



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**Sistema de Recomendación  
basado en Aprendizaje  
Profundo**



Presentado por Raúl Negro Carpintero  
en Universidad de Burgos — 30 de junio  
de 2019

Tutor: Bruno Baruque Zanón







UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



D. Bruno Baruque Zanón, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Raúl Negro Carpintero, con DNI 71305764Z, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 30 de junio de 2019

Vº. Bº. del Tutor:

Vº. Bº. del co-tutor:

D. nombre tutor

D. nombre co-tutor





## **Resumen**

El tema principal son los sistemas de recomendación. El objetivo de este proyecto es comparar el rendimiento entre los modelos clásicos y los modelos basados en aprendizaje profundo sobre diferentes conjuntos de datos.

## **Descriptores**

Sistemas de recomendación, aprendizaje profundo, redes neuronales.

**Abstract**

The main topic are the recommendation systems. The goal of this project is to compare the performance between classic models and those based on deep learning on different datasets.

**Keywords**

Recommendation systems, deep learning, neural networks.



---

# Índice general

---

<b>Índice general</b>	<b>III</b>
<b>Índice de figuras</b>	<b>V</b>
<b>Índice de tablas</b>	<b>VI</b>
<b>Introducción</b>	<b>1</b>
<b>Objetivos del proyecto</b>	<b>5</b>
2.1. Objetivos técnicos . . . . .	5
2.2. Objetivos personales . . . . .	5
<b>Conceptos teóricos</b>	<b>7</b>
3.1. Sistemas de recomendación . . . . .	7
3.2. Medidas de calidad . . . . .	8
3.3. Tratamiento de los datos . . . . .	10
<b>Técnicas y herramientas</b>	<b>13</b>
4.1. Metodologías . . . . .	13
4.2. Control de versiones . . . . .	13
4.3. Repositorio . . . . .	14
4.4. Gestión del proyecto . . . . .	14
4.5. Entorno de Desarrollo Integrado (IDE) . . . . .	14
4.6. Librerías . . . . .	15
4.7. Datasets . . . . .	16
<b>Aspectos relevantes del desarrollo del proyecto</b>	<b>19</b>
5.1. Metodologías . . . . .	19

5.2. Formación . . . . .	19
5.3. Desarrollo del código . . . . .	20
5.4. CUDA . . . . .	21
5.5. Problemas . . . . .	21
5.6. Documentación . . . . .	22
<b>Trabajos relacionados</b>	<b>23</b>
6.1. Artículos . . . . .	23
6.2. Proyectos . . . . .	25
<b>Conclusiones y Líneas de trabajo futuras</b>	<b>27</b>
<b>Bibliografía</b>	<b>29</b>

---

# Índice de figuras

---

5.1. Esquema del patrón MVC [8] . . . . .	21
---	----

---

## Índice de tablas

---

---

# Introducción

---

Los sistemas de recomendación son herramientas fundamentales de las cuales se aprovechan las principales compañías que ofrecen servicios para proveer a los usuarios de ítems relevantes.

Compañías como *Amazon*, *Netflix*, *Spotify* y *YouTube* utilizan la información de sus usuarios y de sus productos; así como las valoraciones que los propios usuarios hacen sobre los productos que consumen con el fin de crear modelos de recomendación capaces de ofrecer a estos usuarios ítems que les serán de interés la próxima vez que interactúen con el servicio.

Tal es la importancia de los sistemas de recomendación que, por ejemplo, *Netflix*, *Spotify* y *Trivago* organizan competiciones en las que piden mejorar sus propios sistemas [27], [22], [23]. El premio que ofreció *Netflix* por mejorar su modelo un 10 % fue de 1.000.000 de dólares.

Como se verá más adelante, existen varios tipos de sistemas de recomendación, siendo los más importantes:

- Modelos colaborativos
- Modelos basados en contenido
- Modelos híbridos

Los **modelos colaborativos** solo tienen en cuenta las valoraciones que los usuarios hacen sobre los ítems. Así, lo que hace es buscar usuarios parecidos (usuarios que valoran de forma parecida los mismos ítems) para recomendar ítems que uno no ha visto y el otro sí.

Los **modelos basados en contenido** tienen en cuenta las características de los ítems. De esta manera, buscan ítems con características parecidas

a las de los ítems con los que ya ha interactuado el usuario positivamente. También hacen uso de las características de los usuarios.

Los **modelos híbridos** son una mezcla de los dos anteriores. Tienen en cuenta tanto las valoraciones como las características de usuarios e ítems.

En este proyecto serán objeto de estudio los modelos de recomendación clásico y los modelos de recomendación basados en aprendizaje profundo. Así pues, se evaluará cada modelo siguiendo unas métricas comunes a los dos para poder compararles.

## Estructura de la memoria

La memoria se divide en:

- **Introducción:** en este apartado se desarrolla de manera breve el tema que se va a tratar en el proyecto, así como la estructura del propio proyecto y los materiales entregados.
- **Objetivos del proyecto:** sección en la que se indican los objetivos que se persiguen con la realización del proyecto, tanto técnicos como personales.
- **Conceptos teóricos:** apartado en el que se explica todo lo necesario para el correcto entendimiento del tema tratado en el proyecto.
- **Técnicas y herramientas:** sección en la que se listan todas las herramientas usadas en el proyecto, así como una breve justificación de su uso en favor de otras herramientas existentes.
- **Aspectos relevantes del desarrollo del proyecto:** apartado en el que se explican temas de especial importancia.
- **Trabajos relacionados:** sección en la que se indican y se desarrollan brevemente tanto artículos como proyectos que están directamente relacionados con este proyecto.
- **Conclusiones y Líneas de trabajo futuras:** apartado en el que se recogen las conclusiones obtenidas una vez finalizado el proyecto, y las posibles mejoras que se pueden hacer en el futuro.

## Materiales adjuntos

Adicionalmente, junto a la memoria también se proporcionan los siguientes anexos:

- **Plan de Proyecto Software:** en este apéndice se desarrolla la planificación temporal que se ha seguido durante la realización del proyecto y la viabilidad del mismo.
- **Especificación de Requisitos:** en este apéndice se indican y se explican los requisitos funcionales obtenidos a partir de los objetivos generales del proyecto.
- **Especificación de diseño:** en este apéndice se explica cómo se ha estructurado el proyecto y cómo se han manipulado los datos de entrada para poder ser usados por los sistemas.
- **Documentación técnica de programación:** esta sección contiene todo lo relacionado con la programación y la ejecución del proyecto.
- **Documentación de usuario:** apéndice en el que se recoge todo lo que el usuario que va a utilizar la aplicación debe saber.





---

# Objetivos del proyecto

---

## 2.1. Objetivos técnicos

- Comprender el funcionamiento de un sistema de recomendación clásico.
- Recoger y evaluar los resultados obtenidos por el modelo clásico.
- Comprender el funcionamiento de un sistema de recomendación basado en aprendizaje profundo.
- Recoger y evaluar los resultados obtenidos por el modelo basado en aprendizaje profundo.
- Comparar los resultados obtenidos por ambos modelos sobre una serie de conjuntos de datos.
- Desplegar el programa Python en un servidor y hacer posible la interacción con nuevos usuarios.

## 2.2. Objetivos personales

- Estudiar los sistemas de recomendación debido a su gran importancia.
- Poner en uso los conocimientos adquiridos durante la carrera.
- Ahondar más en el campo del aprendizaje profundo.



---

# Conceptos teóricos

---

Los conceptos teóricos más importantes del proyecto son los relacionados con los sistemas de recomendación y las redes neuronales.

## 3.1. Sistemas de recomendación

Los sistemas de recomendación son herramientas que generan recomendaciones sobre un determinado objeto de estudio, a partir de las preferencias y opiniones dadas por los usuarios. [9]

En un sistema de recomendación hay dos clases de entidades: usuarios e ítems. Los datos están representados en una matriz de utilidad de tal manera que los usuarios son filas y los ítems columnas. Para cada par hay un valor que representa el grado de preferencia de un usuario para un ítem. Se asume que la mayoría de los valores se desconocen. Es por ello, que el objetivo de un sistema de recomendación es rellenar esos espacios en blanco.

### Tipos de sistemas de recomendación

Existen tres tipos de sistemas (o modelos) de recomendación:

- Modelos basados en contenido
- Modelos colaborativos
- Modelos híbridos

### Modelos basados en contenido

Este tipo de sistemas tienen en cuenta los gustos del usuario y las características de los ítems.

Para cada ítem hay que construir un perfil compuesto por las propiedades del mismo. Por ejemplo, si los ítem son películas, algunas de las propiedades serían género, director, fecha de estreno, reparto, etc. Una vez hecho esto, se recomiendan ítems cuyas propiedades sean parecidas a los ítems que el usuario ha valorado positivamente.

## Modelos colaborativos

Recomiendan ítems basándose únicamente en los gustos de usuarios similares.

En este caso, en lugar de usar el vector ítem-perfil de un ítem, usamos las filas de la matriz de utilidad. Los usuarios son parecidos si sus filas se acercan de acuerdo a alguna distancia.

## Modelos híbridos

Este tipo de sistemas utilizan las dos técnicas anteriores para ofrecer mejores recomendaciones.

Así pues, se recomendarán ítems que sean parecidos a los ítems valorados positivamente por usuarios parecidos.

## 3.2. Medidas de calidad

Podemos hacer uso de diferentes métricas para conocer cómo de bueno es nuestro sistema de recomendación. Dado que las dos librerías que se han utilizado en el proyecto las ha creado la misma persona, se utilizarán las métricas ofrecidas en ellas por su gran similitud. Las métricas son:

- Precisión  $k$
- Recall  $k$
- Score  $AUC$
- Ranking recíproco
- *Root Mean Square Error*

### Precisión $k$

La precisión  $k$  [12] mide el número de elementos relevantes conocidos que se encuentran en las primeras  $k$  posiciones del ranking de predicciones, es

decir, el porcentaje de coincidencias entre los elementos relevantes conocidos y los elementos devueltos por el modelo. Su fórmula es:

$$Precision\ at\ k = \frac{ERC \cap kRP}{k} \quad (3.1)$$

siendo  $ERC$  los elementos relevantes conocidos y  $kRP$  las primeras  $k$  posiciones del ranking de predicciones.

### Recall $k$

El recall  $k$  [13] la división del número de elementos relevantes que hay en las primeras  $k$  posiciones del ranking de predicciones entre el número de elementos relevantes conocidos en el conjunto de test. Su fórmula es:

$$Recall\ at\ k = \frac{ERk}{ERT} \quad (3.2)$$

siendo  $ERk$  los elementos relevantes en las primeras  $k$  posiciones del ranking de resultados y  $ERT$  los elementos relevantes del conjunto de test.

### AUC Score

El  $AUC$  (Area Under the Curve, Área Bajo la Curva) score [11] es la probabilidad de que un elemento relevante escogido aleatoriamente tenga un score mayor que un elemento no relevante escogido aleatoriamente.

### Ranking recíproco

El ranking recíproco [14] es el inverso del score del elemento más relevante devuelto en el ranking de resultados. Su fórmula es:

$$Reciprocal\ rank = \frac{1}{SER} \quad (3.3)$$

siendo  $SER$  el score del elemento más relevante devuelto en el ranking de resultados.

En el modelo de *Spotlight* aparece como  $MMR$ , o *Mean Reciprocal Rank*.

### Root Mean Square Error

El  $RMSE$  se define como el error que hay entre dos conjuntos de datos. Aplicado a nuestro caso, el  $RMSE$  compara un valor conocido con un valor

observado. Su fórmula sería la siguiente:

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (C_i - O_i)^2}{n}} \quad (3.4)$$

siendo  $C$  una recomendación conocida y  $O$  una recomendación aportada por el sistema.

### 3.3. Tratamiento de los datos

Lo más común en los sistemas de recomendación es tener una gran cantidad de usuarios y de ítems. En cambio, la cantidad de valoraciones de ítems por cada usuario que se tiene es muy pequeña. Esto hace que la matriz con la que representamos las valoraciones sea muy grande pero con muchos valores a 0 (o celdas vacías, dependiendo de la representación escogida).

Esto es lo que llamamos *matrices dispersas* [29]. Trabajar con estas matrices supone un gran coste de rendimiento, por lo que necesitamos convertirlas a otras estructuras con las que poder trabajar mejor.

En el caso de *LightFM*, las estructuras utilizadas son:

- Matrices *COO*
- Matrices *CSR*

#### Matrices COO

Las *matrices COO* [24], o matrices coordenadas, dividen la matriz dispersa en tres vectores:

- Vector con los valores no nulos
- Vector con el índice de las filas
- Vector con el índice de las columnas

Esto lo podemos ver más claro con el siguiente ejemplo. Dada la siguiente matriz dispersa

$$\begin{bmatrix} 27 & 0 & 8 & 14 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 \\ 3 & 0 & 7 & 0 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 4 & 0 & 77 & 0 & 0 & 0 \end{bmatrix}$$

obtendríamos los siguientes vectores:

- Vector de valores:

$$\begin{bmatrix} 27 & 8 & 14 & 7 & 3 & 7 & 9 & 1 & 4 & 77 \end{bmatrix}$$

- Vector de filas:

$$\begin{bmatrix} 1 & 1 & 1 & 2 & 3 & 3 & 3 & 4 & 5 & 5 \end{bmatrix}$$

- Vector de columnas:

$$\begin{bmatrix} 1 & 3 & 4 & 2 & 1 & 3 & 6 & 6 & 1 & 3 \end{bmatrix}$$

Este tipo de estructura es utilizada por *LightFM* para representar la matriz de interacciones entre usuarios e ítems.

## Matrices CSR

Otras estructuras ampliamente utilizadas, por ser más compactas que las matrices *COO*, son las matrices *CSR* [25]. Las matrices *CSR* (*Compressed Sparse Row*) no almacenan los índices de las filas, si no que almacenan punteros a los inicios de fila.

Utilizando como ejemplo la matriz dispersa del caso anterior, ahora obtendríamos:

- Vector de valores:

$$\begin{bmatrix} 27 & 8 & 14 & 7 & 3 & 7 & 9 & 1 & 4 & 77 \end{bmatrix}$$

- Vector de columnas:

$$\begin{bmatrix} 1 & 3 & 4 & 2 & 1 & 3 & 6 & 6 & 1 & 3 \end{bmatrix}$$

- Vector de filas:

$$\begin{bmatrix} 1 & 4 & 5 & 9 & 10 & 11 \end{bmatrix}$$

Este tipo de estructura es utilizada por *LightFM* para representar las matrices de features de usuarios e ítems.

## Spotlight

*Spotlight* utiliza directamente los arrays de *numpy*, aunque también ofrece métodos para transformar estos arrays a matrices *COO* y *CSR*.





---

# Técnicas y herramientas

---

## 4.1. Metodologías

### Scrum

Scrum es una técnica de desarrollo ágil que se caracteriza por: [28]

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos completados.
- Solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencia o en cascada.
- Realizar a diario una reunión con el objetivo de obtener realimentación sobre las tareas del equipo y los obstáculos que se presentan.

Al ser un proyecto educativo, las reuniones no han sido diarias, se han realizado cada dos semanas aproximadamente.

## 4.2. Control de versiones

### Git

**Git** es un software de control de versiones distribuido diseñado por el creador del kernel Linux Linus Torvalds. Su objetivo es mantener un registro de los cambios que se producen en los archivos de proyectos que normalmente están compartidos.

### 4.3. Repositorio

- Herramientas consideradas: [GitHub](#), [GitLab](#) y [Bitbucket](#)
- Herramienta escogida: [GitHub](#)

Se escoge *GitHub* por estar familiarizado con la plataforma al haber sido usada durante toda la carrera. Junto con *GitHub* se ha trabajado con *GitHub Desktop* para la realización de los commits.

### 4.4. Gestión del proyecto

- Herramientas consideradas: [ZenHub](#) y [Trello](#) y
- Herramienta escogida: [ZenHub](#)

La integración de *ZenHub* con *GitHub* es muy superior a la de *Trello*. No solo es integra directamente en la página del repositorio en *GitHub* sino que además ofrece herramientas muy útiles, como los reportes, especialmente el de *burndown*. Además, gracias a la extensión de *ZenHub* para navegador podemos ver el estado del repositorio de una manera muy sencilla y rápida.

### 4.5. Entorno de Desarrollo Integrado (IDE)

#### Python

- Herramientas consideradas: [Jupyter](#), [Spyder](#), [PyCharm](#) y [Sublime Text](#)
- Herramientas escogidas: [Jupyter](#), [Spyder](#) y [Sublime Text](#)

Durante el inicio del proyecto se utilizó *Jupyter* para obtener en sus *notebooks* modelos de recomendación iniciales sin prestar mucho detalle a la organización del proyecto y sus paquetes.

Una vez se tuvo una idea clara de cómo iba a estar estructurado el proyecto se pasó a *Spyder*, con el que podíamos tener el código y la consola en el mismo sitio.

Finalmente, se pasó a *Sublime Text* para realizar los últimos cambios en el código de los modelos y para obtener la aplicación *Flask*.

## Documentación

- Herramientas consideradas: **Texmaker** y **Apache OpenOffice**
- Herramienta escogida: **Texmaker**

Se elige *Texmaker* por encima de *OpenOffice* debido a la novedad que supone utilizar  $\text{\LaTeX}$  para crear documentos.

Además, es una herramienta muy potente gracias a los comandos y funciones que tiene, que en ocasiones hacen que sea mucho más fácil realizar una tarea con *Texmaker* que con *OpenOffice*.

## 4.6. Librerías

### NumPy

**NumPy** es una librería muy utilizada en el ámbito de la computación científica gracias a su potente manejo de arrays y a las funciones de álgebra lineal que contiene. Forma parte del ecosistema de *SciPy*.

En el proyecto se emplea *NumPy* para transformar listas de Python en arrays que puedan ser utilizados por los sistemas.

### pandas

**pandas**, al igual que *NumPy*, es una librería muy utilizada en el campo de la computación científica debido a la facilidad que ofrece para manejar datos en tablas y series. También forma parte del ecosistema de *SciPy*.

En el proyecto se emplea *pandas* para el manejo inicial de los conjuntos de datos, pasando los *.csv* a *dataframes*.

### LightFM

**LightFM** es una implementación para Python de populares algoritmos clásicos de recomendación [10]. Está creado por la misma gente que creó *Spotlight*.

Se decidió emplear *LightFM* debido a su aparente sencillez y por ser una librería bastante completa.

## Spotlight

**Spotlight** es una librería creada por la misma gente que *LightFM* que utiliza *PyTorch* para construir modelos de recomendación [16] basados en aprendizaje profundo.

Se decidió emplear *Spotlight* por su aparente sencillez y por haber sido creada por la misma gente que la librería clásica.

## Flask

**Flask** es un framework para Python con el que se obtiene una interfaz web en la que poder interactuar con el proyecto.

## tkinter

**Tkinter** es un módulo que forma parte de Python con el que podemos construir GUIs [20].

Se utiliza en el proyecto para guardar y seleccionar las matrices generadas por los modelos y los propios modelos.

## Chardet

**Chardet** es un detector de encoding universal [19].

Se puede emplear en el proyecto para obtener el encoding de los archivos *.csv*. Con ello, se logra que el usuario no necesite saber qué encoding emplea los archivos de datos que quiere utilizar.

## pickle

**pickle** es un módulo utilizado para la serialización de objetos en Python [21].

Se utiliza en el proyecto para guardar las matrices y los modelos obtenidos por los sistemas.

## 4.7. Datasets

### Anime

Este conjunto de datos [4] contiene información relativa a las preferencias de 73.516 usuarios sobre 12.294 películas o series de anime. Cada usuario ha

valorado unos 106 animes.

El conjunto de datos se encuentra disponible en: [Anime Dataset](#).

## Book-Crossing

Este conjunto de datos [31] contiene 1.149.780 valoraciones de 278.858 usuarios sobre 271.379 libros. Hay alrededor de 11 valoraciones por usuario.

El conjunto de datos se encuentra disponible en [Book-Crossing Dataset](#).

## MovieLens

Este conjunto de datos [7] contiene 100.000 valoraciones de 943 usuarios sobre 1.682 películas. Hay unas 106 valoraciones por usuario.

El conjunto de datos se encuentra disponible en los propios archivos descargados por la librería *LightFM* [15].

## Last.FM

Este conjunto de datos [3] contiene las veces que 1.892 usuarios han escuchado a 17.632 artistas. Cada usuario ha escuchado alrededor de unos 49 artistas.

El conjunto de datos se encuentra disponible en [Last.FM](#).

## Dating Agency

Este conjunto de datos [2] contiene 17.359.346 valoraciones de 135.359 usuarios sobre 168.791 perfiles de otros usuarios. Hay unas 128 valoraciones por usuario.

El conjunto de datos se encuentra disponible en [Dating Agency](#).



---

# Aspectos relevantes del desarrollo del proyecto

---

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto.

## 5.1. Metodologías

Para la gestión del proyecto se ha adaptado la metodología ágil Scrum. Las principales características han sido:

- La duración de los sprints fue de dos semanas aproximadamente.
- Los sprints finalizaban con una reunión en la que se revisaba el trabajo hecho y se planteaban las tareas del siguiente sprint.
- Se utilizó ZenHub para llevar el seguimiento de las tareas.

## 5.2. Formación

El proyecto ha requerido obtener una serie de conocimientos que no se tenían inicialmente. Se ha estudiado el campo de los sistemas de recomendación y la utilización de aprendizaje profundo en Python.

Para la formación en los sistemas de recomendación se usó el libro:

- *Mining of Massive Datasets* (Jure Leskovec, Anand Rajaraman y Jeffrey D. Ullman) [17].

Para la formación en aprendizaje profundo en Python se realizó el siguiente curso:

- *Practical Deep Learning For Coders, v3* (Jeremy Howard) [5].

Para poder usar la aplicación desde una interfaz web se siguió la documentación de *Flask* y de *WTForms*:

- *Welcome to Flask* (Pallets Team) [18].
- *WTForms Documentation* (WTForms Team) [30].

Para la obtención de los modelos de recomendación se utilizaron las siguientes librerías:

- *LightFM* (Maciej Kula) [10].
- *Spotlight* (Maciej Kula) [16].

### 5.3. Desarrollo del código

El proyecto se ha centrado en dos aspectos fundamentales: obtener un modelo clásico y obtener un modelo basado en aprendizaje profundo.

Para la obtención del modelo clásico se estudió en primer lugar hacer uso de la librería *Crab*. Esta opción se descartó debido a la cantidad de fallos que saltaron durante la instalación (no se pudo llegar a instalar) y a la falta de actividad por parte de sus autores en el repositorio (no se ha tocado desde hace 7 años). Por todo esto, se decidió utilizar la librería *LightFM*.

Para la obtención del modelo basado en aprendizaje profundo se estudió en primer lugar hacer uso de la librería de *fast.ai*, pero se descartó porque *Spotlight* parecía ser más amigable en cuanto a facilidad de uso, y porque está creado por la misma gente que creó *LightFM*. Así pues, se espera que la evaluación de los modelos siga los mismos criterios en ambas librerías.

Para la obtención de la aplicación web se utilizó *Flask* tal y como sugirió el tutor. Esta herramienta resultó ser muy fácil de utilizar y cumplió todas las funciones que se esperaban de ella. Junto con *Flask* se utilizó *WTForms* para crear todos los formularios que forman parte de la aplicación.

Todo el código del proyecto se pensó para seguir el *Modelo Vista Controlador*.



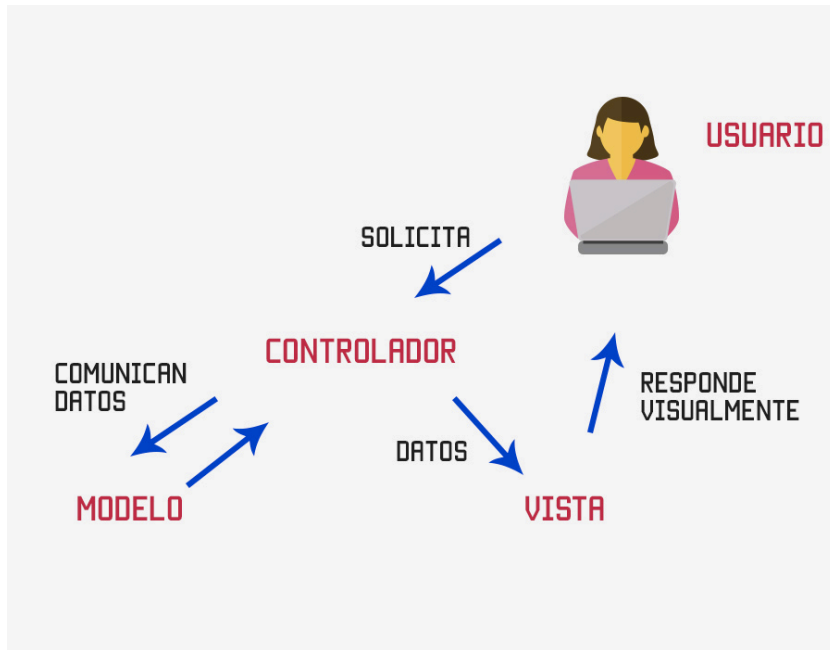


Figura 5.1: Esquema del patrón MVC [8]

## 5.4. CUDA

Para la obtención y el entrenamiento de los modelos de *Spotlight* es posible utilizar la GPU gracias a *CUDA*. Si el equipo en el que corre la aplicación no tuviera GPU, o no estuviera disponible, no pasaría nada (no saltarían errores), simplemente se ejecutarían las operaciones en la CPU.

La instalación de *Spotlight* incluye *CUDA*.

## 5.5. Problemas

Durante el desarrollo del proyecto se han encontrado los siguientes problemas:

### Predicciones en modelo de secuencia

De momento, no se pueden obtener las predicciones para el modelo de secuencia de *Spotlight*. Esto es debido a que a diferencia del resto de modelos, donde para ver las predicciones basta con indicar el usuario cuya predicciones quieres obtener, en el modelo de secuencia no vale el usuario.

Hay que utilizar una secuencia de interacciones del usuario. Esto complica el trabajo a la hora de programar además de complicarle la vida al usuario que va a interactuar con la aplicación.

### **Añadir valoraciones al modelo de secuencia**

También, y de momento, hay problemas a la hora de añadir valoraciones y/o usuarios al modelo de secuencia por lo comentado anteriormente.

Es posible que esto se tenga que dejar como línea de trabajo futuro.

## **5.6. Documentación**

Las opciones que se plantearon para realizar la documentación fueron *Apache OpenOffice* y *Texmaker*. Ya que el Trabajo Fin de Grado se puede ver como una forma de aprender conceptos nuevos que se desmarcan de lo visto durante la carrera, se optó por *Texmaker* debido a la novedad y al querer aprender y usar  $\text{\LaTeX}$ .

Además, la documentación se apoya en el programa *JabRef* [1], con el que se va guardando un registro con las entradas *bibtex* de los artículos y demás elementos que se han consultado a lo largo del proyecto.

---

## Trabajos relacionados

---

Actualmente los sistemas de recomendación y, en general todo lo relacionado con *big data*, está a la orden del día. Cada vez que usamos *Google*, *Facebook*, etc., podemos observar la gran cantidad de anuncios que nos recomiendan productos o servicios que, casualmente (a veces rozando la legalidad [6]), están relacionados con nuestros gustos.

Centrándonos en los sistemas de recomendación, puede que este campo tuviera su *boom* con la competición de *Netflix* entre 2006 y 2009 [27]. A partir de entonces, se han ido sucediendo competiciones en las que empresas muy importantes y conocidas ponen conjuntos de datos a disposición de equipos, con el fin de encontrar modelos que ofrezcan mejores recomendaciones que las de esas mismas empresas.

Además, es muy difícil encontrar algún tipo de servicio en la actualidad de música, de vídeo o de compras que no nos inste a valorar los productos para poder ofrecernos recomendaciones.

A continuación se detallan proyectos y artículos relacionados con los sistemas de recomendación.

### 6.1. Artículos

#### Netflix Prize

En esta competición [27], *Netflix* ofreció un premio de 1,000,000 de dólares al equipo que pudiera obtener un modelo que recomendara un 10 % mejor que el modelo propio de la compañía.

El conjunto de datos que se utilizó durante la competición estaba compuesto por:

- Conjunto de entrenamiento:
  - 100.480.507 valoraciones
  - 480.189 usuarios
  - 17.770 películas
  - Cada usuario ha valorado una media de 200 películas
  - Cada película ha sido valorada por una media de 500 usuarios
- Conjunto de clasificación:
  - 2.817.131 interacciones usuario-película

La métrica que se utilizó para evaluar los modelos fue el *RMSE*. El equipo ganador logró un *RMSE* de 0,8567; es decir, el modelo obtenido por ese equipo fue un 10,06 % mejor que el de *Netflix* (tan solo un 0,06 % les hizo ganar).

En 2010 *Netflix* anunció que no iba a haber una segunda competición debido a temas judiciales con respecto a la privacidad de los clientes de la compañía, aunque los conjuntos de datos fueron creados para preservar la privacidad de los mismos.

Información de la competición: [Netflix Prize](#).

## Spotify

El reto de *ACM RecSys* de 2018 [22] estuvo organizado por *Spotify*. El reto se centró en la continuación automática de playlists. Para ello, *Spotify* dejó una conjunto de datos con un gran número de playlists que, a su vez, contenían canciones asociadas. Los equipos tenían que predecir las canciones que faltaban en las playlists del conjunto de evaluación.

El conjunto de datos que ofreció *Spotify* contenía 1.000.000 de playlists creadas por sus usuarios.

En cuanto a las métricas elegidas para la evaluación, se utilizaron las siguientes:

- Precisión R: número de canciones relevantes obtenidas dividido entre el número de canciones relevantes conocidas.
- NDCG (*Normalized discounted cumulative gain*): cuantifica la calidad del ranking de canciones recomendadas.

- Clicks: característica de *Spotify* por la cual dada una playlist se recomiendan 10 canciones para añadir. Esta lista se puede refrescar para mostrar otras 10 canciones. Mide el número de clicks para refrescar hasta encontrar una canción relevante.

Enlace al reto: [ACM RecSys Challenge 2018](#).

## Trivago

El reto de *ACM RecSys* de 2019 [23] estuvo organizado por *Trivago*. El reto se centró en desarrollar un sistema de recomendación que provea al usuario de alojamiento según sus necesidades y teniendo en cuenta los alojamientos que ha visitado en una sesión. El premio para el equipo ganador es de 8.000 euros.

La métrica escogida fue el *MRR*, o *Mean Reciprocal Rank*.

Enlace al reto: [ACM RecSys Challenge 2019](#).

## 6.2. Proyectos

### Tourist chatbot

En este proyecto la alumna desarrolló un chatbot para *Telegram* dirigido a los turistas [26]. El chatbot es capaz de recomendar puntos de interés a los turistas basándose en las preferencias del usuario. Para ello, se utilizan los dos tipos principales de modelos de recomendación: el basado en contenido y el colaborativo.

Enlace al repositorio: [Tourist chatbot](#).

## 6.3. Fortalezas y debilidades del proyecto

### Fortalezas del proyecto

Las fortalezas del proyecto son las siguientes:

- Se pueden obtener tanto modelos de recomendación clásicos como basados en aprendizaje profundo con una sola aplicación.
- Si no se tiene GPU no pasa nada, el modelo se obtendrá a partir de la CPU.
- Interfaz de usuario sencilla.

## **Debilidades del proyecto**

Las debilidades del proyecto son las siguientes:

- No se pueden obtener las predicciones para el modelo de secuencia.
- No se pueden añadir valoraciones y/o usuarios al modelo de secuencia.
- Tiempo de lectura de datos y obtención de modelos demasiado alto según el conjunto de datos.

---

# Conclusiones y Líneas de trabajo futuras

---

## 7.1. Conclusiones

Todavía hay que sacarlas.

## 7.2. Líneas de trabajo futuras

Dado que no se han podido obtener las predicciones para el modelo de secuencia, se puede dejar como futura mejora el obtenerlas.

Además, también sería una futura mejora el añadir valoraciones y/o usuarios a los conjuntos de datos.





---

## Bibliografía

---

- [1] Jabref. <http://www.jabref.org/>.
- [2] Lukas Brozovsky and Vaclav Petricek. Recommender system for online dating service. In *Proceedings of Conference Znalosti 2007*, Ostrava, 2007. VSB.
- [3] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, New York, NY, USA, 2011. ACM.
- [4] CopperUnion. Anime recommendations database. Technical report, Kaggle, 2017.
- [5] fast ai. *Practical Deep Learning for Coders, v3*.
- [6] Yúbal FM. ¿escucha facebook lo que dices a través del móvil? ellos lo niegan, pero vuelve la polémica. *Xataka.com*, 2018.
- [7] GroupLens. Movielens. In *MovieLens*, 1998.
- [8] Uriel Hernandez. Mvc (model, view, controller) explicado.
- [9] Enrique Herrera-Viedma & Carlos Porcel & Lorenzo Hidalgo. Sistemas de recomendaciones: herramientas para el filtrado de información en internet. *Hipertext.net*, 2004.
- [10] Maciej Kula. Lightfm. <https://github.com/lyst/lightfm>.
- [11] Maciej Kula. *Model evaluation - auc score*.

- [12] Maciej Kula. *Model evaluation - precision at k*.
- [13] Maciej Kula. *Model evaluation - recall at k*.
- [14] Maciej Kula. *Model evaluation - reciprocal rank*.
- [15] Maciej Kula. Metadata embeddings for user and item cold-start recommendations. In Toine Bogers and Marijn Koolen, editors, *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.*, volume 1448 of *CEUR Workshop Proceedings*, pages 14–21. CEUR-WS.org, 2015.
- [16] Maciej Kula. Spotlight. <https://github.com/maciejkula/spotlight>, 2017.
- [17] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of Massive Datasets*, chapter 9, page 513. Universidad de Stanford, 2010.
- [18] Pallets Team. *Welcome to Flask*.
- [19] Python Software Foundation. *Chardet: The Universal Character Encoding Detector*.
- [20] Python Software Foundation. *Graphical User Interfaces with Tk*.
- [21] Python Software Foundation. *pickle - Python object serialization*.
- [22] ACM RecSys. Acm recsys challenge 2018. *recsyschallenge.com*, 2018.
- [23] ACM RecSys. Acm recsys challenge 2019. *recsyschallenge.com*, 2019.
- [24] SciPy.org. *scipy.sparse.coo<sub>m</sub>atrix*.
- [25] SciPy.org. *scipy.sparse.csr<sub>m</sub>atrix*.
- [26] Jasmin Wellnitz. Tfg - development of a chatbot for tourist recommendations. <https://github.com/jaswellnitz/tourist-chatbot>, 2017.
- [27] Wikipedia. Netflix Prize — Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Netflix%20Prize&oldid=842801480>, 2019. [Online; accessed 30-June-2019].

- [28] Wikipedia. Scrum (desarrollo de software) — Wikipedia, the free encyclopedia. [http://es.wikipedia.org/w/index.php?title=Scrum%20\(desarrollo%20de%20software\)&oldid=115368106](http://es.wikipedia.org/w/index.php?title=Scrum%20(desarrollo%20de%20software)&oldid=115368106), 2019. [Online; accessed 25-February-2019].
- [29] Wikipedia. Sparse matrix — Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Sparse%20matrix&oldid=893179870>, 2019. [Online; accessed 12-April-2019].
- [30] WTForms Team. *WTForms Documentation*.
- [31] Cai-Nicolas Ziegler. Book-crossing dataset. Technical report, IIF, 2004.