Universidad Nacional Autónoma De México
Facultad De Ciencias
Inteligencia Artificial, 2018-II

# Proyecto 3:

*Sistema de Recomendaciones*

## Fecha de entrega:

6 de Mayo del 2018

## Integrantes:

Aguilera Pacho, Franco Eduardo
Martínez Flores, Jorge Yael
Muñiz Patiño, Andrea Fernanda
Sánchez Morales, Rodrigo Alejandro

# Introducción

Ofrecer una cantidad muy grande de productos a muchos clientes es un reto que poco a poco se ha facilitado con la tecnología, ya que las empresas y/o distribuidoras pueden llegar a todo tipo de público de manera fácil y rápida. Pero esto a la vez crea nuevos problemas, *¿Cómo ofrecerle al consumidor el producto adecuado? ¿Cómo retener a un cliente?*, puesto que también la tecnología abre el campo de competencia a un grupo mayor. Si no proporcionamos una atención "personalizada" podemos perder al cliente, optando este por otra plataforma.

Antes de que Internet existiera, conseguir información sobre un producto se limitaba a la publicidad y/o al círculo social cercano. Actualmente, tenemos una saturación enorme de información sobre productos y también de proveedores que ofrecen los productos y/o servicios.

Como ejercicio didáctico, se propone al lector imaginar que es parte de una empresa que se dedica a ofrecer productos con ciertas características a un mercado en especial o general. Si nuestro usuario no encuentra de manera rápida algo que le agrade, se podría potencialmente perder a un cliente, esto debido a la tardanza en satisfacer su necesidad. Por ello, con información implícita y explícita los sistemas hacen recomendaciones. Con el único propósito de obtener más ingresos en la venta del producto que ofrecen, o en que el usuario siga utilizando la plataforma.

# ¿ Cómo se soluciona ?

Un **sistema de recomendaciones (*SR*)** es una herramienta que genera recomendaciones sobre un determinado objeto de estudio, a partir de las preferencias y opiniones dadas por los usuarios. Los *sistemas de recomendación* son muy útiles para evaluar y filtrar la gran cantidad de información disponible en la web y/o bases de datos, con el objetivo de asistir a los usuarios en sus procesos de búsqueda y recuperación de información.

Considerando los puntos anteriores, haremos uso de un sistema de recomendaciones para proporcionarle al usuario sugerencias de películas que probablemente puedan gustarle y/o agradarle. Es decir, con base a las preferencias del espectador, el *SR* dará un subconjunto de películas representantes de gustos similares, guiándonos principalmente por el género de las películas anteriormente indicadas como favoritas.

# Sobre el agente

Para atacar el problema usaremos un agente basado en conocimiento. Los agentes de este tipo trabajan con información genérica de forma flexible apoyándose de la lógica. Estos agentes usan una **base de conocimiento**, la cual se representa por medio de aserciones sobre el entorno, a esto le llamamos el **lenguaje de representación del conocimiento**. La flexibilidad de este agente se ve reflejada a la hora de modificar la información de su base de conocimiento, así como la manera en que se construye, sólo a partir de lo que se necesita saber de antemano. Puesto que en el problema del sistema de recomendaciones es necesario interpretar información en un formato más general, usando solo el conocimiento recibido, en este caso, los gustos como

nuestra **base de conocimiento**, un agente como el previamente descrito es nuestra mejor opción. Esta base recibe los datos percibidos del agente, los cuales después son usados para "inferir" nuevo conocimiento, sin salirse de los parámetros dados por datos previos. Es decir, nuestro agente no actúa de manera arbitraria, en cambio, utiliza información explícita, según la información proporcionada por el usuario del sistema se determina qué película es más probable que le agrade. Esto considerando las características particulares que clasifican y las dividen en subconjuntos en cuestión de género y otros aspectos.

| Agente | Rendimiento | Entorno | Actuadores | Sensores |
|---|---|---|---|---|
| Basado en conocimiento. | Proporcionar al usuario la mejor recomendación posible, de acuerdo a sus gustos. | Base de datos. Vista. | Función que calcula la distancia entre vectores. | Variable booleana, que detecta si una película le gusta al usuario. |

Cuadro 1: REAS.

# Entorno de trabajo

Las combinaciones de preferencias que puede seleccionar el usuario final son finitas. Debido a cómo se implementó el sistema de recomendación, si se escoge una combinación $A$ de películas, las cuales le gustaron al usuario, el valor siempre será el mismo cuando se escoja esa misma combinación $A$, por lo cual es determinista.

Podemos apreciar que si tenemos un usuario al que le gusta un conjunto $S$ de películas, las posibles recomendaciones pueden ser las del conjunto $R'$, supongamos que la película $\beta$ pertenece al conjunto $R'$, es decir, $\beta \in R'$ y resulta que la película $\beta$ le gusta al usuario y lo indica, el programa recalculará los valores para dar un nuevo conjunto de recomendaciones. Por lo cual consideramos que el entorno de trabajo es episódico, pues los cálculos de las recomendaciones cambiarán con cada nueva película que se agregue a las preferidas, esperando a la información que el usuario proporcione. Nuestro entorno es parcialmente observable, ya que solo sabemos un pequeño conjunto de películas que le gustan, el SR cuenta con 9 000 películas aproximadamente y la muestra que se usa para la recomendación es demasiada pequeña. Además, no es práctico que el usuario dé toda su lista de gustos sobre el tema. Sin embargo esto es subjetivo según lo considere el creador y/o cliente.

| Entorno de trabajo | Observable | Determinista | Episódico | Discreto | Agente |
|---|---|---|---|---|---|
| *SR* | Parcialmente | Sí | Sí | Sí | Individual |

Cuadro 2: Entorno de trabajo

# Sobre la implementación

Nuestro sistema de recomendaciones es sobre películas, el *SR* hará las recomendaciones según las características más importantes que se consideran en una película, como lo es el género al que pertenecen, pues la temática de la película se puede encerrar en esa característica.

Los sistemas de recomendación, como todos los sistemas de inteligencia artificial, se alimentan de datos, por lo cual se usan alrededor de 9 000 películas, información obtenida de *MovieLens*, datos recopilados por GroupLens, (no se garantiza que estén películas recientes) los cuales están clasificadas por géneros, donde una película puede tener varios géneros. La información de MovieLens se colocó en la base de datos, de la cual se "extraen" los géneros a los que pertenece cada película, para hacer el cálculo del vector y con ello, los valores como frecuencia y distancia, para hacer una selección de películas para recomendar.

Se establece el uso de un esquema de representación declarativo relacional que se basa en conocimientos, se ha optado por basarnos en conocimiento, para así tener más opciones que recomendar y usar solo un agente.

## Sobre el usuario

Existen dos casos para hacer las debidas recomendaciones, tenemos al *Usuario inicial* y *Usuario*. Simplemente al *usuario inicial* se le va a recomendar las películas más vistas por el conjunto de usuarios que ya están en la plataforma, al no tener datos sobre el usuario, lo mejor que el agente puede hacer es recomendar "tendenciasïnferidas por datos de otros usuarios en el sistema. Sin embargo en nuestra implementación no se considerara este tipo de usuario. Nuestro *SR* usa información explicita, dicho de otra manera, recabamos información que el usuario puede proporcionar por lo cual tenemos una vista para que el usuario pueda ingresar las películas que le gustan y las que no, considerando las opciones seleccionadas, se hace el debido cálculo para poder hacer la recomendación, dándole al usuario las posibles películas que pudieran gustarle. Lo que conocemos del usuario sólo es esa pequeña porción de películas que él mismo ingreso.

# Especificaciones técnicas

| **Concepto** | **Herramienta** | **Versión** |
|---|---|---|
| Lenguaje | Java | 1.8.0 |
| Manejador de bases de datos | PostgreSQL | 9.6 |
| IDE | Netbeans IDE | 8.2 |

# Sobre el código

Para el filtrado de información usamos el género de la cada película, según el género o géneros favoritos del usuario filtraremos las películas que coincidan con este. Nuestro *SR* esta basado absolutamente en *Items*, puesto que nuestro sistema no guarda nada de información de los usuarios, simplemente recibe la información hace el calculo y muestra las recomendaciones.

De manera breve podemos exponer el papel de las diferentes variables que se utiliza en el sistema de recomendaciones, dicho de otra forma el agente usa los siguientes recursos:

**Map**: Estructura de datos que almacena dos valores relacionados, Llave y valor, puede acceder a cualquier elemento de la estructura Map en tiempo constante, evita tener que recorrer toda la estructura en tiempo lineal.

**Dataset**: Conjunto de datos utilizados para el proyecto, en este caso una base de datos de películas.

**Variables y funciones**

**Genre Frequency** : La frecuencia de los géneros, cada película tiene una estructura HashMap con los géneros como llave y valores 0 por defecto, cuando el Genre Frequency (GF) se calcula esta encuentra todas las películas que cumplan con un género y almacena la cuenta por género.

**Inversed Genre Frequency**: Calcula la relevancia de los géneros en todo el dataset. Para el conjunto de películas obtenido en el Genre Frequency realiza el cálculo de log10(N / GF) donde N es el número de películas y GF el valor actual de las películas según su género, es decir el número de películas por cada género, este valor también se calcula para género de películas en el dataset.

**Term Frequency**: Frecuencia de términos, se calcula para género si este existe en la película entre la raíz cuadrada de los géneros totales, esto para calcular que tanto abarca de un género una película, entre mas sean los géneros de una película, menor será su frecuencia de término.

**User Movies**: Como es necesario almacenar información para todas las películas respecto a un usuario, cada usuario lleva una HashMap con las películas del dataset con valores igual a 0, esto significa que no le gusta ni le disgusta une película, cuando el usuario agrega una película esta se cambia a valor 1 si le gusta o -1 si le disgusta, este valor será utilizado mas adelante.

**User Profile**: Perfil calculado de los gustos del usuario por género este se calcula haciendo una suma de productos entre las userMovies y las Term Frequencies por género, es decir, multiplica entrada por entrada de los cada geénero de la película y si le gusta o no y al final suma todos los valores multiplicados, esto se hace para cada género y determina si le gusta o no al usuario una película. Entre mayor sean los valores mas estará un usuario interesado por ese género.

**User Prediction**: Para calcular que tanto sería el interés de un usuario por una película, se calcula una suma de producto entre 3 estructuras de datos. Entre la TermFrequency, el userProfile, y la Inversed Genre Frequency, estos valores se calculan y se suman para cada película, entre mas alto sea el valor, mas probable es que le gusta le película al usuario.

Estas operaciones se usan para obtener la mejor recomendación posible, el esquema que se utiliza sirve para obtener en base a los items las recomendaciones pues se hacen cálculos para ver la distancia de una película a otra, según el gusto del usuario. Se ordena de mayor a menor el tamaño del angulo para luego mostrar las recomendaciones.

# Análisis de complejidad

### Complejidad en espacio

La complejidad en espacio está determinada por la base de datos, en la cual tenemos todas las películas que se contemplan para hacer el cálculo para la recomendación. La complejidad está determinada por el número de películas que se usarán, también se debe considerar cuántas características tiene la película. Entonces la complejidad real está determinada por $O(nm)$ donde $n$ es el número de películas que considerará el sistema de recomendaciones y $m$ la cantidad de características que tiene la película.
El sistema no guarda los datos del usuario, simplemente se ingresan las preferencias y con base a eso se hace la recomendación, sin guardar nada de manera persistente, esto no ocupa memoria a considerar. Se decidió esto, debido a la implementación del agente en el sistema.

En nuestro *SR* se consideraran 9 125 películas para hacer la recomendación al usuario y cada una tiene 3 características esenciales, ya que cada film tiene distintos géneros, por lo cual la complejidad es $9125 \cdot 3$ que vendría siendo $O(27375)$ en términos de memoria.

### Complejidad en tiempo

La complejidad teórica de un *SR* dependerá únicamente de la técnica de filtrado de información y de como se implementa el agente. Por depender de esto no queda nada mas que mencionar sobre la complejidad teórica.

Eso nos lleva al análisis de la técnica en especifico que hemos elegido para este *SR*.
Notemos que complejidad en tiempo se mejora notablemente gracias a que se usan maps, convirtiendo la consulta del dato en tiempo constante.

Para agregar las películas favoritas usamos el siguiente código

```
while(salida){
        System.out.println("Agregando peliculas que NO te gustan");
        System.out.print("Introduce el titulo de la pelicula: ");
        String j = sc.nextLine();
        usuario.addMovie(j, false);
        System.out.print("Quieres seguir agregando peliculas? [Si / No] ");
        String str = sc.nextLine();
        str = str.toUpperCase();
        if(str.equals("NO") || str.equals("N"))
            salida = false;
}
```

El cual claramente tiene complejidad lineal $O(n)$.

El método *calculateUserPrediction* es el siguiente a considerar para la complejidad en cuestión de tiempo:

```
while(resultSet.next()){
                int movieId = resultSet.getInt("movieId");
                String titulo = resultSet.getString("titulo");
                String genero = resultSet.getString("genero");
                // Crea la pelicula
                Movie movie = new Movie(movieId, titulo, genero);
                movie.calculateTF();

                // Term Frequency de cada pelicula
                HashMap<String, Double> TF = movie.getTF();

                // Itera sobre los generos de la pelicula
                for(HashMap.Entry<String, Double> entry : userProfile.entrySet()){
                    // Valor del TF
                    double TFvalue = TF.get(entry.getKey());
                    // Valor del IGF
                    double IGFValue = IGF.get(entry.getKey());
                    // Valor del gusto calculado del usuario
                    double userValue = entry.getValue();
                    // Calcula el Sumproduct de los 3 valores para todos
                    // los generos
                    resultado += TFvalue * IGFValue * userValue;
                }
                // Almacena el resultado del calculo con el id para encontrarlo
                result.put(resultado, movieId);
                resultado = 0;
}
```

El otro punto a considerar sería la operación:

```
resultado += TFvalue * IDFValue * userValue;
```

Esto por que "es la operación mas compleja del método" puesto que se hace tantas veces como se itera sobre cada género de cada película.

Otra complejidad que se podría tomar cuenta es la del método *descendingMap* pero por ser este un método ya definido en JAVA podremos suponer que esta optimizado y no considerarlo para nuestra complejidad.
Establecidos ya los puntos y el código que se encarga de hacer los cálculos importantes podemos decir que la complejidad es de orden $O(n^2)$ en el peor de los casos.

# ¿Qué hace nuestro sistema?

A continuación analizaremos un caso en especifico de la forma en la que responde nuestro sistema de recomendaciones.
En la siguiente imagen se muestran las películas que le gustan al usuario hipotético $A$.

```
Agregando peliculas que te gustan
Introduce el título de la película: Toy Story (1995)
¿Quieres seguir agregando películas? [Si / No] si
Agregando peliculas que te gustan
Introduce el título de la película: Mulan (1998)
¿Quieres seguir agregando películas? [Si / No] si
Agregando peliculas que te gustan
Introduce el título de la película: Digimon: The Movie (2000)
¿Quieres seguir agregando películas? [Si / No] si
Agregando peliculas que te gustan
Introduce el título de la película: Valiant (2005)
¿Quieres seguir agregando películas? [Si / No] no
Agregando peliculas que NO te gustan
Introduce el título de la película: Casino (1995)
¿Quieres seguir agregando películas? [Si / No] si
Agregando peliculas que NO te gustan
Introduce el título de la película: Bad Boys (1995)
¿Quieres seguir agregando películas? [Si / No] si
Agregando peliculas que NO te gustan
Introduce el título de la película: Kiss of Death (1995)
¿Quieres seguir agregando películas? [Si / No] Si
Agregando peliculas que NO te gustan
Introduce el título de la película: Marathon Man (1976)
¿Quieres seguir agregando películas? [Si / No] no
```

Figura 1: Películas favoritas del usuario $A$

Como podemos observar el usuario a elegido películas de los géneros *Animation y Children* como películas que le agradan, y las que no le gustan son del género *Crime y Drama*. El sistema de recomendaciones obtiene la información y con base en esta debe ir haciendo las recomendaciones en consideración a las películas de agrado.

En la imagen 2 se muestra las recomendaciones que hace el sistema a partir de las películas anteriormente mencionadas.

Observemos que las películas que se recomiendan son de los géneros *Children y Animation*, siendo estas distintas a las que el usuario nos indico que le gustaban, por ejemplo; *Dragon Hunters* y *The Good Dinosaur*.

Dadas las observaciones anteriores podemos concluir que el $SR$ da recomendaciones dentro de lo esperado, es decir que el parámetro que tienen estas coinciden con los gustos del usuario.

```
Películas que te pueden gustar:
ID: 7140 TÍTULO: Ponyo (Gake no ue no Ponyo) (2008)
GÉNEROS: Adventure|Animation|Children|Fantasy

ID: 7437 TÍTULO: Dragon Hunters (Chasseurs de dragons) (2008)
GÉNEROS: Adventure|Animation|Children

ID: 7441 TÍTULO: Rudolph, the Red-Nosed Reindeer (1964)
GÉNEROS: Adventure|Animation|Children|Fantasy|Musical

ID: 8929 TÍTULO: The Good Dinosaur (2015)
GÉNEROS: Adventure|Animation|Children|Comedy|Fantasy

ID: 7183 TÍTULO: Rock-A-Doodle (1991)
GÉNEROS: Adventure|Animation|Children|Musical

ID: 9080 TÍTULO: Ice Age: The Great Egg-Scapade (2016)
GÉNEROS: Adventure|Animation|Children|Comedy

ID: 1653 TÍTULO: Return of Jafar, The (1994)
GÉNEROS: Adventure|Animation|Children|Fantasy|Musical|Romance

ID: 8196 TÍTULO: Rise of the Guardians (2012)
GÉNEROS: Adventure|Animation|Children|Fantasy|IMAX

ID: 8855 TÍTULO: The Brave Little Toaster Goes to Mars (1998)
GÉNEROS: Animation|Children

ID: 7744 TÍTULO: Gnomeo & Juliet (2011)
GÉNEROS: Adventure|Animation|Children|Comedy|Fantasy|Romance

ID: 8799 TÍTULO: Rudolph the Red-Nosed Reindeer (1948)
GÉNEROS: Animation|Children|Fantasy
```

Figura 2: Recomendaciones del agente.

# Ventajas y Desventajas

Una desventaja clara de un sistema de recomendaciones es la cantidad de información que se necesita para hacer una recomendación adecuada y certera. Es necesaria una base de datos para poder implementarlo, por lo cual el recurso de memoria es un factor en desventaja para un *SR*.

El problema al involucrar gustos, una abstracción humana, no tiene forma de hacer recomendaciones que siempre sean satisfactorias, ya que los gustos pueden ser demasiado ambiguos, incluso dentro de la mente de una misma persona.

Una ventaja es que el agente se puede implementar de acuerdo a las necesidades o según el producto, ya que hay muchas formas de hacer las recomendaciones, por ejemplo; Tenemos lo que son los *amigos virtuales*, supongamos que un sujeto $A$ ve la película $\alpha$, $\beta$ y $\pi$. Supongamos que llega un usuario $B$ nuevo al *SR*, el usuario $B$ ve las películas $\alpha$ y $\beta$ por las colisiones que tienen estos dos usuarios el *SR* procederá a recomendar la película $\pi$ al usuario $B$. Esta es otra forma de implementar el agente para hacer las recomendaciones, teniendo un polimorfismo en el agente.

### En la aplicación

Una clara ventaja de un *SR*, es que bien implementado el sistema puede ayudar a mantener al cliente y satisfacer sus necesidades y con ello mejorar las ganancias de la empresa que está haciendo uso de dicho sistema.

## Conclusiones

Sin duda alguna implementar un *sistema de recomendaciones* no es tarea sencilla, además de que la implementación debe ser muy personalizada ya que no será lo mismo un *SR* sobre películas que uno de libros o música, pueden tener características similares, pero los parámetros a considerar cambian según el producto a ofrecer. También pueden usarse algunas estrategias en los diferentes productos, pero los valores de la entrada de la función cambian y esto es sólo considerando nuestra solución, pues existen varias formas de implementarlo.

Para crear un sistema de recomendaciones tenemos distintas tácticas como: usar multi-agentes, un único agente, distintos filtros de información y gracias a esto podemos acomodar la técnica al *SR* y no el *SR* a las tácticas. El cómo y qué tan bien se haga determinará qué tan bueno será el *SR*. También debemos de notar que en cuanta más información tenemos en nuestra base de datos, se tienen más posibilidades de éxito y esto es gracias al agente basado en conocimiento, puesto que con algún otro agente no sería posible determinar la preferencia.

### Bibliografía


- Russell, S. & Norman, P.. (2009). Artificial Intelligence: A Modern Approach. Upper Saddle River: Prentice Hall.
- Herrera-Viedma & Porcel, & Hidalgo. Sistemas de recomendaciones: herramientas para el filtrado de información en Internet [en linea]. "Hipertext.net", núm. 2, 2004. Consultado: 26 de Abril del 2018
  `https://www.upf.edu/hipertextnet/numero-2/recomendacion.html`
- Desconocido (2006) Introducción a la representación de conocimiento. Notas de Inteligencia Artificial. Recursos de moodle.
- Fernandez, E. L.. (21 de Marzo de 2018). ¿Qué es una recomendación en la era de la inteligencia artificial?. BBVA. Consultado el 3 de mayo del 2018. Recuperado de `https://www.bbva.com/es/recomendacion-inteligencia-artificial/`
- Tavish, A. (7 de Abril del 2015). Information Retrieval System explained in simple terms!. Analytics Vidhya. Consultado el 2 de Mayo del 2018. Recuperado de `https://www.analyticsvidhya.com/blog/2015/04/information-retrieval-system-explained/`
- Shuvayan, D. (11 de Agosto del 2015). Beginners Guide to learn about Content Based Recommender Engines. Analytics Vidhya. Consultado el 2 de Mayo del 2018 Recuperado de `https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-re`