Deusto
Facultad de Ingeniería
Ingeniaritza Fakultatea

Grado en Ingeniería Informática

Informatikako Ingeniaritzako gradua

Proyecto fin de grado

Gradu amaierako proiektua

Estudio sobre la privacidad en el aprendizaje federado mediante el desarrollo de un sistema de recomendación online

Ibai Guilén Pacho

Director: Diego Casado Mansilla

Bilbao, mayo de 2021

Capítulo 1

Contexto

En el siguiente capítulo se introducirán y explicarán los conceptos importantes sobre los que parte el proyecto. Es de gran importancia comprender y conocer lo que son los sistemas de recomendación y el Federated Learning para entender tanto el desarrollo, como la resolución de los problemas y retos que supone su implementación.

1.1. Sistemas de recomendación

Los sistemas de recomendación se basan en una gran cantidad de datos para generar predecciones. En el caso de que el sistema de recomendación fuera el de una tienda, este buscaría correlación entre los usuarios y los tipos de productos que compran, para así, recomendar ciertos artículos al grupo de personas que sea más probable que los compren.

A la hora de realizar la recomendación se aplican "filtros" de diferente índole, entre los más comunes se encuentran:

- Filtros demográficos, que recomiendan en función del sexo, edad, país, oficio, ...
- Filtros basados en contenidos, como Youtube, que recomiendan contenidos similares a los valorados por los usuarios.
- Filtrado colaborativo, que consiste en recomendar al usuario elementos valorados positivamente por usuarios similares a él.

Sin embargo, existen sistemas híbridos que utilizan varias de las estrategias de filtrado anteriores combinadas. Un ejemplo de ello es Amazon, que tiene uno de los algoritmos de recomendación más potentes.

Esto se debe a dos cosas, en primer lugar que cuenta con una gran cantidad de información de los usuarios, tanto su edad, género, país, dirección, rutinas (si tienen Alexa), ... como los artículos que miran, compran, añaden a la lista, etc. El algoritmo es capaz de ofrecer recomendaciones muy precisas a los usuarios y dar una gran calidad de servicio gracias a esto.

De hecho, en Amazon, si queremos revisar las recomendaciones tenemos un apartado propio para ellas al que se puede acceder fácilmente (Fig1.1).

Fig. 1.1: Recomendaciones de Amazon (Fuente: Amazon[**AmazonEsCompra**])

1.2. Federated Learning

El aprendizaje federado, o Federated Learning en inglés, es una rama del Machine Learning que busca entrenar al algoritmo a través de diversos dispositivos descentralizados, manteniendo siempre la información en cada uno de ellos. Estos dispositivos descentralizados forman una red de aprendizaje colaborativo que busca mejorar la precisión del algoritmo mediante la combinación de los modelos de cada participante de la red en un servidor central

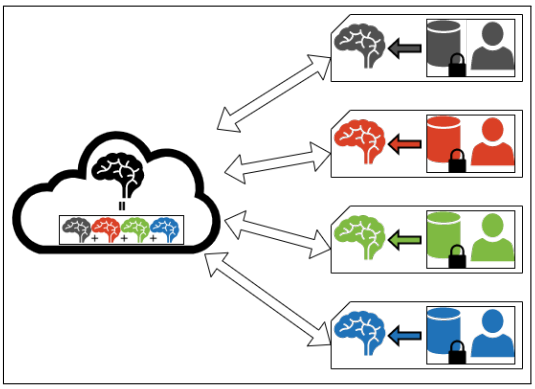

Fig. 1.2: Diagrama de una red de Federated Learning

Al contrario del machine learning, que reunía toda la información en un mismo computador, el aprendizaje federado busca entrenar un algoritmo en cada uno de los dispositivos que participen en la red para luego combinar estos algoritmos. Una vez combinados son mandados de vuelta a los participantes de la red para poder mejorarlos y que estos sean más precisos gracias a la información de otros participantes.

Capítulo 2

Desarrollo

2.1. Requisitos del sistema

Como bien se ha mencionado previamente, este proyecto trata de adaptar el sistema de recomendación realizado por R.Sánchez [**sanchez-corcueraPersuasionbasedRecommenderSystem2020**] a un sistema de recomendación basado en Federated Learning que permita el aprendizaje colaborativo de los participantes. Es por ello que para conocer los requisitos iniciales de diseño del sistema de recomendación se remite al trabajo del autor.

En este apartado se presentarán únicamente los requisitos de la adaptación del sistema y se partirá del trabajo realizado por R.Sanchez. Se presentarán dos tipos de requisitos. En primer lugar los requisitos no funcionales de diseño, las condiciones que debe cumplir la especificación del diseño para cumplir los objetivos del proyecto. En segundo lugar, los requisitos funcionales, las condiciones que ha de cumplir el sistema para ser capaz de funcionar sobre el hardware del que se dispone.

2.1.1. No Funcionales

En cuanto a los requisitos no funcionales se encuentran todos aquellos que se han tenido en cuenta a la hora de plantear el diseño del sistema, es decir, todos los inherentes de la privacidad como patrón de diseño. Por ello el sistema :

RNF1 Mantendrá toda información de los usuarios en sus dispositivos.

RNF2 No expondrá la información de los modelos de inteligencia de los usuarios a otros usuarios.

RNF3 No discriminará entre los modelos de inteligencia artificial a la hora de tomar decisiones.

RNF4 No perjudicará deliberadamente al modelo de un participante.

RNF5 No compartirá información que no sea sintética.

RNF6 No extraerá información de los modelos de los participantes.

RNF7 Combinará los diferentes modelos de los participantes sin acceder a su información.

2.1.2. Funcionales

Los requisitos funcionales son básicos en este proyecto para que el sistema sea compatible y ejecutable en el hardware. Por ello, para que el sistema sea compatible con el harware se han definido los siguientes requisitos:

RF1 El sistema ha de ser ejecutable bajo la arquitectura ARMv8.

RF2 El sistema ha de ser ejecutable bajo la arquitectura x86 y x64.

RF3 El sistema ha de ser ejecutable bajo la arquitectura AArch64.

RF4 El sistema ha de ser ejecutable en Linux.

RF5 El sistema ha de ser ejecutable en Windows.

Además, al margen de los requisitos de compatibilidad también se encuentran los requisitos de recursos, y es que teniendo un hardware poco potente es imprescindible que el software sea lo más óptimo posible para aprovecharlo al máximo. Es por eso que tambien se definen los siguientes requisitos:

RF6 El sistema ha de ser capaz de entrenar los modelos en un espacio de tiempo razonable, siendo este siempre inferior a los 100 segundos por cada 10 epochs.

RF7 El sistema de recomendación no puede colapsar el dispositivo que lo ejecute,

RF8 El sistema de recomendación debe permitir que el dispositivo pueda funcionar con normalidad durante su ejecución.

2.2. Gestión de riesgos

El motivo de esta sección viene de la definición del modelo en espiral como metodología clave para el desarrollo de este proyecto. Este modelo recoge la identificación y gestión de riesgos como pieza fundamental del ciclo de vida, y en un proyecto con tal complejidad técnica y con tantos dispositivos involucrados, es indispensable tener un plan de acción ante cualquier incidente.

Con tantos dispositivos que gestionar, el proceso de configuración de estos puede llegar a ser algo pesado, y más si alguno de estos falla o se rompe. Si se rompiese o se corrompiese una tarjeta SD de una Raspberry Pi habría que reinstalar y configurar todos los paquetes a mano, proceso que puede ser bastante largo puesto que requiere de interacción humana. Para evitar esto se ha tomado la dedcisión de crear un script de aprovisionamiento tanto para la Nvidia Jetson Nano como para las Raspberry Pis, de esta manera, en caso de que una tarjeta SD se averiase, bastaría con cambiarla, configurar el protocolo SSH para acceder remotamente a ella y ejecutar el script. Cuando el proceso terminase el dispositivo estaría listo para poder ejecutar el proyecto sin ningún problema.

Las tareas de inteligencia Artificial no son especialmente livianas para un dispositivo, lo que durante un entrenamiento prolongado en el tiempo podría incurrir en un aumento de temperatura de este. Para evitar que los dispositivos alcancen temperaturas elevadas o que sufran cualquier tipo de cortocircuito por su manipulación inadecuada se ha optado por instalarlos en un rack (Fig 2.1).

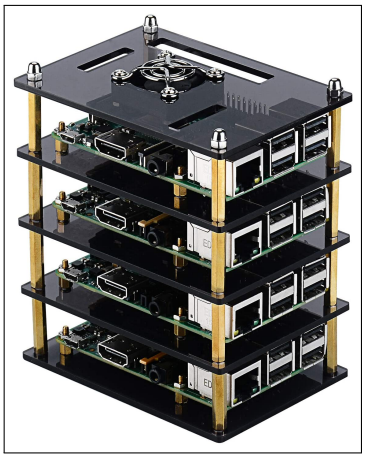

Fig. 2.1: Rack para Raspberry Pi (Fuente: Amazon[**ParaRaspberryPi**])

Esta carcasa aparte de proteger los dispositivos viene equipada con ventiladores que ayudaran a que la temperatura de estos sea siempre la adecuada.

Durante el desarrollo se tiene que acceder a estos dispositivos en multitud de ocasiones, para ello se utiliza el protocolo SSH. Sin embargo, al realizarse este proyecto en un ámbito doméstico, existe la probabilidad de que el router sea reiniciado o apagado por cualquier causa externa al proyecto. En tal caso puede que las direcciones IP sean sustituidas y haya que realizar un análisis de la red o acceder fisicamente a estos dispositivos para averiguar la IP. Para evitar este proceso se ha instalado el paquete *AVAHI*[1] que permite acceder mediante SSH a las raspberrys sustituyendo la dirección IP por el nombre de servidor de los dispositivos.

Para gestionar los cambios que se irán haciendo sobre el software sin problemas es indispensable contar con herramientas de control de versiones. Estas herramientas permiten gestionar cuándo se han realizado qué cambios y en caso de necesitarse se podrían revertir con facilidad. En este caso, tanto para el desarrollo del proyecto como para el desarrollo de la memoria se utilizará Git como controlador de versiones. Además, se utilizará Github como repositorio de código en la nube, para así poder clonar el repositorio a los dispositivos facilmente.

2.3. Especificación del diseño

Teniendo en cuenta los anteriores requisitos y los objetivos del proyecto la especificación del diseño concretará como se han diseñado las diferentes funcionalidades y como se ha implementado el Federated Learning en el proyecto.

[1] AVAHI es un servicio que facilita el descubrimiento de dispositivos en la red local a través de mDNS/DNS-SD.

2.3.1. Instalación de los requisitos en el hardware

2.4. Tecnologías utilizadas

2.4.1. Sistema de recomendación

El sistema de recomendación de R.Sanchez utiliza LightFM[2] para crear los modelos de inteligencia artificial. Para implementar el aprendizaje federado se ha tenido que modificar gran parte del sistema de recomendación previo y se ha tenido que hacer uso de las utilidades que proporciona LightFM que se van a explicar en los siguientes apartados.

2.4.1.1. Creación del modelo

El paso de creación del modelo es un paso complejo puesto que conlleva la correcta gestión de muchos datos e información. Asimismo, estos datos han de estar correctamente estructurados y construidos, puesto que LightFM requiere que la información este convertida a matrices dispersas y no tolera elementos vacíos en ellas. Para ello LightFM proveé de herramientas de creación de datasets que facilitan la creación de las matrices, de forma que sea más fácil incluirlas en el modelo.

Entre estas erramientas se encuentran:

- *build_user_features(...)* Permite crear la matriz CSR de usuarios y atributos de usuarios.
- *build_item_features(...)* Permite crear la matriz CSR de items y atributos de items.
- *build_interactions(...)* Permite crear las matriz COO de interacciones y las matriz COO de sus correspondientes pesos.

Hay que tener en cuenta que las matrices CSR (Compressed Sparse Row) hacen referencia a las matrices que admiten operaciones matriciales y acceso eficientemente; y que las matrices COO (Coordinate list) hacen referencia a las matrices que soportan modificaciones eficientemente, son generalmente utilizadas para construir matrices.

2.4.1.1.1. Atributos de los usuario Para crear el modelo es necesario, entre otras cosas, disponer de las características de los usuarios. En este caso contamos con multitud de atributos sobre cada usuario: edad, género, nivel educativo, país, cultura de trabajo, perfil PST (Pinball, Shortcut, Thought-ful), barreras, intenciones y confianza.

Para convertir toda esa información del usuario a la matriz CSR que exige LightFM habrá que llamar a *build_user_features(...)* con los IDs de usuario y y sus atributos, formando una lista que tenga listas de los IDs de los usuarios con sus listas de atributos, es decir:

$$\Big[\Big[userId_1 \quad \Big[feature_{11} \quad \cdots \quad feature_{1w}\Big]_1\Big] \quad \ldots \quad \Big[userId_v \quad \Big[feature_{v1} \quad \cdots \quad feature_{vw}\Big]_v\Big]\Big]$$

[2]LightFM es una implementación en python de varios algotitmos populares de recomendación.

Teniendo en cuenta que:

$$\begin{aligned} v &\leftarrow \textit{Cantidad de IDs de usuario} \\ w &\leftarrow \textit{Contidad de atributos por usuario} \\ \textit{userId}_v &\leftarrow \textit{ID del usuario } v \\ \textit{feature}_{vw} &\leftarrow \textit{Atributo } w \textit{ del usuario } v \end{aligned}$$

Una vez creada la lista y pasado como parámetro al método, obtendremos la matriz CSR de atributos de usuario.

2.4.1.1.2. Atributos de los elementos Además de los usuarios y de sus atributos, también se ha de disponer de los elementos a ordenar en el ranking, llamados items, y de sus características. En este caso estos items representan las estrategias de persuasión por las que se preguntó a los usuarios en el cuestionario. Sin embargo, al contrario que en el punto anterior, no contamos con tantos atributos sobre estos items, sino que cada estrategia de persuasión cuenta con dos atributos llamados dimensiones. Estos atributos no se encuentran presente en todas las estrategias, lo que da como resultado que haya algunas que tengan dos, una o ninguna dimension.

Para convertir toda esa información de los items a la matriz CSR que exige LightFM habrá que llamar a *build_item_features(...)* con los IDs de las estrategias y y sus atributos, formando una lista que tenga listas de los IDs de las estrategias con sus listas de atributos, es decir:

$$\Big[\Big[itemId_1 \quad \Big[feature_{11} \quad \cdots \quad feature_{1w}\Big]_1\Big] \quad \ldots \quad \Big[itemId_v \quad \Big[feature_{v1} \quad \cdots \quad feature_{vw}\Big]_v\Big]\Big]$$

Teniendo en cuenta que:

$$\begin{aligned} v &\leftarrow \textit{Cantidad de IDs de estrategias} \\ w &\leftarrow \textit{Contidad de dimensiones por estrategia} \\ \textit{itemId}_v &\leftarrow \textit{ID de la estrategia } v \\ \textit{feature}_{vw} &\leftarrow \textit{Atributo } w \textit{ de la estrategia } v \end{aligned}$$

Una vez creada la lista y pasado como parámetro al método, obtendremos la matriz CSR de atributos de usuario.

2.4.1.1.3. Interacciones Para crear un modelo de LightFM se necesita
user_features, item_features
model = LightFM()

2.4.1.2. Ajuste

2.4.1.3. Predicción

2.4.1.4. Evaluación

La evaluación de los resultados obtenidos de las predicciones de los modelos es un apartado fundamental que permite analizar la eficacia de estas predicciones. Además, la utilización de métricas

y técnicas permitirá poder comparar los resultados entre sí para poder demostrar si la predicción es correcta, aceptable o erronea.

2.4.1.4.1. Métricas

2.5. Consideraciones sobre la implementación

En este capitulo se describirán tanto las causas que han llevado a adoptar la previa especificación del diseño del sistema como las razones detrás de ellas. Con el objetivo de ayudar a entender el porqué de las decisiones tomadas se hablará sobre la privacidad, seguridad, la combinación de los modelos y las limitaciones técnicas presentes en este proyecto.

2.5.1. Privacidad

Con el objetivo de no vulnerar la privacidad de los datos, la agregación de los modelos será realiza por consenso sobre usuarios sintéticos. De esta forma, se evitan problemas derivados del robo de datos y las consecuencias tanto legales como personales de este.

Partiendo del principio de que si no circula ningún dato de ningún usuario por la red no se corren tantos riesgos se plantea este sistema de agregación como una alternativa viable a los sistemas tradicionales de ensamblado.

2.5.2. Seguridad

Este sistema está expuesto a los ataques de seguridad derivados de la implementación del Federated Learning. Sin embargo, debido a la naturaleza del algoritmo de consenso, algunos ataques podrían ser minimizados o incluso erradicados.

Un ejemplo de ello es el ataque por envenenamiento. Existen dos tipos de envenenamiento, por datos y por modelo, pero ambos consisten en que un participante de la red, con el objetivo de dañar tanto al sistema como a los participantes, envía modelos de inteligencia artificial (o los datos si es por envenenamiento de datos) debidamente manipulados. En los sistemas de agregación de modelos de inteligencia artificial por ensamblado este tipo de ataque suele ser bastante dañino, ya que es capaz de distorsionar por completo los resultados del modelo final con lo enviado por el atacante.

Sin embargo, debido a la condición del sistema de agregación por consenso, este tipo de ataques no son tan efectivos, ya que el modelo con datos envenenados podría ser ignorado si el método por el que se realiza el consenso es la moda. Con la moda se elegiría el ranking en función de los valores más frecuentes, es decir, el modelo del atacante sería ignorado por presentar predicciones que no se ajustan a ninguno de los demás participantes.

Cabe destacar que con el objetivo de detectar ataques y encontrar posibles modelos maliciosos se podrían descartar del proceso todos aquellos modelos que presenten unas predicciones totalmente diferentes. Se podría analizar la varianza de las predicciones de cada modelo para descartar del proceso a presuntos modelos envenenados.

2.5.3. Combinación de modelos

Debido a que el principal objetivo de este proyecto es el desarrollo de un sistema de recomendación con aprendizaje federado con la privacidad como patron de diseño, la combinación de los diferentes modelos de los participantes ha sido un factor muy importante a tener en cuenta. En el Machine Learning tradicional esta agregación de modelos se producía mediante el ensamblado de modelos. Sin embargo, este proceso no es compatible con este proyecto puesto que para poder ensamblar los modelos habría que compartir información de los participantes de la red.

Para evitar compartir ningún tipo de información que comprometiese la seguridad y privacidad de los participantes de la red de aprendizaje federado pero conseguir que los participantes aprendiesen entre ellos, se optó por compartir los modelos de inteligencia artificial exclusivamente.

2.5.4. Limitaciones técnicas

Debido a que los dispositivos que iban a formar la red iban a ser 4 Rasberry Pis modelo 3b+, el procesado de los datos era un proceso de especial importancio por la *escasa* capacidad de computo de estos dispositivos. Estos dispositivos, como se puede ver en el capítulo de especificación del diseño, son dispositivos que tienen cuatro núcleos y 1.4GHz de frecuencia, lo que comparado con un ordenador de sobremesa actual supone una gran diferencia de potencia de cálculo.

Además, esa no es la unica diferencia que tienen respecto a los ordenadores de escritorio, las Raspberry Pis funcionan sobre arquitectura ARMv8 y no sobre las tan usadas x64 y x86 . Esto implica un gran problema a la hora de instalar los paquetes necesarios para que el proyecto sea ejecutable en estos dispositivos, ya que la mayoría de los paquetes necesarios están pensados para las arquitecturas más comunes (x64 y x86). Sin embargo, se han podido encontrar las versiones funcionales de todos los paquetes para estos dispositivos.

El dispositivo que iba a agregar los datos, la Nvidia Jetson Nano, ha sido más sencilla de configurar puesto que esta funciona sobre arquitectura AArch64, extensión de 64bits de la arquitectura ARM.