UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Aplicación del Aprendizaje Semisupervisado en el descubrimiento de ataques a Sistemas de Recomendación

Presentado por Patricia Hernando Fernández
en Universidad de Burgos — 12 de noviembre de 2022
Tutor: Álvar Arnaiz González

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Álvar Arnaiz González, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que la alumna D.ª Patricia Hernando Fernández, con DNI 71362977A, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado «Aplicación del Aprendizaje Semisupervisado en el descubrimiento de ataques a Sistemas de Recomendación».

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, 12 de noviembre de 2022

Vº. Bº. del Tutor:

D. Álvar Arnaiz González

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

Introducción

1.1. Preámbulo

A diferencia de unas décadas atrás, la sociedad actual está gobernada por los datos. La transición a la era de la información puede ser compleja para determinados colectivos y, consecuentemente, diversos sistemas auxiliares han sido desarrollados con el fin de resumir información y facilitar la toma de decisiones. Entre ellos se encuentran los sistemas de recomendación, que son herramientas que pretenden realizar sugerencias de objetos que pueden resultar interesantes para un determinado perfil.

Económicamente, este tipo de algoritmo es un claro objeto de interés, puesto que puede influir en la toma de decisiones de los compradores y hacer que se inclinen por un determinado producto (por ejemplo, el que tenga una mejor valoración). Los atacantes conocen esta situación y manipulan estas herramientas mediante el uso de perfiles falsos con el fin de beneficiar sus productos o perjudicar los de la competencia.

Este proyecto de investigación pretende explorar cómo el aprendizaje semisupervisado puede ayudar a detectar los ataques a sistemas de recomendación, diferenciando entre perfiles genuinos e inyectados, además de comprobar la veracidad de los planteados por otros investigadores.

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

Conceptos teóricos

Se sintetizarán a continuación algunos de los conceptos teóricos más relevantes para la correcta comprensión del documento.

3.1. Apendizaje automático

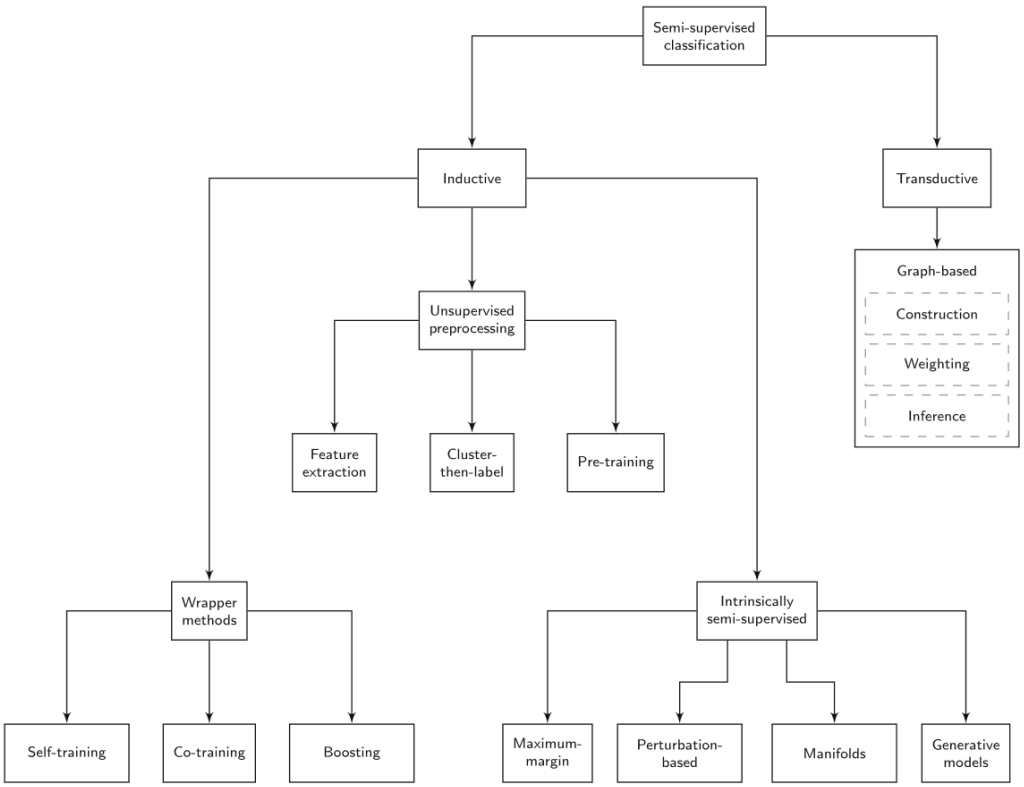
Se denomina aprendizaje automático a aquella rama de la inteligencia artificial cuyo objetivo es desarrollar métodos que permitan que un algoritmo mejore su rendimiento mediante la experiencia y procesado de datos. Consecuentemente, los modelos entrenados realizarán predicciones cada vez más precisas como resultado del algoritmo implementado.

Dentro del aprendizaje automático se diferencian tres grandes grupos en función del tipo de entrada que sea consumida: el aprendizaje supervisado (datos etiquetados), el no supervisado (datos no etiquetados) y el semisupervisado (datos etiquetados y no etiquetados), siendo esta última categoría objeto de estudio en este proyecto de investigación.

3.2. Aprendizaje semisupervisado

Como se ha mencionado anteriormente, se denomina aprendizaje semisupervisado a aquel conjunto de algoritmos que utiliza datos etiquetados y no etiquetados para realizar tareas de aprendizaje. Inicialmente, se pueden diferenciar dos categorías [3]: los métodos inductivos, cuyo objetivo principal es construir un clasificador que genere predicciones para cualquier entrada y los métodos transductivos, cuyo poder de predicción está limitado a los objetos utilizados en la fase de entrenamiento.

Figura 3.1: Clasificación sugerida por [3]



Prescindiendo de los métodos transductivos por ser menos versátiles y útiles en nuestro propósito, los métodos inductivos se subdividen en tres grupos [3]: *wrapper methods* (o métodos de envoltura), *unsupervised preprocessing* y *intrinsically semi-supervised*, siendo materia de estudio los métodos de envoltura.

Métodos de envoltura

Estos modelos utilizan uno o más clasificadores que son entrenados iterativamente con los datos etiquetados de entrada, además de con datos pseudoetiquetados. Se denomina pseudoetiquetado a aquellos datos que inicialmente no estaban etiquetados, pero acabaron estándolo por iteraciones previas de los clasificadores.

Consecuentemente, el procedimiento consta de dos fases que se repiten en cada iteración: el entrenamiento y el pseudoetiquetado. Durante el entrenamiento, los clasificadores se alimentan de datos etiquetados (o pseudoetiquetados). En la fase de pseudoetiquetado, se utilizan datos no

etiquetados para que sean procesados por los clasificadores previamente entrenados.

Dentro de esta categoría, se pueden diferenciar tres grandes grupos: *self-training*, que utilizan únicamente un clasificador, *co-training* [3], que utilizan más de uno y los *pseudo-labelled boosting methods*, que construyen clasificadores individuales que se alimentan de las predicciones más fiables. Se estudiará más en profundidad los métodos *co-training*.

Co-training y Co-forest

En estos algoritmos, varios clasificadores son entrenados iterativamente utilizando datos etiquetados y añadiendo las predicciones (resultados) más confiables al conjunto para ser utilizadas en las siguientes iteraciones. Para que los clasificadores sean capaces de generar información distinta, generalmente se divide el conjunto de entrada según alguna característica (no siendo estrictamente necesario).

//TO - DO: Añadir información sobre las dos vistas independientes del co-training (otros métodos no tienen esta limitación)

El llamado *co-forest*, es un modelo dentro de *co-training*. En su desarrollo, se utilizan árboles de decisión (a mayor número mejor resultado), que son entrenados utilizando los datos etiquetados. En cada iteración, además, se añade al conjunto de datos nuevos elementos pseudoetiquetados. Estos elementos son el resultado de los elementos comunes (nuevas etiquetas) del resto de árboles en la fase anterior, y se usan durante una fase de entrenamiento. Sin embargo, se eliminan una vez se ha completado (la siguiente iteración se realiza inicialmente sólo con los datos etiquetados, etc.), consiguiendo así resultados certeros.

3.3. *Co-Forest*

Algoritmo

Tratamiento del ruido y teoría de errores

Como se muestra en [10], de acuerdo con [1] si el tamaño de los datos utilizados en el entrenamiento (m), la tasa de ruido (η), el error de la hipótesis en el peor caso (ϵ) y una constante (c) cumplen la relación de la ecuación 3.1, entonces la hipótesis aprendida por el árbol h_i (que minimiza el desacuerdo en un conjunto de muestras de entrenamiento con ruido) converge a la hipótesis verdadera.

$$m = \frac{c}{\epsilon^2(1-2\eta)^2} \tag{3.1}$$

De acuerdo con [10], se puede obtener la función de utilidad mostrada en la ecuación 3.2 operando en la expresión 3.1.

$$f = \frac{c}{\epsilon^2} = m(1-2\eta)^2 \tag{3.2}$$

Como se ha mostrado en el pseudocódigo, en la iteración i-ésima un determinado árbol se entrena con sus datos etiquetados L_i y un conjunto de pseudo-etiquetas $L'_{i,t}$. Si se considera que el OOBE cometido en L por H_i es $\hat{e}_{i,t}$, entonces se puede estimar que el número de pseudo-etiquetas erróneas en $L'_{i,t}$ equivale a $\hat{e}_{i,t} * W_{i,t}$ (se recuerda al lector que $W_{i,t}$ es el sumatorio de la confianza de predicción (grado de acuerdo) de H_i en cada muestra de $L'_{i,t}$). Por lo tanto, la tasa de ruido que se encuentra en $L_i \cup L'_{i,t}$ es la estimada por la ecuación 3.3, donde W_0 y η_0 son los parámetros correspondientes a L.

-> CONSULTAR: creo que está bien por ser estimación, pero aún así cada árbol se entrena con un subconjunto aleatorio de L, no con L al completo... Revisar con Alvar.

$$\eta = \frac{\eta_0 W_0 + \hat{e}_{i,t} W_{i,t}}{W_0 + W_{i,t}} \tag{3.3}$$

Concordando con 3.2, la función de utilidad f es inversamente proporcional a ϵ^2. Por lo tanto, si se quiere reducir el error cometido, se debe aumentar la utilidad de cada árbol en cada iteración [10]. Consecuentemente, se debe cumplir la ecuación 3.4.

$$\frac{\hat{e}_{i,t}}{\hat{e}_{i,t-1}} < \frac{W_{i,t-1}}{W_{i,t}} < 1 \tag{3.4}$$

Parámetros del algoritmo

Intuitivamente se puede deducir la ecuación 3.4, ya que el error debe disminuir y la confianza de predicción aumentar con cada iteración. Sin embargo, aunque esto se cumpla, puede ser que se deje de cumplir que $\hat{e}_{i,t} W_{i,t} < \hat{e}_{i,t-1} W_{i,t-1}$, ya que puede ocurrir que $W_{i,t} >>> W_{i,t-1}$. Por este motivo y para cumplir con lo expuesto en 3.4, se limita W_{max} como se muestra en 3.5 al realizar el muestreo de U en el algoritmo.

$$W_{max} = \frac{\hat{e}_{i,t-1} W_{i,t-1}}{\hat{e}_{i,t}} > W_{i,t} \tag{3.5}$$

El algoritmo original propuesto por [4] y el utilizado en [10] dejan, sin embargo, una cuestión pendiente. Como se puede observar en la ecuación 3.5 W_{max} requiere para calcularse tanto el OOBE como la W obtenida en la iteración anterior, y ambos autores inician W a 0. Esto resulta en que en la primera iteración $W_{max} = 0$ y, por lo tanto, evita que se realice un muestreo de U para pseudo-etiquetar (pararía el algoritmo). En su tésis, Engelen [7] propone solucionar este problema iniciando $W = min(\frac{1}{10}|U|, 100)$, aunque destaca que imponer esta constante hace que el impacto de los datos sin etiquetar en el algoritmo dependa profundamente del tamaño del *dataset*.

3.4. Ataques a sistemas de recomendación

Los ataques a los sistemas de recomendación (generalmene denominados *shilling attacks* [5] o *profile injection attack* [8]) tienen como objetivo manipular las sugerencias que propone un determinado algoritmo para conseguir que un cliente se incline hacia un elemento deseado. Esta alteración del sistema se consigue inyectando perfiles falsos.

Múltiples estudios se han centrado en formalizar las características de estos ataques con el fin de detectarlos. Entre ellas se encuentran [5]:

- **Intención:** normalmente, se pretende manipular la opinión general acerca de un elemento (ya sea para bien o para mal). Según el objetivo se pueden diferenciar dos tipos de ataques: ***push attacks***, que pretenden hacer un objeto más atractivo o ***nuck attacks***, cuya intención es la contraria. En caso de que el atacante no busque alterar la opinión acerca de un producto sino restar credibilidad a un sistema (mediante valoraciones aleatorias), se habla de ***random vandalism*** [2].
- **Fuerza:** la calidad de los ataques se mide teniendo en cuenta el **tamaño del relleno** (número de valoraciones asignadas a un perfil atacante, que suele rondar entre el 1 y el 20 % del total de los ítems [5]) y el **tamaño del ataque** (número de perfiles inyectados en el sistema, rondando entre el 1 y el 15 %).
- **Coste:** se distinguen dos tipos: ***knowledge-cost***, que hace referencia al coste de construir perfiles y ***deployment-cost***, que es el número de perfiles que se deben inyectar para conseguir un ataque efectivo [8].

Modelo	I_S:	Valoración I_F:	I_0:	Valoración I_t:
Random	$\emptyset$	Aleatoria siguiendo una distribución normal definida por todas las valoraciones para todos los ítems del sistema $\mathcal{N}(\mu, \sigma)$.	$\emptyset$	máxima o mínima
Average	$\emptyset$	Aleatoria siguiendo una distribución normal definida por las otras valoraciones para ese ítem en concreto $\mathcal{N}(\mu_i, \sigma_i)$.	$\emptyset$	máxima o mínima

Tabla 3.1: Descripción de los ataques básicos [10]

Tipos de ataques

En la actualidad se distinguen multitud de ataques distintos. Con el fin de formalizarlos matemáticamente, se han establecido ciertos conjuntos de interés dependiendo de los ítems que contengan [10].

- I_S**:** conjunto de ítems seleccionados para recibir un tratamiento especial (puede ser vacío).
- I_F**:** conjunto de ítems seleccionados para «rellenar».
- I_0**:** conjunto de ítems pertenecientes al sistema de recomendación sin valorar.
- I_t**:** conjunto de ítems objetivo.

Ataques básicos

Se distinguen dos tipos: *random attack* y *average attack* [5]. Ambos tienen parámetros y características muy similares como se muestra en la tabla 3.1. La principal diferencia reside en que el *average attack* es mucho más potente debido a que cuenta con mayor información acerca del sistema: las valoraciones a los ítems de relleno siguen una distribución $\mathcal{N}(\mu_i, \sigma_i)$, en lugar de $\mathcal{N}(\mu, \sigma)$. Es decir, la valoración para un determinado ítem se adecúa a la distribución concreta de ese ítem en lugar de a la de todo el *dataset*.

Ataques con poco conocimiento del sistema

Los más populares son *bandwagon attack* (o *popular attack*) y *segment attack*. Sus principales rasgos se ilustran en la tabla 3.2.

Modelo	I_S:	Valoración I_F:	I_0:	Valoración I_t:
Bandwagon (average)	Ítems populares (valoración máxima) o ítems desfavorecidos (puntuacuón mínima) (reverse)	Aleatoria siguiendo una distribución normal definida por las otras valoraciones para ese ítem en concreto $\mathcal{N}(\mu_i, \sigma_i)$.	$\emptyset$	máxima o mínima (reverse)
Bandwagon (random)	Ítems populares (valoración máxima) o ítems desfavorecidos (puntuacuón mínima) (reverse)	Aleatoria siguiendo una distribución normal definida por todas las valoraciones para todos los ítems del sistema $\mathcal{N}(\mu, \sigma)$.	$\emptyset$	máxima o mínima (reverse)

Tabla 3.2: Descripción de los ataques con poco conocimiento del sistema.

La principal característica del *bandwagon attack* es que el conjunto I_S ya no está vacío, sino que contiene algunos de los ítems más populares de la base de datos [10]. Estos ítems recibirán también la máxima puntuación posible, de forma que ya no sólo se puntúa el conjunto objetivo. Existe una variante de este ataque llamada *reverse bandwagon attack*, cuyo objetivo es hacer *nuke*. De esta forma, I_S contiene los ítems menos populares y reciben la puntuación mínima (junto con I_t).

En el *segment attack*, se realiza un pequeño «estudio de mercado» y se introduce en I_S ítems en los que estaría interesado un usuario que fuese a valorar también I_t (de forma que el ataque es más realista).

Ataques con gran conocimiento del sistema

Este tipo de ataques resulta menos relevante que los anteriores debido a la dificultad de su ejecución. En la mayoría de los casos, se necesita una gran cantidad de información, siendo poco realista que se produzca una situación de estas características en la realidad.

Por ejemplo, el llamado *perfect knowledge attack* [8] basa su efectividad en reproducir la distribución exacta de la base de datos real (exceptuándo los ítems objetivos). El *sampling attack* construye los perfiles a inyectar basándose en una muestra de perfiles reales [5].

Como se puede intuir, conocer datos estadísticos exactos sobre una base de datos o metadatos asociados a perfiles de usuarios es poco realista (cada

Modelo	Estrategia de ofuscación
Noise Injection	$\forall i \in I_F \cup I_S : R_i = r_i + aleatorio * \alpha$
Target Shifting	$\forall i \in I_F \cup I_S : R_i = r_i; I_T : r_{max} - 1$ o $r_{min} + 1$
AOP	I_F escogido del top ítems más populares.

Tabla 3.3: Descripción de las estrategias de ofuscación

vez menos debido a las mayores medidas de seguridad) y por lo tanto estos ataques resultan meramente teóricos.

Ataques ofuscados

Los ataques ofuscados [5] se basan en intentar «camuflar» los perfiles inyectados haciéndolos pasar por reales. Algunas de las características de su implementación se pueden consultar en la tabla 3.3

El ataque de *noise injection* introduce a los conjuntos I_S e I_F un «ruido» (número aleatorio que sigue una distribución Gaussiana) multiplicado por una constante α. *Target shifting* incrementa (o decrementa) en una unidad la valoración de I_t con el fin de crear diferencias entre ataques similares sin influir excesivamente el resultado y el *Average over popular items (AOP)* pretende ofuscar el *average attack* cambiando la forma de selección de I_F (en lugar de seleccionar los ítems del conjunto total de la colección, se seleccionan los X % ítems más populares).

Otros tipos de ataques

Además de los ataques previamente ilustrados, existen otros con objetivos más diversos o estrategias distintas. El anteriormente mencionado *random vandalism* (cuya intención únicamente es degradar la calidad del recomendador para causar descontento entre los usuarios) pertenece a esta categoría. Se pueden distinguir, además, ataques basados en copiar comportamientos de usuarios influyentes (modelo *PUA* (*Power User Attack*)) o ítems poderosos (modelo *PIA* (*Power Item Attack*)) [5]. Sin embargo, son menos abundantes.

Técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Si se han estudiado diferentes alternativas de metodologías, herramientas, bibliotecas se puede hacer un resumen de los aspectos más destacados de cada alternativa, incluyendo comparativas entre las distintas opciones y una justificación de las elecciones realizadas. No se pretende que este apartado se convierta en un capítulo de un libro dedicado a cada una de las alternativas, sino comentar los aspectos más destacados de cada opción, con un repaso somero a los fundamentos esenciales y referencias bibliográficas para que el lector pueda ampliar su conocimiento sobre el tema.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Trabajos relacionados

Dentro de este proyecto se pueden diferenciar distintas líneas de investigación, principalmente las dirigidas hacia el desarrollo y comprensión de los algoritmos de aprendizaje semisupervisado y las centradas en formalizar ataques a sistemas de recomendación.

Aprendizaje semisupervisado aplicado a la detección de ataques en sistemas de recomendación

Co-Forest aplicado a la detección de ataques [10]

En esta sección, el artículo fundamental es «*Semi-supervised recommendation attack detection based on Co-Forest*» [10]. En este *paper*, se propone un método de detección basado en Co-Forest y se producen distintas comparativas con otros algoritmos para comprobar su eficacia, consiguiendo unos resultados muy aceptables. Partiendo de esta base nace el presente documento, que pretende explorar la solución propuesta por estos autores y expandirla.

Naive Bayes aplicado a la detección de ataques [9]

También es muy relevante citar el trabajo expuesto en «*HySAD: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation*» [9], puesto que propone una aproximación Naive Bayes para separar perfiles de atacantes de perfiles genuinos y además utiliza los tipos de *datasets* que son probados posteriormente por Zhou y Duan (Amazon, Netflix y MovieLens). Es de los primeros trabajos basados en algoritmos semisupervisados que se encuentra en las bibliografías principales de los investigadores posteriores.

Ataques en sistemas de recomendación

La importancia de proteger los sistemas de recomendación ha sido contemplada desde principio de siglo, siendo común la proposición de otros tipos de aprendizaje para detectar los ataques.

Recolección de los tipos de ataques y propuestas de reconocimiento [5]

Respecto a la descripción de los tipos de intrusión, la correcta definición formal (matemática) de sus parámetros y una recopilación de la gran mayoría de ataques existentes, es fundamental referenciar el artículo de Mingdan, Quingshan «*Shilling attacks against collaborative recommender systems: a review*» [5]. Se trata de una recopilación reciente (2018) de las principales investigaciones de los autores más populares en la materia que destaca por su completitud.

Definición de conceptos [8]

Previo a este documento,también es relevante contemplar otros trabajos, como la tesis de William y Mobasher «*Thesis: Profile injection attack detection for securing collaborative recommender systems.*» [8]. En ella se introduce el concepto de inyección y se parametrizan características como el tamaño de ataque. Es destacable la autoridad de estos investigadores en la materia, siendo propietarios de muchos documentos de interés.

Primeras definiciones formales [6]

«*Collaborative recommendation: A robustness analysis*» [6], de O'Mahony y Hurley, es uno de los trabajos con más antiguedad pero mayor número de referencias que se encuentra. En él se definen los modelos de construcciones en base a conocimiento del sistema y pone a prueba la robustez de los recomendadores evaluando su estabilidad y precisión ante la presencia de perfiles inyectados (análisis matemático muy completo).

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2:343–370, 1988.

Robin Burke, Michael P. O'Mahony, and Neil J. Hurley. *Robust Collaborative Recommendation*, pages 805–835. Springer US, Boston, MA, 2015.

Jesper Engelen and Holger Hoos. A survey on semi-supervised learning. *Machine Learning*, 109, 02 2020.

Ming Li and Zhi-Hua Zhou. Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 37(6):1088–1098, 2007.

Si Mingdan and Qingshan Li. Shilling attacks against collaborative recommender systems: a review. *Artificial Intelligence Review*, 53, 01 2018.

Michael O'Mahony, Neil Hurley, Nicholas Kushmerick, and Guénolé Silvestre. Collaborative recommendation: A robustness analysis. *ACM Trans. Internet Technol.*, 4(4):344–377, nov 2004.

Jesper Van Engelen and Holger Hoos. Semi-supervised ensemble learning. master's thesis., 07 2018.

Chad Williams, Research Advisor, and Bamshad Mobasher. Thesis: Profile injection attack detection for securing collaborative recommender systems, 2006.

Zhiang Wu, Junjie Wu, Jie Cao, and Dacheng Tao. Hysad: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation. page 985–993, 2012.

Quanqiang Zhou and Liangliang Duan. Semi-supervised recommendation attack detection based on co-forest. *Comput. Secur.*, 109(C), oct 2021.