UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Sistema de Recomendación basado en Aprendizaje Profundo Documentación Técnica

Presentado por Raúl Negro Carpintero
en Universidad de Burgos — 4 de junio de 2019
Tutor: Bruno Baruque Zanón

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

A.2. Planificación temporal

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

Este anexo recoge los objetivos generales y la especificación de requisitos del proyecto.

B.2. Objetivos generales

El proyecto persigue los siguientes objetivos generales:

- Comprender los sistemas de recomendación tanto clásicos como basados en aprendizaje profundo.
- Recoger y evaluar los resultados obtenidos por los dos modelos sobre diferentes conjuntos de datos.
- Comparar los resultados.

B.3. Catalogo de requisitos

Los requisitos derivados de los objetivos del proyecto son los siguientes:

Requisitos funcionales

- **RF-1 Gestión de los datos intermedios:** el programa tiene que ser capaz de gestionar los datos intermedios:

    - **RF-1.1 Guardar matrices:** el programa tiene que ser capaz de guardar las matrices generadas por los modelos para ahorrar tiempo en siguientes ejecuciones.
    - **RF-1.2 Cargar matrices:** el programa tiene que ser capaz de cargar el conjunto de datos que el usuario quiera.
- **RF-2 Gestión de los modelos:** el programa tiene que ser capaz de gestionar los modelos:
    - **RF-2.1 Mostrar modelos:** el programa tiene que ser capaz de mostrar todos los modelos disponibles.
    - **RF-2.2 Seleccionar modelos:** el programa tiene que ser capaz de dejar al usuario seleccionar el modelo que quiera.
    - **RF-2.3 Guardar modelos:** el programa tiene que ser capaz de guardar los modelos obtenidos para ahorrar tiempo en futuras ejecuciones.
    - **RF-2.4 Cargar modelos:** el programa tiene que ser capaz de cargar los modelos previamente guardados.
- **RF-3 Gestión de los resultados:** el programa tiene que ser capaz de gestionar los resultados:
    - **RF-3.1 Guardar resultados:** el programa tiene que ser capaz de guardar los resultados generados por los modelos.
    - **RF-3.2 Comparar resultados:** el programa tiene que ser capaz de comparar los resultados obtenidos por los distintos modelos.
    - **RF-3.3 Mostrar predicciones:** el programa tiene que ser capaz de mostrar las predicciones obtenidas por los distintos modelos.
- **RF-4 Gestión de los usuarios:** el programa tiene que ser capaz de introducir las valoraciones de nuevos usuarios:
    - **RF-4.1 Añadir valoraciones:** el programa tiene que ser capaz de simular la entrada de nuevos usuarios permitiendo el añadido de nuevas valoraciones.
- **RF-5 Gestión de los conjuntos de datos:** el programa tiene que:
    - **RF-5.1 Mostrar los conjuntos de datos:** el programa tiene que ser capaz de mostrar los distintos conjuntos de datos de prueba.
    - **RF-5.2 Seleccionar el conjunto de datos:** el programa tiene que ser capaz de mostrar dejar que el usuario seleccione uno de los conjuntos de datos de prueba.
    - **RF-5.3 Añadir conjunto de datos:** el programa tiene que ser capaz de dejar que el usuario añada un conjunto de datos propio.

- **RF-6 Ayuda de la aplicación:** el programa tiene que ofrecer información al usuario:
  - **RF-6.1 Mostrar información sobre modelos:** el programa tiene que ser capaz de mostrar información sobre los distintos modelos que se pueden escoger.
  - **RF-6.2 Mostrar información sobre conjuntos de datos:** el programa tiene que ser capaz de mostrar información sobre los distintos conjuntos de datos de prueba.

Requisitos no funcionales

- **RNF-1 Usabilidad:** la interfaz gráfica tiene que ser intuitiva y fácil de usar.
- **RNF-2 Soporte:** el programa tiene que dar soporte a versiones iguales o mayores a Python 3.
- **RNF-3 Localización:** el programa tiene que estar preparado para soportar varios idiomas.

B.4. Especificación de requisitos

En esta sección se mostrará el diagrama de casos de uso y se desarrollará cada uno de ellos.

Diagrama de casos de uso

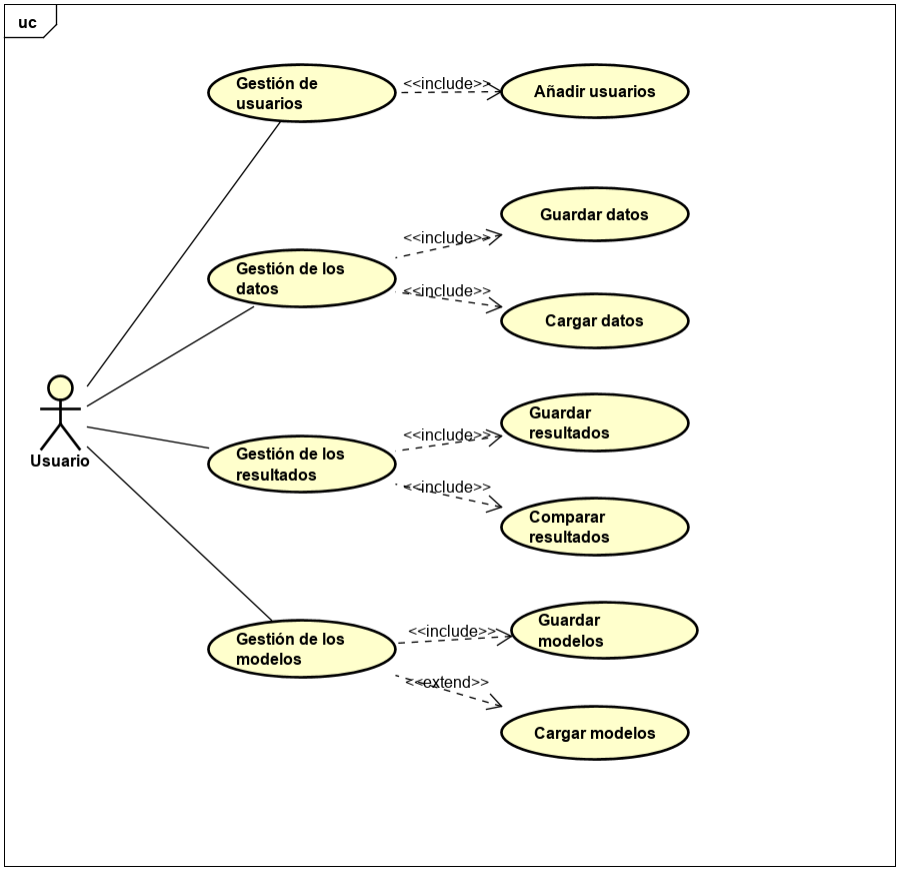

Figura B.1: Diagrama de casos de uso

Actores

Con la aplicación solo interactuará un actor, el usuario que esté probando la aplicación en un momento determinado.

Casos de uso

A continuación, se desarrollará cada caso de uso:

Apéndice C

Especificación de diseño

C.1. Introducción

En este apéndice se explica cómo están conformados los datos que utilizan las librerías usadas en el proyecto, así como la forma en la que está estructurado el mismo.

C.2. Diseño de datos

Todos los datos que he utilizado a lo largo del proyecto están en formato .csv. Lo más normal es que para cada conjunto de datos tenga los siguientes archivos:

- *ratings.csv*
- *users.csv*
- *items.csv*

La estructura de estos archivos suele ser: *idUser, idItem, rating, timestamp* para *ratings.csv*, *idUser, name, feature1, ..., featureN* para *users.csv* y i*dItem, name, feature1, ..., featureN* para *items.csv*.

Para poder trabajar con los datos primero los paso a *DataFrames* de *pandas* [3].

Datos con LightFM

Una vez obtenidos los *DataFrames* para cada *.csv*, necesito convertirlos a *Dataset* de *LightFM* [2] para poder trabajar con ellos.

Esta clase se encarga de convertir los datos almacenados en los *DataFrames* en *matrices COO* y *matrices CSR*.

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Para la realización de este proyecto se ha seguido el patrón arquitectónico MVC (*Modelo Vista Controlador*). El objetivo de este patrón es dividir el código en función de su propósito. Sus partes son:

- *Modelo*: el acceso a los datos. Se corresponde con las clases de Entrada y Salida, que leen los datos para dárselo al sistema de recomendación y guardan los resultados.
- *Vista*: la visualización de los datos. Se corresponde con las clases de Interfaz, que muestran la información solicitada.
- *Controlador*: la manipulación de los datos. Se corresponde con los clases de Sistema, que crean los sistemas de recomendación gracias a los datos proporcionados por las clases de Entrada.

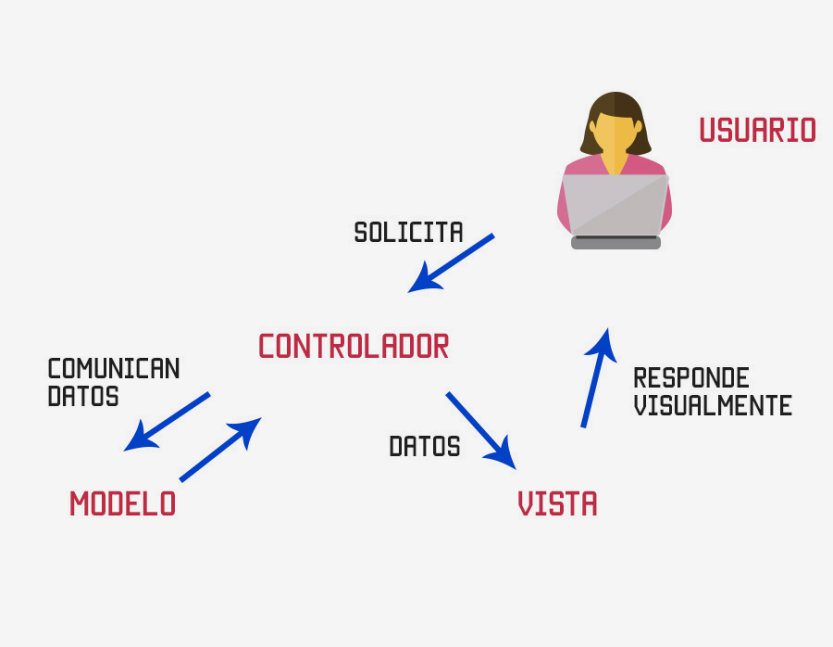

Figura C.1: Esquema del patrón MVC [1]

La estructura del proyecto siguiendo este patrón quedaría de la siguiente forma:

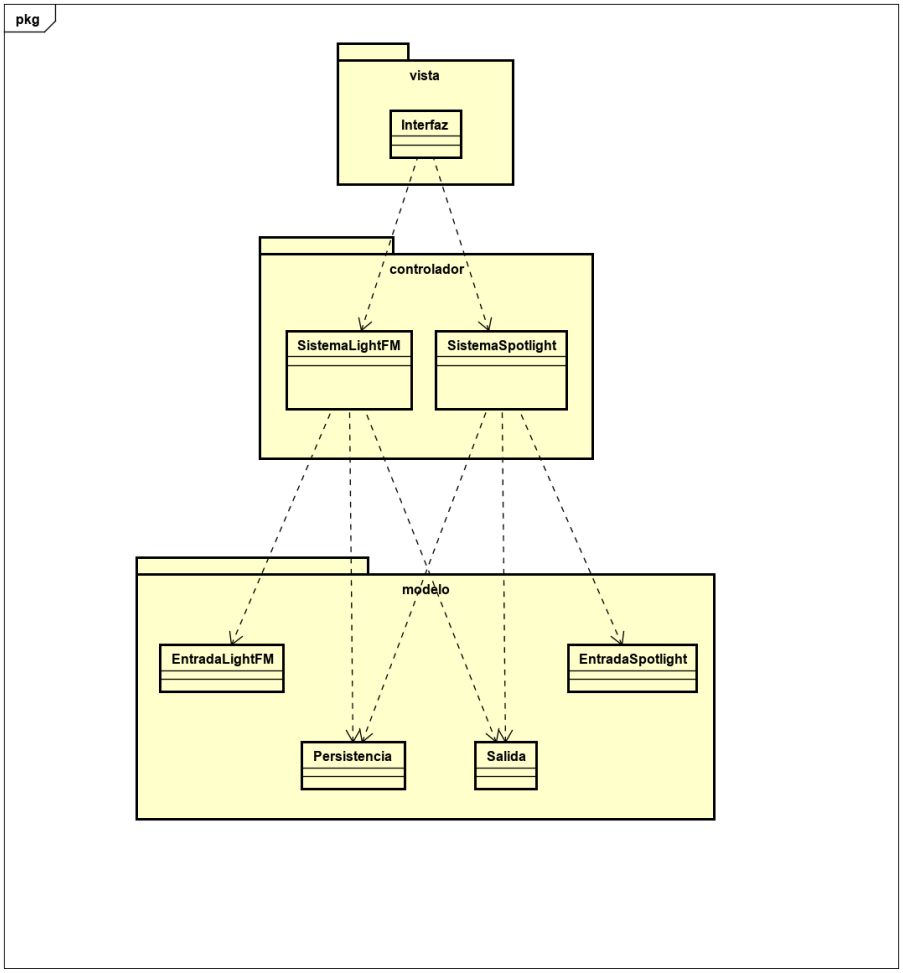

Figura C.2: Diagrama UML del proyecto

Por separado, los paquetes contienen:

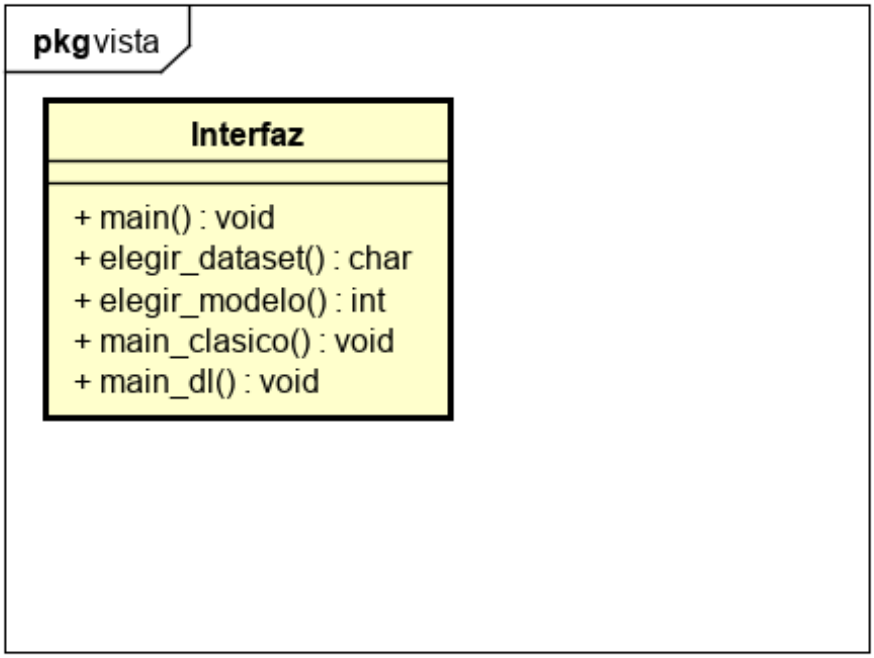

Figura C.3: Diagrama UML del paquete *vista*

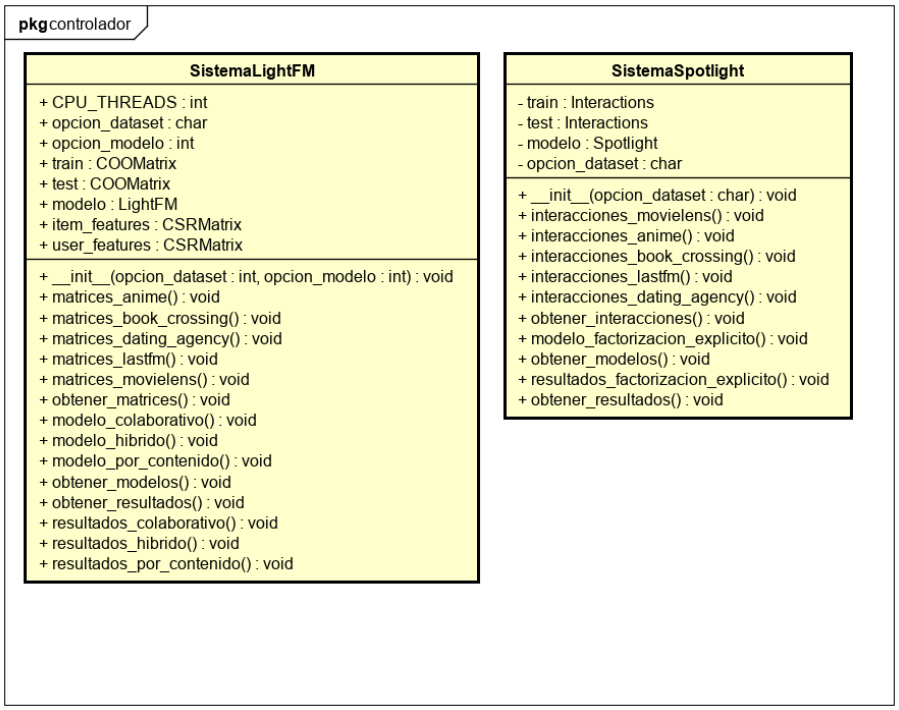

Figura C.4: Diagrama UML del paquete *controlador*

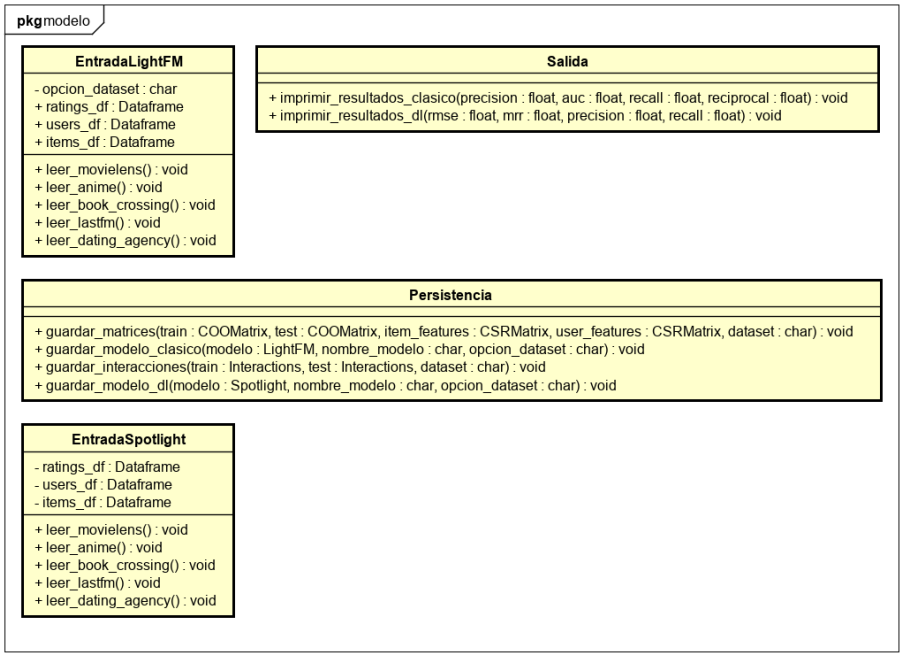

Figura C.5: Diagrama UML del paquete *modelo*

Diseño con LightFM

Para la parte del modelo clásico, se tienen los siguientes archivos: En la clase `EntradaLightFM` se tiene:

- `leer_x`: estos métodos recogen los datos de los *.csv* para cada conjunto de datos. Estos métodos son utilizados por los métodos de obtención de matrices de la clase `SistemaLightFM`.

En el archivo `Salida` se tiene:

- `imprimir_resultados_clasico`: este método imprime las métricas del modelo clásico escogido.

En el archivo `Persistencia` se tiene:

- `guardar_matrices`: este método guarda en un archivo *pickle* las matrices que *LightFM* necesita para obtener los modelos.

- `guardar_modelo`: este método guarda el modelo obtenido por *LightFM* en un archivo *pickle*.

En el archivo `Interfaz` se tiene:

- `elegir_dataset`: este método muestra un menú mediante el cual elegimos un conjunto de datos que utilizar.
- `elegir_modelo`: este método muestra un menú mediante el cual elegimos un modelo concreto a crear.
- `main_clasico`: programa principal si el modelo escogido es *LightFM*.

En la clase `SistemaLightFM` tenemos:

- `matrices_x`: estos métodos crean las matrices necesarias para cada conjunto de datos.
- `modelo_x`: estos métodos crean los distintos modelos de recomendación.
- `resultados_x`: estos métodos obtienen los resultados para cada modelo de recomendación.

Diseño con Spotlight

Para la parte del modelo basado en aprendizaje profundo, se tienen los siguientes archivos: En la clase `EntradaSpotlight` se tiene:

- `leer_x`: estos métodos recogen los datos de los *.csv* para cada conjunto de datos. Estos métodos son utilizados por los métodos de obtención de interacciones de la clase `SistemaSpotlight`.

En el archivo `Salida` se tiene:

- `imprimir_resultados_dl`: este método imprime las métricas del modelo basado en aprendizaje profundo escogido.

En el archivo `Persistencia` se tiene:

- `guardar_interacciones`: este método guarda en un archivo *pickle* las interacciones que *Spotlight* necesita para obtener los modelos.
- `guardar_modelo_dl`: este método guarda el modelo obtenido por *Spotlight*.

En el archivo `Interfaz` se tiene:

- `main_dl`: programa principal si el modelo escogido es *Spotlight*.

En la clase `SistemaSpotlight` tenemos:

- `interacciones_x`: estos métodos crean las interacciones necesarias para cada conjunto de datos.
- `modelo_x`: estos métodos crean los distintos modelos de recomendación.
- `resultados_x`: estos métodos obtienen los resultados para cada modelo de recomendación.

Apéndice D

Documentación técnica de programación

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía

[1] Uriel Hernandez. Mvc (model, view, controller) explicado.

[2] Maciej Kula. *Dataset construction*.

[3] Pandas. *DataFrame*.