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 — 3 de julio de 2019
Tutor: Bruno Baruque Zanón

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Bruno Baruque Zanón, profesor del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Raúl Negro Carpintero, con DNI 71305764Z, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Sistema de Recomendación basado en Aprendizaje Profundo.

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, 3 de julio de 2019

Vº. Bº. del Tutor:

D. Bruno Baruque Zanón

Resumen

El tema principal del proyecto son los sistemas de recomendación. Estas herramientas generan predicciones sobre ítems que pueden ser relevantes para los usuarios. Tal es su importancia que, prácticamente, todas las grandes compañías que ofrecen algún tipo de servicio o producto utilizan estos sistemas para dar recomendaciones a los usuarios. El objetivo es comparar el rendimiento entre los modelos clásicos y los modelos basados en aprendizaje profundo sobre diferentes conjuntos de datos.

Además, se ofrecerá al usuario la posibilidad de interactuar con los modelos desde una interfaz web, a través de la cual podrán generar modelos, guardarlos, ver su desempeño siguiendo determinadas métricas y ver las predicciones que calculan.

Descriptores

Sistemas de recomendación, aprendizaje profundo, redes neuronales, modelos de recomendación, interfaz web.

Abstract

The main topic of the project are the recommendation systems. These tools generate predcitions on items that can be relevant to the users. Their impact is so big that practically every big company that offers any kind of service uses them. The goal is to compare the performance between classic models and those based on deep learning on different datasets.

Furthermore, the user will be provided with the posibility of interacting with the models through a web interface, so they will be able to generate models, save them, evaluate their performance in different metrics and get the predictions they create.

Keywords

Recommendation systems, deep learning, neural networks, recommendation models.

Índice general

Índice de figuras

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 [33], [29], [30]. El premio que ofreció *Netflix* por mejorar la capacidad de predicción del modelo que se empleaba en aquel momento un 10 % fue de 1.000.000 de dólares.

En este proyecto serán objeto de estudio los modelos de recomendación clásicos 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. Debido a la gran cantidad de datos que se generan actualmente, es interesante aplicar técnicas de aprendizaje profundo en los sistemas de recomendación con el fin de intentar obtener recomendaciones mejores que con los sistemas clásicos y, en la medida de lo posible, en menor tiempo.

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. [17]

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.

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

- Modelos basados en contenido
- Modelos colaborativos
- Modelos híbridos

Los tres tipos de modelos se han podido obtener con *LightFM*, no así con *Spotlight*, con el cual no se puede obtener un modelo híbrido.

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.

En *Spotlight*, este tipo de sistema se correspondería con el modelo de factorización implícito y con el modelo de secuencia implícito. En estos modelos se utiliza la matriz de interacción entre usuarios e ítems pero sin las valoraciones.

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.

En *Spotlight*, este tipo de sistema se correspondería con el modelo de factorización explícito.

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 [20] 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:

$$Precision\ at\ k = \frac{ERC \subset kRP}{k} \tag{3.1}$$

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

Recall k

El recall k [21] 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:

$$Recall\ at\ k = \frac{ERk}{ERT} \tag{3.2}$$

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 [19] 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 [22] es el inverso del score del elemento más relevante devuelto en el ranking de resultados. Su fórmula es:

$$Reciprocal\ rank = \frac{1}{SER} \tag{3.3}$$

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 este caso, el *RMSE* [23] compara un valor conocido con un valor observado. Su fórmula sería la siguiente:

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

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* [35]. 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*

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

Matrices COO

Las *matrices COO* [7], o matrices coordenadas *(COOrdinate)*, 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*, o *Compressed Sparse Row* [8]. Estas matrices 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.

3.4. Paralelismo

Debido a la gran cantidad de datos que se pueden llegar a manejar en los sistemas de recomendación y, en general, en cualquier aplicación de *Big Data*, *Deep Learning*, etc., es muy interesante el uso de computación paralela.

Esto se consigue gracias a *CUDA*, *Compute Unified Device Architecture* [27]. Es una plataforma de computación en paralelo desarrollada por *nVidia* para ser usada en sus tarjetas gráficas.

Ya que las *GPUs* tienen muchísimos más núcleos que las *CPUs*, las primeras se pueden utilizar para ejecutar operaciones en paralelo. Esto hace que se ahorre mucho tiempo en campos como los sistemas de recomendación.

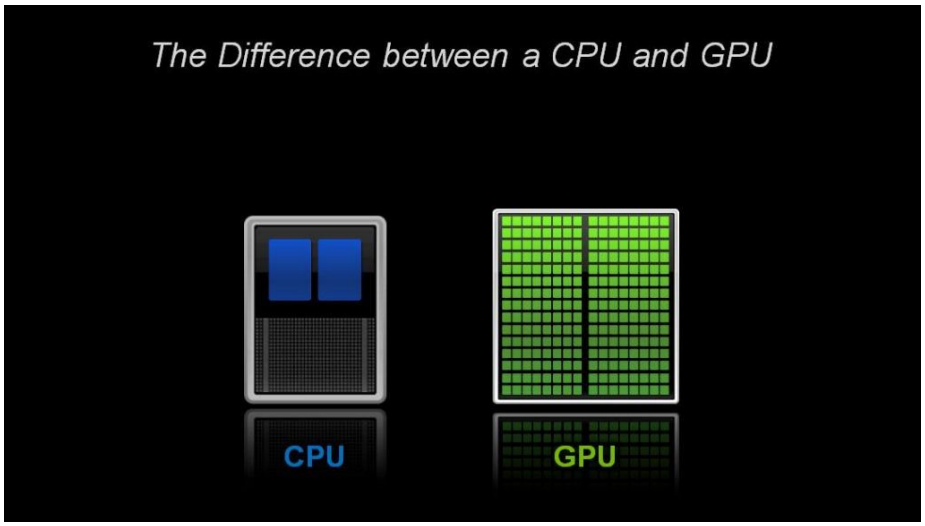

Figura 3.1: Diferencia de núcleos entre CPU y GPU

En el proyecto se puede utilizar *CUDA* con el sistema de *Spotlight*.

Técnicas y herramientas

4.1. Metodologías

Scrum

Scrum es una técnica de desarrollo ágil que se caracteriza por: [34]

- 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 en el que solo ha habido un programador no ha hecho falta llevar a cabo las reuniones diarias. Tan solo se han realizado las reuniones de fin de sprint, que han tenido lugar cada dos semanas aproximadamente.

4.2. Patrones de diseño

MCV

Para el desarrollo del código se optó por seguir el patrón *Modelo Vista Controlador*.

Según este patrón, el código se divide en tres partes principales:

- **Modelo:** contiene los archivos dedicados a manipular los datos.
- **Vista:** contiene los archivos con los que el usuario interactúa para usar la aplicación.
- **Controlador:** recibe las peticiones hechas desde la vista y pide los datos al modelo.

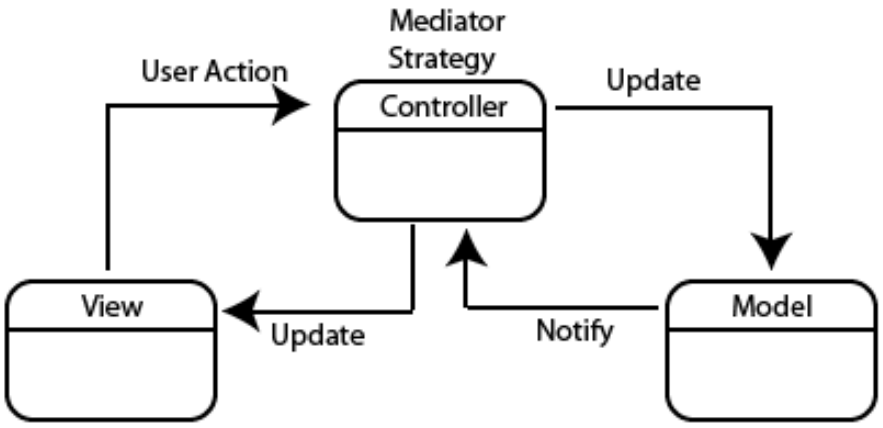

Figura 4.2: Esquema del patrón MVC [28]

4.3. 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.

4.4. 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.

4.5. 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.

4.6. 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 LaTeX [32] 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*.

4.7. 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* [2].

LightFM

LightFM es una implementación para Python de populares algoritmos clásicos de recomendación [18].

Se decidió emplear *LightFM* debido a su aparente sencillez y por ser una librería bastante completa. También se escogió debido a que no es un proyecto que haya pasado desapercibido, hay una buena cantidad de artículos en los que se menciona.

También se estudió la posibilidad de utilizar la librería *Crab*, pero no se pudo llegar a instalar. Además, el repositorio llevaba mucho años sin actualizarse.

Spotlight

Spotlight es una librería creada por la misma gente que *LightFM* que utiliza *PyTorch* para construir modelos de recomendación [25] basados en aprendizaje profundo.

Al principio se estudió utilizar la librería de *fast.ai* para obtener los modelos de aprendizaje profundo. La idea se descartó al descubrir que la misma gente que había creado *LightFM* tenía una librería similiar para deep learning. Así pues, se decidió utilizar *Spotlight* al suponer que el cálculo de las métricas iba a resultar muy parecido entre ambas herramientas.

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 [3].

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 [1].

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 [5].

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

4.8. Datasets

Anime

Este conjunto de datos [13] 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 [36] 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 [16] 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* [24].

Last.FM

Este conjunto de datos [12] 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 [11] 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.

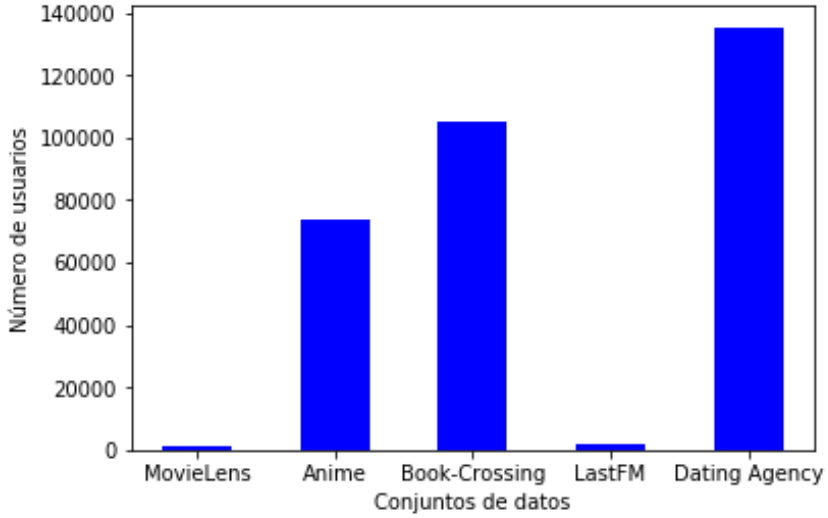

Figura 4.3: Número de usuarios por cada dataset de prueba

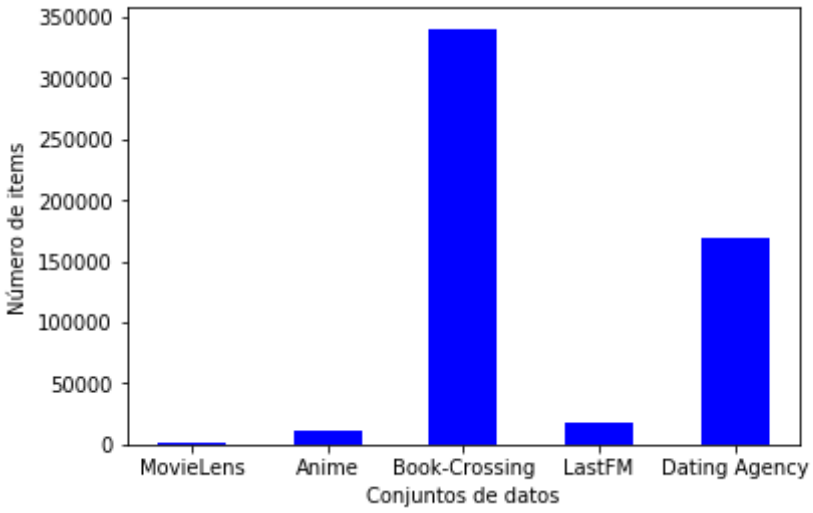

Figura 4.4: Número de ítems por cada dataset de prueba

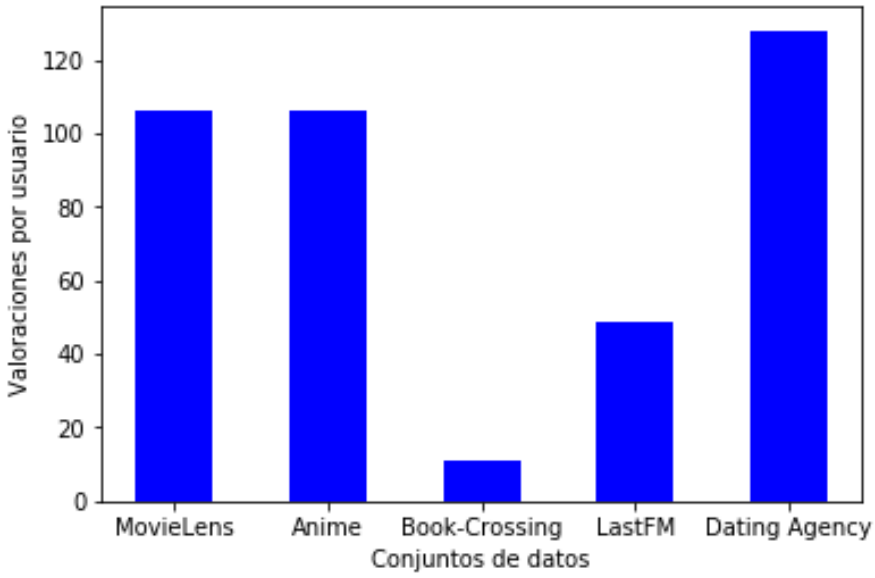

Figura 4.5: Número de valoraciones por usuario de cada dataset de prueba

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) [26].

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

- *Practical Deep Learning For Coders, v3* (Jeremy Howard) [6].

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

- *Welcome to Flask* (Pallets Team) [9].
- *WTForms Documentation* (WTForms Team) [10].

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

- *LightFM* (Maciej Kula) [18].
- *Spotlight* (Maciej Kula) [25].

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 se dieron 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* [6], 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*, lo cual hace que la manera de trabajar con estas dos herramientas sea muy parecida. 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*. Es una librería que aparece en una gran cantidad de artículos elogiando su facilidad de uso y su potencial. Cumplió todas las funciones que se esperaba 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*.

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 ha conseguido obtener las predicciones para el modelo de secuencia de *Spotlight*. Esto es debido a que a diferencia del resto de modelos, con los cuales para ver las predicciones basta con indicar el usuario cuya predicciones se quiere obtener, en el modelo de secuencia no vale el usuario. Hay que utilizar una secuencia de interacciones del usuario. Esto dificulta el trabajo a la hora de programarlo, además de complicarle la vida al usuario que va a interactuar con la aplicación.

En lugar de que la persona que vaya a probar la aplicación tenga que elegir el id del usuario cuyas predicciones quiere ver, tendría que saber de ante mano la secuencia de interacciones que ese usuario ha tenido con los ítems para que el modelo pueda calcular los siguientes con los que podría interactuar.

Añadir valoraciones

Tampoco ha sido posible llevar a cabo la adición de más valoraciones por parte de usuarios ya existentes o nuevos. No queda muy claro cómo proceder con los métodos de *fit_partial* de los modelos. Además, según han comentado algunos usuarios de las librerías, parece que hay que tener en cuenta con antelación si se va a querer añadir nuevos usuarios, porque deberían existir en el conjunto de datos original (aunque no interactúen con nada) [14].

Heroku

Se ha intentado desplegar la aplicación *Flask* a través de *Heroku* y *Gunicorn*, pero no ha sido posible debido a que la aplicación de *Heroku* no era capaz de trabajar con los paquetes del proyecto. Siempre daba fallo de *ModuleNameNotFound*.

Se ha intentado cambiar la forma de importar los paquetes sin suerte.

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 LaTeX.

Además, la documentación se apoya en el programa *JabRef* [4], 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 [15]), 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 [33]. 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 [33], *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 [29] 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 divido 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 [30] 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 [31]. 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.
- Tiempo de lectura de datos y obtención de modelos demasiado alto según el conjunto de datos (hasta 12 horas se tardó en obtener, entrenar y obtener las métricas de un modelo de *LightFM* con el conjunto de datos de *Dating Agency*).
- No se pueden utilizar todas las métricas disponibles en los modelos implícitos de *Spotlight*.

Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

Como se ha podido observar a lo largo de todos los artículos que se han leído, y por la experiencia personal con determinados servicios, está claro que los sistemas de recomendación son unas herramientas muy potentes y que tienen una importancia mayúscula en nuestras vidas. Hemos normalizado completamente que compañías como *Neflix*, *Amazon*, *Spotify*, etc. nos ofrezcan productos o servicios que piensan que nos pueden gustar y que, de hecho, en muchos casos nos acaban gustando.

A continuación se muestran parte de los resultados obtenidos por los distintos modelos con el conjunto de datos de *Movielens*, ya que al haber realizado cambios de última hora en los sistemas, es inviable entrenar a tiempo todos los modelos varias veces con todos los datasets de prueba:

Modelo	**Precisión k**	**AUC Score**	**Recall k**	**Ranking recíproco**
Colaborativo	0,1879	0,8675	0,1218	0,4304
Híbrido	0,1759	0,8655	0,1160	0,4017
Por contenido	0,1746	0,8640	0,1143	0,3910

Tabla 7.1: Métricas modelos *LightFM*

Como se puede observar, el modelo colaborativo es el que mejor sale evaluado. Esto contrasta con la teoría, ya que el modelo híbrido debería ser

mejor por utilizar tanto las valoraciones como las features de los usuarios y de los ítems.

Se tendría que repetir las pruebas con otros conjuntos de datos distintos para ver si la tendencia es esa o cambia.

Modelo	**RMSE**	**MRR**	**Precisión k**	**Recall k**
Fact. expl. time	1,1457	0,0102	0,0424	0,0168
Fact. expl.	1,1471	0,0114	0,0409	0,0177
Fact. impl. time	-	0,0625	0,2317	0,1380
Fact. impl.	-	0,0671	0,2367	0,1482
Secuencia impl. -	0,0304	-	-	

Tabla 7.2: Métricas modelos *Spotlight*

Como se puede observar, los mejores modelos son los que no hacen uso de los timestamps. Entre ellos, el que mejor sale parado es el modelo de factorización implícito sin timestamps (tampoco hace uso de las valoraciones).

Si comparamos ambos modelos:

Modelo	**Precisión k**	**Recall k**	**MRR**
Colaborativo	0,1879	0,1218	0,4304
Fact. impl.	0,2367	0,1482	0,0671

Tabla 7.3: *LightFM* vs *Spotlight*

Como se puede observar, el modelo de factorización implícito obtiene mejores valores de precisión y recall que el modelo colaborativo, pero sale perdiendo, y por mucho, en el ranking recíproco.

Con estos resultados, se lleva a la conclusión de que el mejor modelo de *LightFM* es el que solo utiliza las valoraciones para recomendar, mientras que el mejor modelo de *Spotlight* es uno de los que no utilizan las valoraciones para nada.

A parte de estos resultados evidentes, se ha podido comparar el tiempo que tarda uno de los sistemas de *LightFM* en leer el conjunto de *Dating Agency*, obtener, entrenar y evaluar el modelo frente a lo que tarda un modelo de *Spotlight* en hacer lo mismo. Mientras que el modelo clásico tardaba unas

12 horas en completar las tareas, el modelo basado en aprendizaje profundo lo ha hecho en unas 4 horas. Esto es lo que podemos esperar de los modelos de deep learning gracias al uso de *CUDA*.

7.2. Líneas de trabajo futuras

Dado que no se han podido obtener las predicciones para el modelo de secuencia, se podría plantear como futura línea de trabajo obtenerlas.

También se deja como futura mejora el añadir valoraciones de usuarios existentes o de usuarios nuevos.

Desgraciadamente, ni *LightFM* ni *Spotlight* implementan métodos de búsqueda de hiperparámetros. Tampoco lo hace *PyTorch*, por lo que podría ser interesante el crear un método capaz de obtener los mejores parámetros para los modelos, ya sea una búsqueda aleatoria o en malla.

Otra posible tarea es la de desplegar la aplicación en un servidor con hardware çompetitivoz comprar los resultados para determinar hasta qué punto el hardware del equipo ralentiza la obtención y el entrenamiento de los modelos. La mejora va a ser muy notable sí o sí. Tan solo hay que ver que los modelos clásicos tardan hasta 12 horas en completar todo el proceso ejecutándose en la CPU (Intel Core i5-6000) mientras que los basados en aprendizaje profundo tardan unas x horas ejecutándose en una GPU de gama media-alta pero con ya unos años a sus espaldas(nVidia GTX 970). Es por eso que sería interesante ver el desempeño de los modelos basados en deep learning en hardware más moderno y potente (tal ver una nVidia GTX Titan o Titan V).

Dado que se han realizado cambios de última hora en la obtención de los sistemas, estaría bien en un futuro el poder tener una comparativa más exhaustiva y extensa de las métricas de cada modelo.

También se deja como futura mejora la creación de tests y pruebas sobre el código, que no ha sido posible realizar debido a la falta de tiempo.

Otra mejora sería la de conseguir desplegar el proyecto para que pueda ser accesible desde fuera, no como ahora que está en local. Se ha intentado con *Heroky* y *Gunicorn*, pero no ha habido suerte.

Bibliografía

[1] Chardet: The universal character encoding detector. https://pypi.org/project/chardet/. [Internet; Accedido 01-junio-2019].

[2] Dataframe. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html. [Internet; Accedido 18-diciembre-2018].

[3] Graphical user interfaces with tk. https://docs.python.org/3/library/tk.html. [Internet; Accedido 03-junio-2019].

[4] Jabref. http://www.jabref.org/. [Internet; Accedido 09-febrero-2019].

[5] pickle - python object serialization. https://docs.python.org/3/library/pickle.html. [Internet; Accedido 25-marzo-2019].

[6] Practical deep learning for coders, v3. https://course.fast.ai/. [Internet; Accedido 03-diciembre-2018].

[7] scipy.sparse.coomatrix. https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.coo_matrix.html#scipy.sparse.coo_matrix. [Internet; Accedido 04-abril-2019].

[8] scipy.sparse.csrmatrix. https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html#scipy.sparse.csr_matrix. [Internet; Accedido 04-abril-2019].

[9] Welcome to flask. http://flask.pocoo.org/docs/1.0/. [Internet; Accedido 04-mayo-2019].

[10] Wtforms documentation. https://wtforms.readthedocs.io/en/stable/. [Internet; Accedido 14-junio-2019].

[11] Lukas Brozovsky and Vaclav Petricek. Recommender system for online dating service. In *Proceedings of Conference Znalosti 2007*, Ostrava, 2007. VSB.

[12] 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.

[13] CopperUnion. Anime recommendations database. https://www.kaggle.com/CooperUnion/anime-recommendations-database/home, 2017. [Internet; Accedido 06-febrero-2019].

[14] Jeremy Fischer. https://github.com/lyst/lightfm/issues/347#issuecomment-410769208. [Internet; Accedido 01-julio-2019].

[15] Yúbal FM. ¿escucha facebook lo que dices a través del móvil? ellos lo niegan, pero vuelve la polémica. https://www.xataka.com/seguridad/facebook-vuelve-a-negar-que-te-escuche-a-traves-del-movil-una-acusacion- 2018. [Internet; Accedido 30-junio-2019].

[16] GroupLens. Movielens. In *MovieLens*, 1998.

[17] Enrique Herrera-Viedma & Carlos Porcel & Lorenzo Hidalgo. Sistemas de recomendaciones: herramientas para el filtrado de información en internet. https://www.upf.edu/hipertextnet/numero-2/recomendacion.html, 2004. [Internet; Accedido 01-noviembre-2018].

[18] Maciej Kula. Lightfm. https://github.com/lyst/lightfm. [Internet; Accedido 11-diciembre-2018].

[19] Maciej Kula. Model evaluation - auc score. http://lyst.github.io/lightfm/docs/lightfm.evaluation.html#lightfm.evaluation.auc_score. [Internet; Accedido 07-febrero-2019].

[20] Maciej Kula. Model evaluation - precision at k. http://lyst.github.io/lightfm/docs/lightfm.evaluation.html#lightfm.evaluation.precision_at_k. [Internet; Accedido 07-febrero-2019].

[21] Maciej Kula. Model evaluation - recall at k. http://lyst.github.io/lightfm/docs/lightfm.evaluation.html#lightfm.evaluation.recall_at_k. [Internet; Accedido 07-febrero-2019].

[22] Maciej Kula. Model evaluation - reciprocal rank. http://lyst.github.io/lightfm/docs/lightfm.evaluation.html#lightfm.evaluation.reciprocal_rank. [Internet; Accedido 07-febrero-2019].

[23] Maciej Kula. Model evaluation - rmse score. https://maciejkula.github.io/spotlight/evaluation.html#spotlight.evaluation.rmse_score. [Internet; Accedido 20-mayo-2019].

[24] 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.

[25] Maciej Kula. Spotlight. https://github.com/maciejkula/spotlight, 2017. [Internet; Accedido 07-mayo-2019].

[26] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of Massive Datasets*, chapter 9, page 513. Universidad de Standford, 2010.

[27] nVidia. Cuda zone. https://developer.nvidia.com/cuda-zone. [Internet; Accedido 02-julio-2019].

[28] Antonio Blanco Oliva. Modelo vista controlador en wordpress. https://www.ablancodev.com/wordpress/modelo-vista-controlador/. [Internet; Accedido 02-julio-2019].

[29] ACM RecSys. Acm recsys challenge 2018. http://www.recsyschallenge.com/2018/, 2018. [Internet; Accedido 30-junio-2019].

[30] ACM RecSys. Acm recsys challenge 2019. http://www.recsyschallenge.com/2019/, 2019. [Internet; Accedido 30-junio-2019].

[31] Jasmin Wellnitz. Tfg - development of a chatbot for tourist recommendations. https://github.com/jaswellnitz/tourist-chatbot, 2017.

[32] Wikipedia. Latex — wikipedia, la enciclopedia libre, 2015. [Internet; descargado 30-septiembre-2015].

[33] 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].

[34] Wikipedia. Scrum (desarrollo de software) — Wikipedia, the free encyclopedia. http://es.wikipedia.org/w/index.php?title=Scrum%20(desarrollo%20de%20software)&oldid=115368106, 2019. [Online; accessed 25-February-2019].

[35] 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].

[36] Cai-Nicolas Ziegler. Book-crossing dataset. http://www2.informatik.uni-freiburg.de/~cziegler/BX/, 2004. [Internet; Accedido 06-febrero-2019].