1.425.054 aristas, estas aristas relacionan 8.414 juegos que son los valorados por encima de la nota umbral, los nodos no conectados significan que al menos con la muestra de usuarios estos juegos no han sido bien valorados.

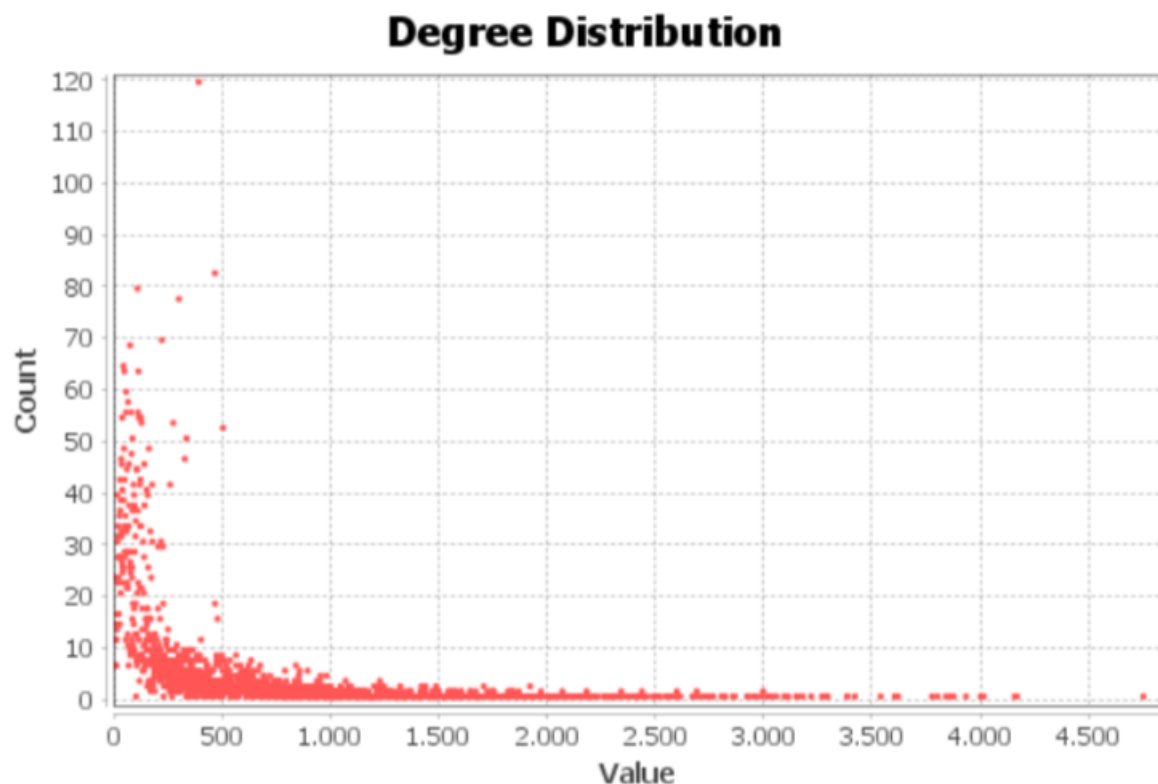
Dado que para hacer las recomendaciones necesitamos que existan conexiones entre los juegos, tenemos que descartar los nodos que no tienen conexiones, por lo que no tendremos en cuenta los 28.861 nodos desconectados para esta red.

En este grafo los nodos representan Juegos de mesa y las aristas representan que un par de juegos ha sido valorado por el mismo usuario con una nota superior al umbral establecido y representando en el peso de arista el número de veces que esto ocurre.

Si analizamos la red podemos observar que existen 3 componentes conexas, un diámetro de 4, con una distancia media de 2,04, una densidad de 0,04, bastante grande, un coeficiente de clustering de 0,787 y el grado medio de la red que se sitúa en un poco más de 338 con un grado máximo de 4839 y visualizando la distribución de grados podemos comprobar la existencia de Hubs.

Results:

Average Degree: 338,734



Algoritmo de recomendación

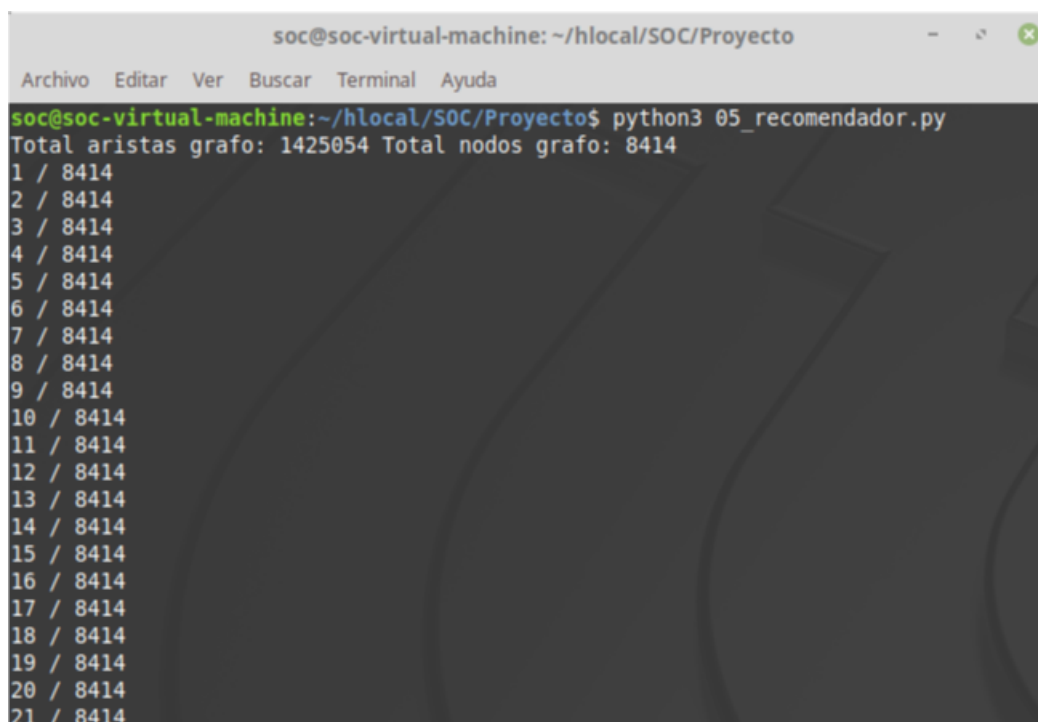
Para poder ofrecer recomendaciones tenemos que establecer como de similares son dos nodos, por lo que tenemos que usar una métrica adecuada que nos proporcione los mejores resultados. Como lo que mas nos interesa es la relevancia de las recomendaciones que ofrecemos usaremos el algoritmo que mejor puntuación ha obtenido en la métrica de Precisión (CARO MARTÍNEZ, 2017, pág. 82) para un grafo análogo, esta es la métrica EW [Edge Weight], con esta métrica determinamos la similitud entre dos nodos en base al peso de la arista que los une.

$$EW(x; y) = A_{xy}$$

A_{xy} = peso del enlace que une los nodos x e y

Este algoritmo se implementa en el script [05_recomendador.py](#)²⁰, en este script generamos un grafo, usando la librería NetworkX²¹, mediante el fichero de aristas²² generado en el paso 3 de la extracción de datos podemos obtener toda la información, ya que no necesitamos más información que la de las relaciones entre los nodos. Vamos recorriendo todos los nodos generados del grafo y recorriendo sus vecinos, los ordenamos en función del grado de la arista obteniendo los de mayor grado. Con esta información generamos una matriz de similitud, pero en vez de generarla completa solo aparecen las relaciones mas similares que son las que usaremos para generar las recomendaciones, esta matriz la guardamos en un [fichero](#)²³ de salida.

Para ejecutarlo con la instrucción: `python3 05_recomendador.py`



```
soc@soc-virtual-machine: ~/hlocal/SOC/Proyecto
Archivo Editar Ver Buscar Terminal Ayuda
soc@soc-virtual-machine:~/hlocal/SOC/Proyecto$ python3 05_recomendador.py
Total aristas grafo: 1425054 Total nodos grafo: 8414
1 / 8414
2 / 8414
3 / 8414
4 / 8414
5 / 8414
6 / 8414
7 / 8414
8 / 8414
9 / 8414
10 / 8414
11 / 8414
12 / 8414
13 / 8414
14 / 8414
15 / 8414
16 / 8414
17 / 8414
18 / 8414
19 / 8414
20 / 8414
21 / 8414
```

²⁰ https://github.com/victordpc/SOC/blob/master/Proyecto/05_recomendador.py

²¹ <https://networkx.github.io/>

²² <https://github.com/victordpc/SOC/blob/master/Proyecto/Files/AristasGrafo.csv>

²³ <https://github.com/victordpc/SOC/blob/master/Proyecto/Files/Recomendaciones.csv>

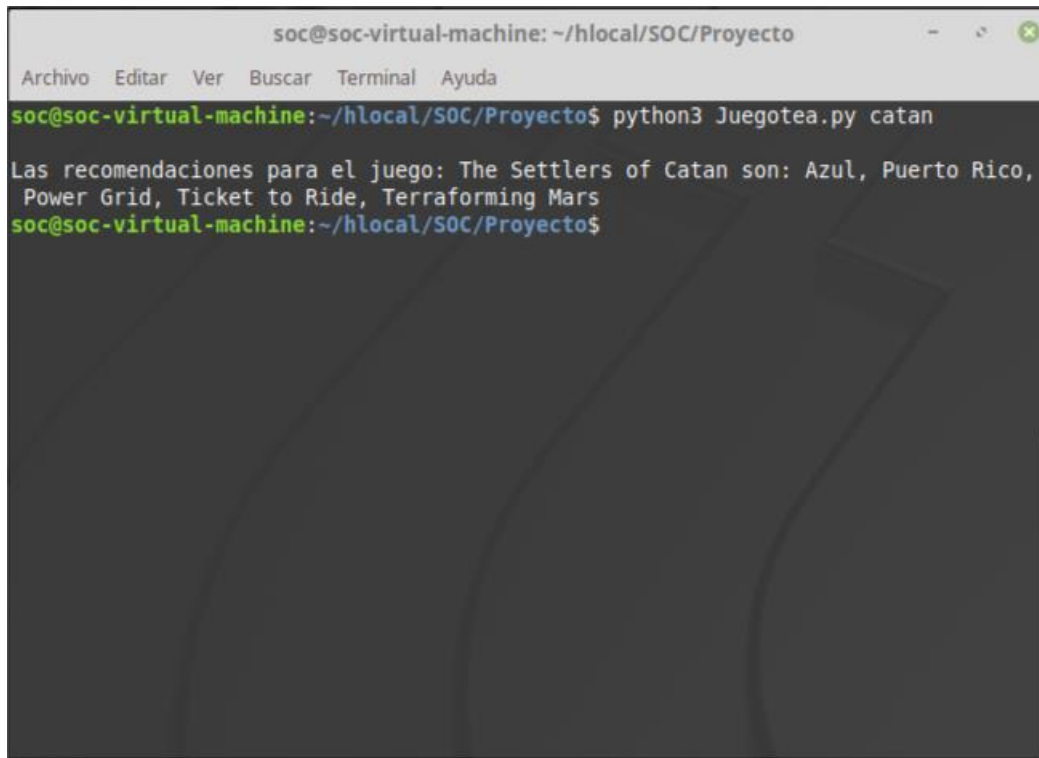
Ejecución del sistema de recomendación

La interfaz con el sistema de recomendación es mediante el script [Juegotea.py](https://github.com/victordpc/SOC/blob/master/Proyecto/Juegotea.py)²⁴. Este script debe recibir como parámetros de línea de comandos uno o varios juegos sobre los que se quiere obtener la recomendación, para obtenerlas se carga el fichero de salida del proceso de aplicar el algoritmo y el fichero de los nodos con la información de los juegos.

Para cada uno de las entradas se realiza una búsqueda en la lista de juegos de los que se tiene información y si se encuentra se busca en la lista de juegos recomendados, en caso de tener varias entradas se combinan los resultados y en caso de aparecer varias veces el mismo juego se suman los pesos asociados, tras esto se eliminan posibles ocurrencias en las recomendaciones de los juegos de entrada, se ordenan los resultados y se seleccionan los primeros para mostrar al usuario de salida, si no se encuentra un juego entre los que se disponen información se avisa al usuario de que no se dispone de datos para mostrar.

Si se ejecuta con el comando -h se muestra la ayuda.

Para ejecutarlo con la instrucción: `python3 Juegotea.py [arg ...][-h]`

A screenshot of a terminal window titled 'soc@soc-virtual-machine: ~/hlocal/SOC/Proyecto'. The terminal shows the command 'python3 Juegotea.py catan' being executed. The output is 'Las recomendaciones para el juego: The Settlers of Catan son: Azul, Puerto Rico, Power Grid, Ticket to Ride, Terraforming Mars'. The prompt 'soc@soc-virtual-machine:~/hlocal/SOC/Proyecto\$' is visible at the bottom of the terminal output.

```
soc@soc-virtual-machine: ~/hlocal/SOC/Proyecto
Archivo  Editar  Ver   Buscar  Terminal  Ayuda
soc@soc-virtual-machine:~/hlocal/SOC/Proyecto$ python3 Juegotea.py catan
Las recomendaciones para el juego: The Settlers of Catan son: Azul, Puerto Rico,
Power Grid, Ticket to Ride, Terraforming Mars
soc@soc-virtual-machine:~/hlocal/SOC/Proyecto$
```

²⁴ <https://github.com/victordpc/SOC/blob/master/Proyecto/Juegotea.py>

```
soc@soc-virtual-machine: ~/hlocal/SOC/Proyecto
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
soc@soc-virtual-machine:~/hlocal/SOC/Proyecto$ python3 Juegotea.py scythe agricola
la NoExisto

Las recomendaciones para los juegos: Scythe, Agricola son: Terraforming Mars, The Castles of Burgundy, Terra Mystica, Pandemic Legacy: Season 1, Azul

No se ha encontrado información para el juego: NoExisto
soc@soc-virtual-machine:~/hlocal/SOC/Proyecto$
```

Observaciones finales y conclusiones

Durante el desarrollo de este trabajo hemos aprendido a lidiar con los problemas externos provocados por consumir datos de los que no tenemos control y hemos tenido que abandonar varias líneas de trabajo por encontrar demasiadas dificultades.

Además, hemos tenido que ajustar los datos de entrada ya que en las primeras pruebas para obtener los datos obteníamos mas de 300.000 juegos y 5 millones de aristas que nos resultaba imposible de tratar y hemos tenido que reducir los datos de entrada.

Además, con el tipo de algoritmo que hemos empleado si los datos de entrada son pocos se promocionan los juegos que hacen de Hubs, por esto sería bueno ampliar el sistema de recomendación para añadir más juegos y cambiar el sistema de entrada de información para poder tener mas información de entrada de forma mas sencilla, por ejemplo se podría ampliar para dando un usuario de la red social obtener sus datos y usarlos para poder refinar más las búsquedas.

Como conclusión creemos que es una línea de trabajo que tiene futuro ante la inexistencia de este tipo de sistemas.

Bibliografía

boardgamegeek. (s.f.). Obtenido de https://boardgamegeek.com/wiki/page/BGG_XML_API

boardgamegeek. (s.f.). Obtenido de https://boardgamegeek.com/wiki/page/BGG_XML_API2

CARO MARTÍNEZ, M. (2017). *SISTEMAS DE RECOMENDACIÓN BASADOS EN TÉCNICAS DE PREDICCIÓN DE ENLACES PARA JUECES EN LÍNEA*.

<http://docs.python-requests.org/en/master/>. (s.f.).

<https://networkx.github.io/documentation/stable/>. (s.f.).