UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

19/20

Trabajo fin de grado

Estudio de técnicas de ataques en sistemas de recomendación aplicados al dominio turístico

Xiangnan Wu

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Estudio de técnicas de ataques en sistemas de recomendación aplicados al dominio turístico

Autor: Xiangnan Wu
Tutor: Alejandro Bellogin

julio 2020

Xiangnan Wu
Estudio de técnicas de ataques en sistemas de recomendación aplicados al dominio turístico

Xiangnan Wu
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi amor

Aprende como si fueras a vivir toda la vida,
y vive como si fueras a morir mañana.

Charles Chaplin

Agradecimientos

En primer lugar me gustaría agradecer a la Escuela Politécnica Superior por la aportación de la plantilla utilizada para la creación de mi trabajo fin de grado.

En particular quiero destacar el trabajo realizado por mi tutor Alejandro Bellogin por su apoyo incondicional durante la realización de este proyecto.

Y por supuesto no quiero olvidarme de mi novia Cristina, ella siempre estuvo a mi lado cuando la necesitaba. Finalmente me gustaría dar un agradecimiento en general a todos los personales de la EPS durante mi estancia en la escuela.

Resumen

Debido a la aparición de la Web2.0, los sistemas de recomendación han tenido un gran desarrollo en las ultimas décadas. Ante la era de la información masiva, los métodos de recomendación se presentan como una manera eficiente de filtrar información y escoger lo que realmente se quiere. El método más común y mejor valorado en la industria y la comunidad científica es el filtrado colaborativo. Este tipo de técnica se basa en la similitud de los ítems o del perfil de usuario, por lo que resulta bastante vulnerable a los ataques externos. En este contexto, el propósito de este trabajo es estudiar los diferentes ataques conocidos hasta ahora y poder detectarlos mediante algoritmos de aprendizaje automático.

En este documento se ha estudiado en profundidad la detección de los ataques basado en filtrado colaborativo así como los principales trabajos de investigación, lo cual ha producido las siguientes contribuciones:

1. En base al funcionamiento del filtrado colaborativo, se ha investigado sobre los conceptos de ataques y detecciones en este tipo de mecanismo.

2. Basándose en la estrategia de ataque se puede distinguir dos tipos: los estándares y los de confusión. En este trabajo se van a seleccionar los 3 tipos de ataques estándares más comunes: RandomAttack, AverageAttack, BandwagonAttack. Además, también se va a implementar un tipo de ataque híbrido que resulte de la combinación cualesquiera de los tres. Tras ejecutar las inyecciones de perfiles en sistema, se va a intentar evaluar la efectividad de dicho ataque mediante las métricas de HitRatio y Prediction shift. En esta parte se ha visto que fillerSize es un factor decisivo durante el proceso de ataque, ya que en numerosos escenarios este parámetro define el nivel de similitud entre ítems. En cambio, en aquellos sistemas que tienen una matriz de similitud densa, el factor attackSize es el que domina ya que hay una gran posibilidad de puntar un ítem popular.

3. Entender los algoritmos de detección basados en aprendizaje automático: BayesDetector, SemiSAD, PCASelectUsers. Analizar la idea básica de cada algoritmo y su proceso de implementación. Una vez implementados los modelos de detección se va a intentar realizar una evaluación sobre los distintos tipos de ataques mediante Precision, Recall y F-measure. Tras analizar los resultados obtenidos mediante estas métricas, llegamos a la conclusión de que la técnica de SemiSAD ha sido el mejor método.

Palabras clave

Filtro colaborativo, sistema de recomendación, detección de ataque, inyecciones de perfiles de usuario, ataques en sistema de recomendación

Abstract

Since the beginning of the Web 2.0, recommender systems have had a great development in the last decades. While facing the age of massive information, the recommendation methods are presented as an efficient way to filter information and choose what you really want. The most common and best valued method in industry and scientific circles is the collaborative filtering. This type of technique is based on similarities between items or user profiles, which makes it vulnerable to external attacks. In this context, the purpose of this work is to study the different proposed attacks until now and being able to detect them through machine learning algorithms.

In this document, we have studied deeply the detection of attacks based on collaborative filtering, together with their corresponding main research works, which has produced the following contributions:

1. Based on the behaviour of collaborative filtering, we have investigated about the concepts of attacks and detections in this type of mechanism.

2. Based on the attack strategy, it is possible to discriminate two types: standard and confusion. In this project, we are going to select the 3 most common standard attack types: RandomAttack, AverageAttack, BandwagonAttack. Besides, we will also implement a type of attack that is hybrid, resulting from the any combination of those three. After executing the profile injections in the system, we will try to evaluate the effectiveness of such attack by means of the metrics HitRatio and Prediction shift. In this part, we have observed that fillerSize is a decisive factor during the attack process, since in several scenarios this parameter defines the level of similarity between items. On the other hand, in those systems with a dense similarity matrix, the attackSize factor is the dominating one since there is a high probability to rate a popular item.

3. Understand the detection algorithms based on machine learning: BayesDetector, SemiSAD, PCA-SelectUsers. Analyse the basic idea behind each algorithm and its implementation process. Once these detection models are implemented, we will aim to perform an evaluation about the different typs of attacks using Precision, Recall, and F-measure. After analysing the obtained results with these metrics, we arrived to the conclusion that the SemiSAD technique is the best method.

Keywords

Collaborative filtering, recommendation system, attack detection, user profiles injections, attacks on recommender system

ÍNDICE

Listas

Lista de algoritmos

Lista de códigos

Lista de cuadros

Lista de ecuaciones

Lista de figuras

Lista de tablas

Lista de cuadros

1 Introducción

Internet es la tecnología decisiva de la era de la información. Gracias a ella las personas somos capaces de encontrar la información de forma casi instantánea. En internet se almacena todo tipo de informaciones y datos, los cuales, están a alcance de todo los que tengamos acceso a ella. Debido a estas cantidad masivas de información, en numerosas situaciones sufrimos una sobrecarga de datos que nos dificulta tomar decisiones, ya que tenemos miles de opciones a la vista.

Ante la gran cantidad de información ya es difícil ojear rápidamente todos los perfiles de información, por no hablar de encontrar fácilmente lo que los usuarios realmente necesitan. Esta gran cantidad de datos necesita ser procesada de alguna manera, es entonces cuando los pioneros del campo de tratamiento de información sacaron la idea del buscador: una herramienta capaz de filtrar la información y ayudar a los usuarios a encontrar lo que están buscando. Más tarde nacieron los sistemas de recomendación, al igual que los buscadores, los sistemas de recomendación tiene como objetivo ayudar a los usuarios a encontrar cosas. La única diferencia está en que un buscador es pasivo, ya que no empieza a buscar información hasta que no se le dice lo que se quiere buscar. En cambio, un sistema de recomendación puede hacer el filtrado de la información en cualquier momento y generar una recomendación para el usuario correspondiente. Está técnica está teniendo una gran popularidad en los últimos años, sobre todo los sistemas de recomendación personalizados adaptados a las preferencias de cada usuario. En muchos casos, los usuarios simplemente navegan por internet y, sin previo aviso, se le muestra un artículo sobre sus intereses. Aunque el usuario al principio no tenía intención de comprar o leer dicho artículo, muy habitualmente termina cediendo gracias a la excelente técnica de publicidad. Este es el poder de las técnicas de recomendación.

1.1. Motivación del proyecto

Desde un punto de vista empírico, la recomendación personalizada es una herramienta que evalúa un producto cuando el usuario ni siquiera sabe de su existencia, es decir, en base a su historial de valoración y las valoraciones de los otros usuarios sobre un artículo, intenta predecir una puntuación del usuario sobre un determinado ítem. Llegado a este punto, podemos decir que los sistemas de recomendación son dependientes del feedback de los usuarios: cuanto mayor sea este número, más consistente será su recomendación.

Gracias al desarrollo de la Web 2.0, las páginas estáticas de la Web pasaron a ser dinámicas, permitiendo la interacción entre usuarios y los sistemas de recomendación. Actualmente hay numerosas empresas que han triunfado gracias a las técnica de recomendación personalizada, tales como

Amazon, Facebook, Youtube, etc. La gran mayoría de los sistemas de recomendación de estas empresas se basan en el filtrado colaborativo, ya que es el más común y estudiado y tiene un rendimiento bastante bueno.

Aunque la aplicación del filtrado colaborativo fue un éxito en numerosas empresas, presenta un serio problema de seguridad y es que es muy sensible a los ataques externos. Los usuarios maliciosos intentan atacar los sistemas de recomendación para obtener beneficios, tanto materiales como inmateriales. Por ejemplo, un caso simple sería cuando se falsifica la valoración de un artículo para que los demás lo compren. Este acto parece que solo beneficia y no perjudica a nadie; sin embargo la realidad no es así. Las empresas que son incapaces de afrontar los ataques externos terminan cayendo, ya que tras un número de recomendaciones malas, los usuarios pierden progresivamente la confianza en la herramienta. La seguridad se convierte en una cuestión urgente.

1.2. Objetivos y carácter innovador

En este trabajo se quiere investigar en profundidad los ataques en filtrado colaborativo así como modelos de detección de ataque, especialmente en el sector de ocio (dominio turístico). Los resultados de la investigación obtenidos en este trabajo se pueden integrar fácilmente en todo el marco del algoritmo de recomendación para mejorar la robustez de los sistemas de recomendaciones personalizadas.

La robustez es un tema avanzado en el campo de estudio de los sistemas de recomendación. Aunque actualmente la seguridad es un tema cada vez más relevante, en comparación con otras áreas de investigación como la aplicación de redes neuronales (RNN), en los sistemas de recomendación ha habido pocos estudios. Los pioneros en este tema más conocidos (como Mehta, Hurley, Mobasher, Burke y O'Mahony) fundaron esta línea de investigación tanto para ataques como para la detección de los sistemas de recomendaciones, ellos fueron los que definieron por primera vez diversos ataques como técnicas de detecciones.

Sin embargo, el estudio de la robustez se ha basado en la gran mayoría de publicaciones sobre datos de Movielens, que se caracterizan por tener siempre un id de usuario, un id de ítem y un rating. En comparación, en este trabajo se van a utilizar también los datos de Foursquare, que contienen usuarios, las coordenadas geográficas de los ítems y los tiempos de visitas en cada lugar. El rating, al no ser un valor numérico dado, puede implicar procesado extra para tratar con este tipo de datos.

Como nunca se había hecho ataques ni sus detecciones sobre datos de este tipo (es decir, de Foursquare), no existe una comparativa de eficiencia que indique el correcto funcionamiento de las distintas técnicas. Como solución al problema anterior, se ha realizado tanto el generador de ataque como los detectores para Movielens y para Foursquare, de manera que se pueda comprobar su correcto funcionamiento según lo que indica la literatura.

1.3. Estructura del trabajo

Este documento consta de 5 secciones, la estructura está organizada de la siguiente manera:

En el **capítulo 1** hemos hecho una breve introducción al estudio de la detección de los ataques y

su importancia, describiendo el estado actual de la investigación en los campos relacionados con la robustez de recomendación. A continuación se han presentado las principales tareas realizadas y el carácter innovador de este trabajo.

En el **capítulo 2** se incluye el estado del arte en sí: en esta parte se describe detalladamente lo que es un sistema de recomendación y los diferentes algoritmos de recomendación centrándonos el filtrado colaborativo tanto basado en usuario como basado en ítem. También se exponen los riesgos presentes en los sistemas de recomendación, los tipos de ataques más comunes y las métricas utilizadas para evaluar los ataques. Junto con los ataques se encuentra también la detección, el cual se divide según las metodologías se basen en un estudio estadístico, aprendizaje automático supervisado o no supervisado.

En el **capítulo 3** se explica el diseño utilizado para implementar los mecanismos necesarios de este estudio basado en ataques y detección en los sistemas de recomendación. Luego se añade una explicación del desarrollo, es decir, cómo se ha llevado a cabo este estudio de investigación, las dificultades encontradas y cómo se han resuelto.

En el **capítulo 4** se muestran las pruebas y los resultados obtenidos. En esta parte se van a discutir los resultados tras evaluar los 4 tipos de ataques mediante la métrica de Hit Ratio y Prediction Shift. Posteriormente vamos a intentar identificar los ataques mediante los detectores implementados y a evaluar la eficacia de estos detectores mediante Precisión, Recall y F-measure.

En el **capítulo 5** se resume el trabajo de investigación llevado a cabo y se ofrece una visión general de las futuras orientaciones en el campo de la investigación de la robustez de los sistemas de recomendación.

2

Estado del arte

La era de la información es el nombre asociado al período de la historia de la humanidad que va ligado con el gran desarrollo experimentado en las tecnologías de la información y la comunicación. La información presente en los distintos medios de comunicaciones, sobre todo internet, hacen que la vida sea más variada y colorida. En internet se puede encontrar de todo: películas, libros, revistas científicas, etc. Incluso para un mismo tipo de productos o servicios puede haber distintas marcas, gamas y precios. Esto provoca la sobrecarga de información a la que estamos sometidos constantemente. La gente necesita una herramienta que les ayude a filtrar la información, este es el motivo por el cual nacieron los motores de búsqueda y los sistemas de recomendación.

Los motores de búsqueda son una herramienta que ayuda a las usuarios a encontrar la información deseada, sin embargo, como ya se ha mencionado previamente, el servicio proporcionado es pasivo ya que el usuario previamente necesita proporcionar las palabras clave a buscar. En cambio, los sistemas de recomendación, una de las aplicaciones de inteligencia artificial con servicio activo, tiene la ventaja de poder encontrar lo deseado incluso cuando el usuario no puede describir bien las características de lo que se desea. Para las técnicas de recomendación lo más importante es el algoritmo de recomendación en su capa subyacente, ya que la calidad del algoritmo es un elemento decisivo en su rendimiento.

El estudio de los algoritmos en este dominio siempre ha tenido un gran interés para los investigadores. Debido a que los algoritmos de recomendación actuales son bastante sensibles a los comportamientos históricos del usuario, por lo que presenta ciertos problemas de seguridad. En este capítulo vamos a hablar de los algoritmos de recomendación y el tema de seguridad en los sistemas de recomendación, presentando la relación existente entre los dos en cuanto a su teoría básica y sus investigaciones antecedentes.

2.1. Concepto de sistema de recomendación

Los sistemas de recomendación son herramientas software y técnicas que proporcionan sugerencias de un ítem a un usuario [1]. Una sugerencia puede ser qué ítem comprar, qué música escuchar, o qué nuevas noticias leer.

Los orígenes de estos sistemas se remontan a mediados de la década de los 2000. La aparición de estos sistemas estuvo estrechamente vinculada al nacimiento de la Web 2.0 y al cambio de de paradigma del marketing en Internet que ello supuso. Al comienzo, las páginas webs eran meros catá-

logos donde se presentaban sus productos o servicios. Sin embargo, con la aparición de plataformas dedicadas a vídeos, blogs, wikis y, sobretodo, las redes sociales, el usuario pasó de ser un mero consumidor de información unidireccional a interactuar de forma activa con el contenido que consumía. De esta manera, nacieron los sistemas de recomendación con el propósito de recoger el feedback directo de los usuarios sobre los contenidos que visitaban para poder realizar recomendaciones de ítems que creen que van a ser de su agrado [1, 2]. A su vez, un ítem se define generalmente como lo que el sistema recomienda al usuario, y que puede cambiar según el dominio de aplicación (una película en el caso de Netflix o una canción o un artista musical en Spotify).

2.2. Tipos de sistema de recomendación

Aunque se aplique al mismo conjunto de datos, los diferentes algoritmos de recomendación pueden actuar de forma totalmente distinta. Este es el motivo por el cual la investigación de los distintos tipos de algoritmos de recomendación se ha convertido en un campo de estudio cada vez más relevante. En esta sección vamos a centrarnos en aquellos tipos que consideramos más importantes y utilizados.

2.2.1. Filtrado colaborativo

Una de las primeras técnicas adoptadas en el área de los sistemas de recomendación fue el algoritmo de filtrado colaborativo. Se llaman colaborativos porque el sistema realiza las recomendaciones basándose en la valoraciones positivas de usuarios con un perfil de gustos similar al que se quiere recomendar.

Una gran ventaja de este tipo de sistemas de recomendación es que no necesitan guardar un gran catálogo de información, simplemente con el vector de valoraciones de cada usuario permite realizar recomendaciones de ítems de diferentes características, creando una sensación de novedad. Como es dependiente de las valores de los usuario, este tipo de recomendación no funcionan tan bien en sistemas con pocos usuarios e ítems. También presenta dificultad a la hora de realizar recomendaciones para nuevos usuarios o ítems [1, 2]. Más adelante, en la sección 2.3 vamos a centrarnos en más detalles de estos sistemas.

2.2.2. Basado en contenido

En este caso, el sistema intentar aprender a recomendar ítems que sean similares a los que el usuario había calificado positivamente en el pasado [3]. La similitud de los ítems es calculada en base a las características asociadas con otros ítems a comparar. Por ejemplo, si un usuario ha valorado positivamente una película que pertenece al genero de comedia, el sistema ya puede aprender a recomendar partiendo de este género. Los modelos de interés de los usuarios pueden construirse a menudo con métodos de aprendizaje automático como redes neuronales, SVM y árboles de decisión [3].

Esta técnica necesita almacenar la información del pasado en el sistema y analizar la similitud entre los ítems que han visitado y los que no han visitado para realizar la recomendación. El filtrado basado en contenido reduce la subjetividad de las opiniones de los usuarios y las falsas valoraciones y son capaces de evitar el problema de los nuevos ítems, ya que con tener el perfil de usuario basta con

calcular la similitud entre los ítems. Sin embargo, presenta problema con los usuarios nuevos ya que al no tener un perfil definido no sabe cómo recomendar. Además, este tipo de recomendadores solo pueden recomendar ítems limitados a perfiles similares, por lo que carece de novedad.

2.2.3. Basado en conocimiento

En este tipo de sistema se utiliza la información proporcionada por el usuario sobre sus preferencias o restricciones de manera directa. Son muy útiles en mercados muy cambiantes en los que los gustos del pasado reciente no son válidos en el presente. Este tipo de sistema resuelve los problemas de recomendaciones de los dos recomendadores anteriores frente a la aparición de nuevos usuarios e ítems [1, 2].

2.2.4. Filtrado demográfico

En este tipo de filtrado se intenta clasificar los usuarios según su perfil demográfico. Las recomendaciones se hacen basándose en las cualidades demográficas de este perfil en lugar de valoraciones de los diferentes ítems.

2.2.5. Recomendación híbrida

Debido a las diferentes condiciones de inicialización de los sistemas de recomendaciones y los distintos antecedentes de los usuarios, cada uno de los métodos descritos anteriormente presenta sus ventajas e inconvenientes. En la aplicación real, para mejorar la eficacia de un recomendador lo más fácil que se nos puede ocurrir es combinar las diferentes técnicas. Es muy habitual pedir que un nuevo usuario que se da de alta en un sistema cree un perfil de usuario. De esta manera se le pueden hacer recomendaciones en base a sus cualidades demográficas hasta que el número de feedback de ítems sea suficiente para poder realizarle recomendaciones con filtrado colaborativo o basado en contenido.

Aunque tras mezclar diferentes métodos hemos conseguido sintetizar las ventajas de cada uno, en escenarios específicos la recomendación híbrida puede rendir peor que un recomendador simple.

2.3. Filtrado colaborativo

El método de filtrado colaborativo se puede decir que es una forma de compartir información entre un grupo de personas, donde los usuarios se benefician de la experiencia de otros usuarios. Este sistema pretende recomendar los ítems a los usuarios a través de la similitud entre usuarios y/o la similitud entre los ítems. Este concepto fue propuesto por Goldberg en el año 1992, e inicialmente fue utilizado en los sistemas de Tapestry [4]. Dentro del filtrado colaborativo lo podemos subdividir en dos categorías: los métodos basados en modelos como regresión, factorización de matrices, learning to rank, modelos bayesianos, redes neuronales, etc., y los métodos basados en memoria, los cuales se basan en similitud entre usuarios o entre ítems.

A continuación vamos a centrarnos en estos últimos ya que son los que utilizaremos más adelante.

2.3.1. Filtrado colaborativo basado en usuario

Este tipo de algoritmo, también conocido como user-based, busca primero los usuarios con intereses similares al usuario que se pretende realizar la recomendación, posteriormente a partir de las valoraciones de los usuarios predice si el usuario a recomendar le puede interesar o no el ítem. Para los usuarios que tenga mayor similitud con el usuario target se le puede llamar vecinos. A continuación, vamos a explicar paso a paso como funciona el método user-based.

Paso 1. Calcular la similitud entre usuarios

Para encontrar los vecinos es necesario calcular las similitudes entre usuarios. Normalmente se usa la similitud de coseno o Pearson para realizar este calculo. El rango de los valores calculados a través del coeficiente de correlación de Pearson es [-1, 1]. El resultado tras calcular Pearson cuanto más cercano sea a 1, más relevantes serán los dos vectores (si cada fila de la matriz de puntuación es considerado como un vector) y más similares serán los dos usuarios. Por el contrario, si el coeficiente de correlación calculado es 0, significa que los dos vectores no están relacionados, lo que refleja que entre los dos usuarios no han tenido ninguna valoración en común; de igual modo, según más se acerca a -1, menos se parecen los dos usuarios. En la formula 2.1 se indica cómo se calcula el coeficiente de Pearson:

$$sim(u,v) = \frac{\sum_{i \in I: r(u,i) \neq \varnothing, r(v,i) \neq \varnothing} (r(u,i) - \bar{r}_u)(r(v,i) - \bar{r}_v)}{\sqrt{\sum_{i: r(u,i) \neq \varnothing, r(v,i) \neq \varnothing} (r(u,i) - \bar{r}_u)^2 \sum_{i: r(u,i) \neq \varnothing, r(v,i) \neq \varnothing} (r(v,i) - \bar{r}_v)^2}} \in [-1,1] \quad (2.1)$$

donde $\bar{r}_u$ indica el rating promedio del usuario u.

Paso 2. Encontrar los vecinos más próximos

A través del paso anterior generamos un conjunto de similitudes entre el usuario target y los otros usuarios del sistema. Lo ordenamos de mayor a menor, escogiendo los top-k usuarios más similares. El valor del k puede influir en la efectividad del sistema, si escogemos un k pequeño puede disminuir la precisión, en cambio, si escogemos un k demasiado grande se nos puede disparar la complejidad algorítmica. El factor K se debe escoger en base al tamaño y la densidad del conjunto de datos a procesar.

Paso 3. Predecir la puntuación del usuario

En el paso 2, hemos obtenido los top-k usuarios más similares. En este paso vamos a calcular la puntuación estimada que un usuario u le puede dar a un ítem i. Se calcula a partir de la siguiente fórmula:

$$\hat{r}(u,i) = C \times \sum_{v \in N_k(u), r(v,i) \neq \varnothing} sim(u,v) r(v,i) \quad (2.2)$$

donde C se usa para obtener una predicción en el rango deseado y $N_k(u)$ es el top-k de vecinos más similares a u.

Generar recomendaciones a partir de los datos de vecinos próximos es la mayor característica del método user-based. La ídea de este tipo de método está en que si a un grupo de usuarios similares al target les interesa un cierto tipo de ítem, a lo mejor al usuario target le puede interesar también. Este algoritmo tiene baja complejidad computacional, con una precisión bastante alta de acertar en lo que se recomienda. Sin embargo, cuando el conjunto de datos es escaso o cuando los usuarios han valorado pocos artículos, puede dar a lugar a una desviación en el cálculo de la similitud entre los usuarios e incluso hacer que el algoritmo no pueda encontrar ningún vecino. El segundo problema está en la escalabilidad del algoritmo, el número de usuarios e ítems aumentan considerablemente con el tiempo, y en ese momento el cálculo del vecino más cercano se vuelve cada vez más complejo por lo que el algoritmo no es adecuado para los casos en los que el volumen de datos es particularmente grande.

2.3.2. Filtrado colaborativo basado en ítem

Tras analizar el filtrado colaborativo basado en usuario, vemos que presenta muchas deficiencias. En el año 2001, el profesor Sarwar anuncio un nuevo método de recomendación desde el punto de vista de la similitud entre los ítems en vez de la similitud entre usuario [5]. Este método, también conocido como item-based, primero estima la similitud entre los distintos ítems y selecciona los k-items más parecidos. El algoritmo de item-based se puede dividir en tres partes fundamentales.

Paso 1. Calcular la similitud entre los ítems

Si cualquier ítem se puede convertir como target ítem, entonces obtenemos un matriz de similitud de ítems. Hay varias formas de calcular su similitud, pero las más comunes son el coseno y la correlación de Pearson. La siguiente fórmula muestra cómo se calcula la primera, puesto que la otra es equivalente a su versión con usuarios:

$$sim(i,j) = \frac{\sum_{u \in U: r(u,i) \neq \varnothing, r(u,j) \neq \varnothing} r(u,i) r(u,j)}{\sqrt{\sum_{u: r(u,i) \neq \varnothing} r(u,i)^2 \sum_{u: r(u,j) \neq \varnothing} r(u,j)^2}} \in [0,1] \tag{2.3}$$

Paso 2. Buscar los vecinos más próximos

Mediante la matriz de similitud, estimamos la similitud de un ítem con los otros y lo ordenamos de mayor a menor escogiendo los k primeros. Estos serán los k vecinos más próximos.

Paso 3. Predecir la puntuación del usuario

A través de los k vecinos más cercanos y la formula siguiente podemos predecir la puntuación que le dará el usuario u al ítem i.

$$\hat{r}(u,i) = C \times \sum_{j: r(u,j) \neq \varnothing} sim(i,j) r(u,j) \tag{2.4}$$

El propósito de este método es recomendar un ítem para usuarios con gustos similares, ya que hay mucha probabilidad de que el usuario puntué de la misma forma que sus usuarios con gustos similares. En la vida real, debido a que la profundidad de recomendación es limitado, es decir, la lista de recomendación presenta limites, por lo que el sistema solo obtiene los k primeros vecinos para realizar los cálculos y predecir la puntuación del target ítem.

2.4. Ataques en los sistemas de recomendación

Debido a que los sistemas de recomendación son de carácter abierto, esto permite que cualquier usuario sea libre de dar la valoración deseada y consultar las valoraciones de los otros. Esto hace que múltiples sistemas de recomendación sean vulnerables a los ataques externos, para fines tales como la competencia comercial. Los usuarios maliciosos pretenden implantar artificialmente una gran cantidad de perfiles falsificados en el sistema, convirtiéndoles en los vecinos más cercanos de la gran parte de los usuarios existentes. Este acto provoca un desplazamiento de las preferencias de los usuarios hacia el ítem objetivo o target, lo que hace que el sistema genere recomendaciones a su favor. Este tipo de ataque se denomina a menudo como ataque de inyección de perfiles de usuario.

Los ataques según su objetivo se pueden dividir en dos tipos: Push Attack o Nuke Attack. El primero tiene como objetivo aumentar la frecuencia de los objetos y el segundo por el contrario intenta disminuir la frecuencia recomendada. A continuación vamos a analizar la tabla 2.1 para ilustrar la idea básica de un ataque de inyección de perfiles.

User	i1	i2	i3	i4	i5	i6	i7
User1	+	-		+	+		+
User2	-	+	+	-	-		-
User3	+	-	+		-	-	-
User4	-	+	+	-			
User5	-		-	-	-		-
User6	+	-	+	+	+		+
User7		-	+	+	-	-	+
Bob	+	-	+	+	+		?
Attack1	+	-	+		-	-	-
Attack2	-	+	+	-			-
Attack3	-		-	-	-		-
Attack4	+	-	+	+	+		-
Attack5		-	+	+	-	-	-

Tabla 2.1: Esta base de datos muestra perfiles de usuarios auténticos y también los falsificados. En este ejemplo, el usuario Bob está buscando una predicción para el elemento 7, que es el objetivo de un **Nuke Attack**.

En la tabla anterior, los + representan recomendaciones positivas y los - recomendaciones negativas. Podemos observar que, inicialmente, la recomendación para Bob sobre el ítem 7 es un + debido a la similitud con los usuarios con perfiles auténticos. Sin embargo, tras inyectar los perfiles falsos y calcular nuevamente los vecinos, la estimación ha pasado a ser - debido a las valoraciones de los perfiles falsos.

2.4.1. Modelos de ataque

Para que un atacante pueda sacar provecho de las recomendaciones generadas por el sistema de recomendación, este debe falsificar los perfiles de usuarios escogiendo un modelo de ataque adecuado, el cual es la forma de generar perfiles de usuarios falsos utilizando los datos relativos de los usuarios auténticos presentes en el sistema de recomendación. Generalmente, si suponemos que el número de los ítems es n, entonces el perfil de usuario atacado es un vector de n dimensiones. Un perfil de usuario está formado por el conjunto de los ítems seleccionados (I_S), ítems de relleno (I_F), ítems no calificados (I_N) e ítems objetivos (I_T). I_S son seleccionados usando la métrica correspondiente al tipo de ataque, dependiendo del tipo puede tener una puntuación distinta. Los I_F son seleccionados aleatoriamente y su funcionalidad principal es aparentar que un usuario falsificado es un usuario normal del sistema. Los I_T como su nombre indica son los objetivos o targets a atacar, en los Push Attack suele tener la puntuación más alta y en caso contrario, para los Nuke Attack tener la puntuación más baja. Los I_N son los restos de ítems no valorados. Un perfil de usuario se compone de la manera mostrada en 2.1.

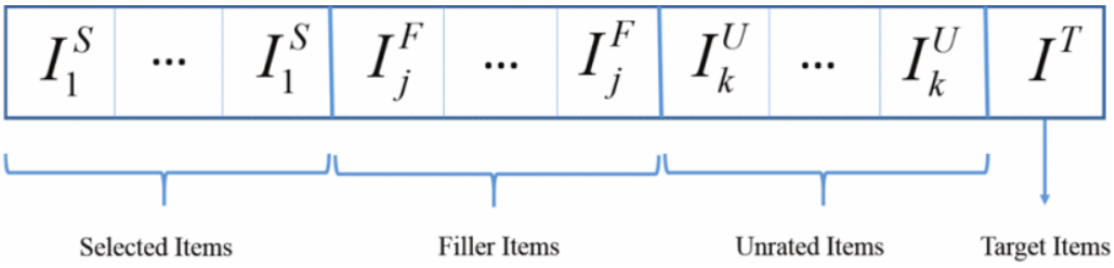


Figura 2.1: Marco general de como es un perfil de usuario

A la hora de crear los perfiles de usuarios, lo primero que hay que hacer es seleccionar el I_T, luego escoger los I_S e I_F. Con I_N no hay que hacer nada porque simplemente se dejan en blanco.

2.4.2. Tipo de ataque

Dependiendo del tipo de ataque que se use, las estrategias para seleccionar las diferentes partes de un perfil de usuario es totalmente distinto, como consecuencia el impacto sobre el sistema de recomendación original es claramente dependiente de la estrategia seleccionada. Se puede decir que cuanto mayor impacto tenga en el sistema, mejor es el ataque. Los ataques se dividen principalmente en dos tipos: los ataques estándares (Standard Attack Model) y los ataques de confusión (Obfuscated Attack Model). Dentro de los estándares se encuentran: Random Attack, Average Attack, Bandwagon Attack y Segment Attack. Y dentro de los de confusión están: Noise Injection y Target Shifting.

A continuación vamos a entrar en detalle explicando en qué consiste cada modelo, más adelante, en la figura 2.1, se hace un resumen de los 5 modelos de ataques estándares.

1. Random Attack

El random attack o ataque aleatorio es el más simple y fácil de implementar. Dependiendo de si se quiere realizar un Push Attack o Nuke Attack se le asigna $I_T = r_{max}$ o $I_T = r_{min}$ como puntuación

al ítem objetivo. Las puntuaciones de I_F se escogen de forma aleatoria entre un rango definido. Este rango conviene escogerlo alrededor de la media satisfaciendo la condición de distribución normal. Este tipo de ataque no requiere I_S e I_N, porque son $\emptyset$ los dos.

Los ataques aleatorios no requieren conocimiento previo del sistema y son fáciles de implementar. En la práctica real, se requiere rellenar una cantidad considerable de ítems para cada perfil de usuario por lo que tiende a aumentar el coste de ejecución. Además, tiene poca efectividad en comparación con los demás modelos. En resumen, la relación coste/beneficio no compensa.

2. Average Attack

El Average Attack o ataque de media es bastante similar al aleatorio en la forma de tratar los I_S, I_T, I_N. La única diferencia con el método anterior está en cómo puntúa los ratings de I_F, en este caso la puntuación ya no es aleatoria sino que es la media de las puntuaciones del sistema. Dicho de otro modo, la puntuación en I_F va a ser siempre la media, siguiendo la distribución normal.

Para realizar el ataque de media es necesario precalcular la medía de las valores de los ítems. La efectividad de este método es mejor que el aleatorio, ya que los perfiles inyectados serán muy parecidos a los usuarios auténticos. Precalcular las medias de los distintos ítems supone un aumento considerable del coste de conocimiento y ejecución. Sin embargo, muchas empresas como Amazon, Netflix o Youtube ya tienen las medias precalculas y visibles para todos, este hecho hace que el ataque de media sea el método con mejor coste/beneficio.

3. Bandwagon Attack

El bandwagon attack o ataque de popularidad se basa también en un modelo aleatorio. Es de sentido común que la mayoría de la gente prefieren visitar los ítems más activos. Por lo que la táctica de este ataque es seleccionar los ítems más populares como I_S y otorgarles la puntuación máxima r_{max} o la puntación mínima r_{min}. Las estrategias para escoger I_F, I_T, I_N son las mismas que en el modelo aleatorio.

Dado que los perfiles generados a partir de este ataque son muy similares a la gran mayoría de los usuarios, tiene una efectividad bastante alta. Este modelo necesita precalcular la popularidad de los ítems, pero supone un coste menor en comparación con el cálculo de las medias del caso anterior. Al igual que en el caso anterior, muchas empresas tienen publicadas en su página cuales son sus artículos más populares. Por lo que el coste de conocimiento es prácticamente nulo y el coste computación es relativamente bajo.

4. Segment Attack

El segment attack o ataque por segmento consiste en agrupar los usuarios por intereses. Tras seleccionar los I_T, asignarle la puntuación máxima si es un Push o mínima en caso contrario, y escoger los ítems más similares como I_S y los menos similares como I_F. Dependiendo del tipo de ataque que se pretenda realizar, en caso de Push Attack, se asigna a I_S la valoración más alta y a I_F la puntuación más baja, en caso de Nuke Attack, se asigna la valoración más alta a I_F y la más baja a I_S.

A partir de la técnica de asignación, podemos ver que la relación entre los ítems es el foco de

atención de este modelo. Por lo consiguiente, tiende a funcionar mejor en los filtrado colaborativo basado en ítem.

5. Sampling Attack

El sampling attack necesita conocimiento previo de la totalidad del sistema a atacar, por lo que no es muy realista. Necesita procesar todos los registros de valoraciones de todos los usuarios, construir los perfiles falsos como una copia de los auténticos. La única modificación que se requiere es modificar el rating de I_T.

Como los perfiles generados son tan realistas como los auténticos, es muy complicado detectar estos tipos de usuarios maliciosos. Sin embargo, debido a que en la práctica es imposible que alguien obtenga la información de la totalidad de un sistema de recomendación, este modelo es meramente teórico, casi imposible de que suceda en el mundo real.

6. Noise Injection

Este modelo pertenece a los ataques de confusión. En comparación con los 5 anteriores es más dificil de detectar ya que es más discreto. La inyección de ruido actúa sobre los ataques estándares, cuando se intenta asignar un valor a I_S o I_F, basta con multiplicarle por un numero aleatorio α que respete la distribución gaussiana de números aleatorios. El factor α afecta al nivel de confusión del modelo.

7. Target Shifting

En los ataques estándares al I_T se le asigna la valoración máxima r_{max} si se trata de un Push Attack o la valoración mínima r_{min} si se trata de un Nuke Attack. La técnica de target shifting actúa también sobre los modelos estándares, y consiste en evitar los ratings extremistas asignando $I_T = r_{max} - 1$ o $I_T = r_{min} - 1$. Este pequeño cambio puede afectar considerablemente la efectividad de muchos detectores.

8. Hybrid Attack

El ataque híbrido consiste en combinar 2 o más ataques para generar un nuevo ataque. Esta combinación puede dificultar la tarea de muchos detectores, ya que la mayoría de ellos están pensados para fines específicos. Esta idea surge de la filosofía de combinar diferentes técnicas con el fin de obtener mejores resultado. Se explicará con más detalle en el capítulo 4

2.4.3. Evaluación de los ataques (coste y efectividad)

Para realizar cualquier ataque, se debe hacer un estudio previo del sistema y luego inyectar los perfiles considerados. Los costes asociados a este proceso son: el coste de conocimiento y el coste de ejecución. El coste de conocimiento es el gasto que supone obtener los datos necesarios para realizar los ataques. La dificultad en obtener los datos varía según el modelo seleccionado y la plataforma que se pretenda atacar. El coste de ejecución consiste en la dimensionalidad ($perfilesxitems$) de los

Attack models	Push	Nuke
Random attack	$I_s = \emptyset$; give I_F random ratings; $i_t = r_{\max}$	$I_s = \emptyset$; give I_F random ratings; $i_t = r_{\min}$
Average attack	$I_s = \emptyset$; the ratings for I_F are distributed around the mean for each item i; $i_t = r_{\max}$	$I_s = \emptyset$; the ratings for I_F are distributed around the mean for each item i; $i_t = r_{\min}$
Segmented attack	I_s are the target item's similar items and $I_s = r_{\max}$; $I_F = r_{\min}$; $i_t = r_{\max}$	I_s are the target item's similar items and $I_s = r_{\min}$; $I_F = r_{\max}$; $i_t = r_{\min}$
Bandwagon attack	I_s are the frequently rated items and $I_s = r_{\max}$; give I_F random ratings; $i_t = r_{\max}$	I_s are the frequently rated items and $I_s = r_{\min}$; give I_F random ratings; $i_t = r_{\min}$
Sampling attack	$I_s = \emptyset$; copy a existing user profile as I_F; $i_t = r_{\max}$	$I_s = \emptyset$; copy a existing user profile as I_F; $i_t = r_{\min}$

Figura 2.2: Resumen de los ataques estándares tanto para Push Attack como para Nuke Attack

perfiles a inyectar y la complejidad algorítmica del modelo seleccionado. Los atacantes tienen como objetivo maximizar el coste/beneficio.

El tamaño de ataque y el tamaño de relleno son dos indicadores importantes de la magnitud del ataque. El tamaño de ataque determina el número total de perfiles inyectados en el sistema, y el tamaño de relleno I_F determina el tamaño del conjunto de ítems a rellenar en el perfil del usuario. En general, cuanto mayor sea el tamaño de ataque y relleno, mayor será la intensidad del ataque y tendrá mayor impacto en el sistema recomendado, pero también mayor será el costo del ataque requerido. Generalmente para evaluar la efectividad de los ataques se utilizan las métricas conocidas como Prediction Shift y Hit Ratio.

La predicción de desplazamiento (*Prediction Shift*) es la variación en la puntuación prevista de un ítem por parte de los usuarios antes y después de un ataque. El desplazamiento promedio de la predicción se toma en cuenta para todos los ítems del sistema y se hace una media ponderada. Cuanto menor sea el desplazamiento promedio de la predicción, mayor será la robustez y estabilidad del sistema recomendado. A su vez, cuanto mayor sea el desplazamiento promedio de la predicción, mayor será el impacto del ataque, lo que indica menos robustez y estabilidad del sistema.

Suponiendo que U e I son los conjuntos de usuarios e ítems en el conjunto de datos del test, respectivamente. Para cada par de usuario e ítem (u,i), la predicción de desplazamiento se puede expresar como $\Delta_{u,i} = p_{u,i}{}' - p_{u,i}$, donde p y p' son las predicciones previas y posteriores al ataque. Un valor positivo significa que el ataque corresponde a un Push Attack y un valor negativo significa que corresponde a un Nuke Attack. La predicción de desplazamiento de un ítem sobre todos los usuarios se calcula con la ecuación 2.5.

$$\Delta_i = \sum_{v \in U} \Delta_{u,i} / |U| \tag{2.5}$$

Suponiendo que hay I items en el sistema, la predicción de desplazamiento del sistema se calcula en base a la ecuación 2.6.

$$\overline{\Delta} = \sum_{i \in I} \Delta_i / |I| \tag{2.6}$$

La predicción de desplazamiento es un buen indicador para determinar si tras los ataques, estos han tenido el efecto deseado. Sin embargo, puede ocurrir el caso de que algún ítem con un valor de predicción alta no aparece en la lista de los ítems recomendados. Para mejorar la eficacia de medición, los investigadores propusieron una nueva métrica capaz de capturar el impacto de un ataque en la lista de recomendación: Hit Ratio. Suponiendo que R_u representa los top-N ítems recomendados para el usuario u, si i aparece en u entonces $H_{u,i} = 1$, en otro caso, $H_{u,i} = 0$. La fórmula para medir la media de los hit ratio de un ítem se expresa de la siguiente manera en la ecuación 2.7.

$$HitRatio_i = \sum_{u \in U} H_{u,i} / |U| \tag{2.7}$$

Suponiendo que hay I ítems en el sistema, la media de los hits ratio del sistema se calcula mediante la ecuación 2.8.

$$\overline{HitRatio} = \sum_{i \in I} HitRatio_i / |I| \tag{2.8}$$

2.5. Detección en los sistemas de recomendación

El algoritmos de detección de ataques pretenden reforzar la solidez y la seguridad de los sistemas de recomendación [6]. Aunque existen los métodos basado en estadísticas, como Neyman-Pearson, propuesto por Hurley [7], los métodos de detección de ataques principalmente se basan en técnicas de aprendizaje automático, el cual puede ser supervisado, semi-supervisado y no supervisado.

2.5.1. Tipos de algoritmos de aprendizaje automatico

Algoritmos de aprendizaje supervisado

En el aprendizaje supervisado, los algoritmos trabajan con los datos etiquetados, intentado encontrar una función que, dadas ciertas variables de entrada, les asigne la etiqueta de salida adecuada. El algoritmo se entrena con los datos etiquetados por el usuario y según estos datos intenta predecir el valor de etiqueta correspondiente.

Los algoritmos supervisados fueron aplicados en el campo de detección de ataques por primera vez por Chirita en el año 2005 [8]. Chirita incorporó las definiciones de las métricas RDMA (Rating Deviation from Mean Agreement) y DegSim (Degree of Similarity with Top Neighbors). Posteriormente Burke incorporó 3 métricas basandose en RDMA [9]: WDMA (Weighted Deviation Mean Agreement),

WDA (Weigth Deviation Agreement) y LengthVar. Finalmente en el año 2011, Cao y Wu unificaron las métricas anteriores, añadiendo el detector de naives bayes y EM (Expectation Maximization) crearon un modelo de aprendizaje semi-supervisado llamado SemiSad [7]. La descripción detallada de las distintas métricas y el modelo SemiSad se encuentra en la sección 2.5.3.

Algoritmos de aprendizaje semi-supervisado

El estudio de los algoritmos semi-supervisados empezó a mediados de los años 90 como una técnica que utiliza tanto los datos etiquetados como los no etiquetados. Este tipo de algoritmo se basa en mejorar la respuesta del modelo usando un proceso de retroalimentación. Hay tres tipos dentro del aprendizaje semi-supervisado: el modo generativo, la inferencia transductiva y el paradigma de co-entrenamiento. El más sencillo usa el modo generativo y utiliza EM (Expectation Maximization) para modelar la estimación de las etiquetas. Un ejemplo de este tipo de implementación se encuentra en la sección 2.5.3.

Algoritmos de aprendizaje no supervisado

El aprendizaje no supervisado se da cuando no se dispone de datos etiquetados para el entrenamiento, es decir, en este tipo de algoritmo se intentar hacer predicción directamente a partir de los datos de entrada sin entrenamiento previo. El objetivo es encontrar algún tipo de organización de los datos que nos ayude a clasificar las clases.

Los pioneros en este campo fueron O'Mahony y su equipo de investigación [10]. Estos intentaron filtrar los usuarios sospechosos a partir de los vecinos. Usan un enfoque de clustering o agrupación en los sistemas de reputación para detectar usuarios maliciosos que tiene como objetivo reducir la popularidad de los ítem target. Este método necesita revisar y agrupar la base de datos periódicamente, comprobar si el punto central del clustering ha sufrido algún cambio, aquellos usuarios que han provocado este cambio se consideran maliciosos.

Posteriormente Mehta y Nejdl han propuesto un método de análisis semántico probabilístico latente (PLSA-SelectUsers) [11]. PLSA es un modelo mixto que calcula una distribución probabilística sobre las comunidades en función de factores latentes y se ha reportado que es robusto a los ataques. Junto a PLSA, se formula también un modelo llamado PCA-SelectUsers. PCA es un modelo de reducción de dimensionalidad lineal, el cual puede ser usado para seleccionar dimensiones totalmente diferentes. En la sección 2.5.4 se va a describir más detalladamente como funciona el algoritmo de PCA-SelectUser.

2.5.2. Bayes Detector

El detector de bayes [6] se basa en la idea de descomposición de matrices, embebido de usuario y el modelo bayesiano.

La **descomposición de matrices** sirve para capturar las relaciones ímplicitas de los usuarios para mapear usuarios e ítems a un espacio de dimensionalidad menor y obtener el factor potencial de los usuarios e ítems. Su función de pérdida se computa como:

$$L = \sum_{u \in m, i \in n} (r_{u,i} - \widehat{r}_{u,i}) + \lambda(\sum_{u} \|p_u\|^2 + \sum_{i} \|q_i\|^2) \tag{2.9}$$

Input: User-item rating matrix R; user labels U.
Output: Labels of users to be recognized
1: Constructing SPPMI matrix M.
2: **for** user u in U **do**
3: **for** user v in U **do**
4: Count the number of items user u and user v both rated.
5: Compute SPPMI.
6: **end for**
7: **end for**
8: **while** not converged **do**
9: Jointly Decompose R and M.
10: Divide different type of users to Spammer set SU and Normal user set NU.
11: Optimize the posterior probability.
12: Update implicit user vectors P and implicit item vectors Q.
13: **end while**
14: Use P to predict user labels

Figura 2.3: Pseudo código de la técnica de bayes

En la ecuación 2.9, p y q son los factores latentes y k se denota como la dimensionalidad de dichos factores, λ es el coeficiente de regulación para evitar overfitting.

El **embebido de usuario** es equivalente al embebido de palabra. En esta técnica, se usa el embebido de usuario para encontrar una potencial interacción de información entre usuarios e integra la información estructural en la expresión del usuario. Una vez confirmado el usuario del centro, podemos considerar a otros usuarios que han valorado objetos similares como contexto de este usuario. Para describir la interacción latente sobre la matriz SPPMI (Shifted Positive Point Mutual Information) se construye calculando el número de usuarios comunes que han valorado los ítems.

$$PMI(u,v) = \log \frac{\#(u,v.) \cdot |D|}{\#u\#v} \qquad SPPMI(u,v) = max\{PMI(u,v) - \log s, 0\} \tag{2.10}$$

En la ecuación 2.10, |D| es el número de usuarios u y v que satisfacen la condición #(u)=$\sum_v$ #(u,v). Y s es la cuenta de las muestras negativas.

En el **modelo bayesiano**, los usuarios son divididos en usuarios normales NU y usuarios spammers SU, dado el conjunto U de usuarios e I de ítems. Por lo que $U = NU \cup SU$. Se puede expresar como:

$$\prod_{i,su,nu} (\theta|su > nu) \infty \prod_{i,su,nu} p(su >_i nu|\theta)p(\theta) \tag{2.11}$$

En la ecuación 2.11 se tiene $su >_i nu$, que indica que para un ítem i, el usuario u se parece más a un spammer. Con la información $p(su >_i nu) = sigmoid(r_{su,i} - r_{nu,i})$ y $r_{u,i} = p_u^T q_i$ y asumiendo que las partes siguen una distribución Gaussiana, podemos reescribir L como **??**EQ:bayes] donde p_{su} son los rasgos implícitos de los spammers y p_{nv} son los de los normales.

$$L = \sum_{i,su,nu} [\ln sigmoid(p_{su}^T q_i - p_{nu}^T q_i) - \lambda(\|p_{su}\|^2 + \|p_{nu}\|^2\{\|q_i\|^2)] \tag{2.12}$$

Tras dividir los tipos de usuario como S o N, vamos a optimizar la probabilidad a posteriorí con la siguiente fórmula. En la cual, $m_{u,v}$ denota la información de interacción entre u y v, g_v es el contexto, y w_u y c_v son los pesos del usuario y del contexto.

$$
\begin{aligned}
L = & \sum_{u,i}(r_{u,x} - p_u^T q_\lambda) + \sum_{u,v}(m_{uv} - p_u^T g_v - w_u - c_v) \\
& - \sum_{i,su,nu} [\ln \delta(p_{su}^T \cdot q_i - p_{nu}^T \cdot q_i)] \\
& + \lambda(\sum_u \|p_u\|^2 + \sum_{su} \|p_{su}\|^2 + \sum_{nu} \|p_{nu}\|^2 + \sum_i \|q_i\|^2 + \sum_v \|g_v\|^2),
\end{aligned} \tag{9}
$$

Las reglas de actualización son:

$$
\begin{aligned}
\frac{\partial L}{\partial p_u} &= \lambda p_u - (r_{u,i} - p_u^T q_i) q_i - (m_{uv} - p_u^T g_v - w_u - c_v) g_v \\
\frac{\partial L}{\partial p_{su}} &= \frac{-q_i}{1 + e^{p_{su}^T q_i - p_{nu}^T q_i}}, \\
\frac{\partial L}{\partial p_{su}} &= \frac{q_i}{1 + e^{p_{su}^T q_i - p_{nu}^T q_i}}, \\
\frac{\partial L}{\partial q_i} &= \lambda q_i - (r_{u,i} - p_u^T q_i) p_u - \frac{p_{su} - p_{nu}}{1 + e^{p_{su}^T q_i - p_{nu}^T q_i}}, \\
\frac{\partial L}{\partial g_v} &= \lambda g_v - (m_{uv} - p_u^T g_v) p_u) \\
\frac{\partial L}{\partial w_u} &= m_{uj} - p_u^T g_v - w_u - c_v, \\
\frac{\partial L}{\partial c_v} &= m_{uj} - p_u^T g_v - w_u - c_v,
\end{aligned} \tag{10}
$$

2.5.3. SemiSAD

La técnica de SemiSAD [7] consiste en utilizar las distintas métricas preprocesadas para entrenar con el modelo de Gaussian Naive Bayes, tras ello se intenta optimizar el clasificador ya entrenado con EM-λ (Expectation Maximization).

UID/Profile	Class S/N	Metric 1 *H(X)*	Metric 2 *ADegSim*	.....	Metric *m* *RDMA*

Figura 2.4: Formato del dato tras el preprocesado con las métricas

Definiciones de las métricas

Definición 1 (Entropía) Suponiendo que $X_u = \{n_i, i = 1, 2, ..., r_max\}$ es el set estadístico de usuario u, donde n_i es la frecuencia de ocurrencia de los i ratings posibles en el perfil u. Entonces la entropía $H(u)$ se puede computar como indica la ecuación 2.13. El rango de la entropía está entre [0,$\log_2 r_{max}$]. El rango mínimo significa que todos los ratings son iguales ($r_{max} = 1$) y el máximo valor $\log_2 r_{max}$

indica que para todo i,j $n_i = n_j$.

$$H(u) = -\sum_{i=1}^{r_{max}} \frac{n_i}{S} \log_2 \frac{n_i}{S} \quad donde \quad S = \sum_{i=1}^{r_{max}} n_i \tag{2.13}$$

Definición 2 (DegSim) Degree of Similarity with Top Neighbors es el promedio de las similitudes de los k vecinos más cercanos de u y se puede computar como:

$$DegSim_u = \frac{\sum_{v=1}^{k} sim_{u,v}}{k} \tag{2.14}$$

Definición 3 (LengthVar) Length Variance fue introducido para medir la diferencia entre la longitud de un perfil dado y la longitud media de todos los perfiles. Suponiendo que #(P_u) como longitud de u y N como el numero total de perfiles. LengthVar se puede computar como:

$$LengthVar_u = \frac{|\#(P_u) - \overline{\#}|}{\sum_{i=1}^{N} (\#(P_u) - \overline{\#})^2} \tag{2.15}$$

Definición 4 (RDMA) Rating Deviation from Mean Agreement es capaz de identificar los ataques a través de una examinación de la desviación media de los ítems. Suponiendo que NR_i como el numero de ratings realizados por todos los usuarios al item i y N_u como el número de ítems valorados por u, $RDMA_u$ se puede computar como:

$$RDMA_u = \frac{\sum_{i=1}^{N_u} \frac{|r_{ui} - \overline{r_i}|}{NR_i}}{N_u} \tag{2.16}$$

Definición 5 (FMTD) Filler Mean Target Difference es efectivo para los modelos de ataques cuyo ratings de target y filler presentan grandes diferencias. Por ejemplo, Bandwagon y Segment. $FMTD_u$ se calcula de la siguiente manera, suponiendo que P_uT es el set de los ítems con mayor rating y P_uF es el conjunto de los otros ítems valorados.

$$FMTD_u = |(\frac{\sum_{i \in P_{uT}} r_{ui}}{P_uT}) - (\frac{\sum_{j \in P_{uF}} r_{uj}}{P_uF})| \tag{2.17}$$

Tras entrenar el clasificador Naive Bayes con las i-métricas pasamos a aplicar EM. Siendo la probabilidad de que un perfil pertenezca a la clase C si la métrica $M_i = x_i$: $P(x_i|C) = g(x_i, \mu_{Ci}, \sigma_{Ci})$, donde $g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. μ y σ son la media y la desviación estándar de la métrica i. A partir de lo anterior, podemos definir ahora que: $P(U|C) = \prod_{i=i}^{n} P(x_{ui}|C)$. En general, el paso M para calcular la probabilidad de un perfil de una clase se define como:

$$P(u_k \in C) = P(c|u_k) = \frac{P(C)P(u_k|C)}{P(u_k)} \tag{2.18}$$

La probabilidad de P(C) representa P(N) y P(S), el cual se toma uniformemente. Y P(u) es tomado como un constante. En el siguiente paso M, vamos a calcular la nueva media y desviación basándonos en las probabilidades calculas en el paso E.

$$\mu_{C_i} = \frac{1}{|C|}\sum_{u=1}^{|C|} w_u X_{ui} \qquad \sigma_{C_i} = \sqrt{\frac{1}{|C|}\sum_{u=1}^{|C|} w_u^2 (X_{ui} - \mu_{C_i})^2} \tag{2.19}$$

En la ecuación 2.19, $|C| = |L_c| + \sum_{u=1}^{U} w_u$ y $|L_c|$ es el número de usuarios etiquetado como c en L. Con $\Delta(u) = \lambda$ si $u \in U$ y $\Delta(u) = 1$ si $u \in L$ w_u se puede interpretar como:

$$w_u = \Delta(u)\frac{P(u \in C)}{\sum_j P(u \in C_j)} \tag{2.20}$$

Algorithm Pseudo-code of the Semi-SAD algorithm.

Input:

L: Labeled user profile set

U: Unlabeled user profile set

Output:

A shilling attack detector, θ that takes an unknown user profile and predicts a class label.

1: Take pre-process on L and U to derive the data format
2: Train an initial naïve Bayes classifier, θ, based on labeled user profile set, L, only.
3: **repeat until** none of estimated parameters (μ_{Ci} and σ_{Ci}) changes
4: (E-step) Utilize the current classifier, θ, to calculate the probability of each user profile belonging to each class
5: (M-step) Improve the current classifier, θ.
6: **end of repeat**
7: Take the classifier, θ, to be the shilling attack detector.
8: Compute the ratio

$$\Omega = ln\frac{P(S|u)}{P(N|u)} = ln\frac{P(S)}{P(N)} + \Sigma_{k=1}^{n} ln\frac{P(x_k|S)}{P(x_k|N}$$

9: **for** all user profiles in U
10: **if**($\Omega > \eta$) the user is Shilling attacker.
11: **else** the user is Normal user.
12: **end of for**

Figura 2.5: Pseudo código del algoritmo de SemiSAD

2.5.4. PCASelectUser

PCA (principal component analysis) es la técnica más simple y mejor (en el sentido de error medio cuadrático) en cuanto a la reducción de dimensionalidad lineal y en [11] se propuso como algoritmo de detección. Hay tantas componentes principales (PCs) como números de variables de la combinación lineal. Si el primer componente principal es $s_1 = x^T w_1$, donde el vector de coeficientes de p-dimensión

es $W_1 = (w_{1,1}, ..., w1_{1,p})^T$ resulta $W_1 = \arg\max_{|w|=1} Var(X^T W)$.

Principalmente, PCA es equivalente a ejecutar un eigen-decomposition (descomposición de autovectores y autovalores) sobre la matriz de covarianza del dato original. Los autovectores constituyen una matriz de proyección, la cual podrá ser usada para transformar los datos básicos en unos formando el componente principal. Formalmente, supone una matriz X^{mxn}, donde cada columna corresponde a $x_i = (x_{i,1}, ..., x_{i,m})$. Para simplificar se asume que el dato está centrado en 0. Ahora vamos a calcular la matriz de co-varianza: $C = (1/(n-1))XX^T$

Mediante el teorema de descomposición espectral tenemos C tal que así: $C = U\lambda U^T$. Donde λ es la matriz diagonal que contiene los autovalores de C. U contiene los correspondiente autovectores. De aquí podemos sacar los PCs de las filas de la matriz S, donde $S = U^T X$ Ordenando las filas de U en orden según los autovalores sacados a partir de λ, obtenemos los PCs en orden ascendente.

Algorithm PCASelectUsers (**D**)

1: $\mathbf{D} \leftarrow$ z-scores($\mathbf{D}$)
2: $\mathbf{D} \leftarrow \mathbf{D}^T$
3: $COV \leftarrow \mathbf{D}^T\mathbf{D}$
4: $\mathbf{U}\lambda\mathbf{U}^T =$ Eigen-value-Decomposition(COV) { Get Eigenvectors of COV }
5: $PCA_1 \leftarrow \mathbf{U}(:, 1)$ {First Eigenvector of COV}
6: $PCA_2 \leftarrow \mathbf{U}(:, 2)$ {Second Eigenvector of COV}
7: $PCA_3 \leftarrow \mathbf{U}(:, 3)$ {Third Eigenvector of COV}
8: **for all** columnid $user$ in **D** **do**
9: $Distance(user) \leftarrow PCA_1(user)^2 + PCA_2(user)^2 + PCA_3(user)^2$
10: **end for**
11: Sort $Distance$

Output: Return r users with smallest $Distance$ values

Figura 2.6: Pseudo código del algoritmo de PCASelectUsers

En la figura 2.6, el z-score de un usuario u para un ítem con media $\overline{v}_u$, desviación σ, la votación $v_{u,y}$ se computa de la siguiente manera: $z_{u,y} = \frac{v_{u,y} - \overline{v}_u}{\sigma_u}$

2.5.5. Métricas utilizadas para evaluar los detectores

El objetivo final del algoritmo de detección de ataques es detectar los perfiles falsos implantados en el sistema original. Las métricas más comunes y las más utilizadas son precisión, recall y F-measure (ecuación 2.21) para evaluar el rendimiento del algoritmo de detección de ataques [10].

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \qquad F = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.21}$$

3

Diseño y desarrollo

En este capítulo vamos a explicar el diseño que se ha seguido para poder implementar el mecanismo de estudio sobre los ataques y detecciones. También se va a indicar el desarrollo de la investigación seguida a lo largo del trabajo, las decisiones tomadas y las dificultades encontradas y cómo se han resuelto.

Como bien sabemos, el propósito de este trabajo es estudiar la aplicación de las distintas técnicas de ataques y detecciones en el dominio turístico. En comparación con los estudios previos sobre sistema de recomendación basándose en los datos de películas (normalmente usando Movielens), nuestro dataset se basa en los datos de checkins de Nueva York. Este tipo de dataset, a diferencia de los anteriores, contiene las marcas temporales de los checkins, así como las coordenadas geográficas de los ítems. Casi todos los métodos tradicionales de ataques y detecciones en sistema de recomendación se basan en los ratings o en su preprocesado. Por lo que la tarea de diseñar un mecanismo de estudio sobre los ataques y detecciones se deberá basar primeramente en ratings, comprobar su correcto funcionamiento y, una vez hecho esto, trasladarlo al dataset de Nueva York para comprobar su comportamiento.

3.1. Diseño

Este trabajo no tiene un carácter productivo, es decir, no se obtiene un producto visible como resultado. Como este trabajo está más orientado a la investigación, el mecanismo implementado es un programa en Python que contiene partes como una especie de librería para poder simular las ataques y detecciones en sistemas reales. Aunque así, en cuanto al diseño de este mecanismo se ha seguido rigurosamente la filosofía de ingeniería del software. Siguiendo los patrones de diseño necesarios, ya sea para mejorar la usabilidad o estabilidad como para mantener una línea de desarrollo posterior.

El diseño del trabajo consta de dos partes, el ataque y la detección. Estas dos clases aunque comparten métodos de procesados, no tienen relación entre ellos dado que uno genera los ataques en formato de base de datos y el otro simplemente lo lee. Dicho de otro modo, no se conocen porque no tienen necesidad de conocerse.

Tanto la implementación realizada para los datos tradicionales sobre ratings (en Movielens) como la implementación para las marcas temporales (en Foursquare) sigue el mismo diseño e implementación general. Por lo que las diagramas de clases en las figuras 3.1 y 3.2 son de propósito general y es válida para ambas implementaciones.

3.1.1. Diagrama de clases

Un diagrama de clases define la relación entre las distintas clases, si tienen herencia o no, si uno está compuesto por otro, si uno pertenece a otro, etc. Como la clase de ataque y la clase de detección ni siquiera conocen la existencia del otro, para simplificar ha sido conveniente dividir el diagrama de clases en dos partes: uno dedicado al ataque y otro a la detección.

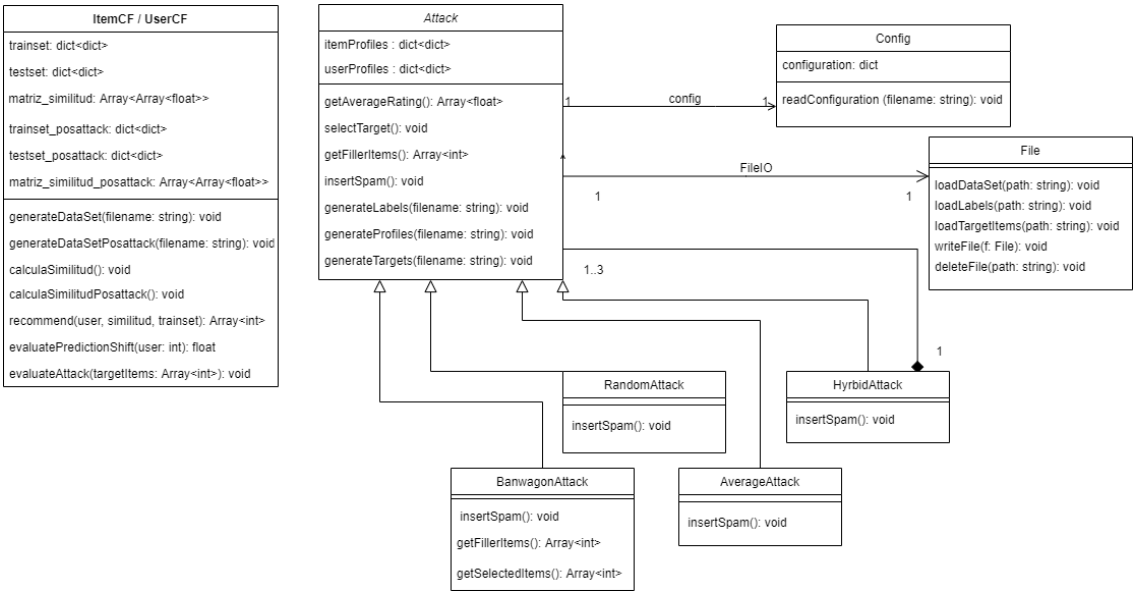


Figura 3.1: Diagrama de clase en ataque

3.1.2. Diagramas de Secuencia

Un diagrama de secuencia muestra la interacción de los objetos en la misma secuencia de tiempo. Con el fin de mostrar cómo se lleva a cabo un ataque y una detección se añaden las siguiente figuras (Figura 3.3 y 3.4). Se ha escogido realizar estos dos diagramas de secuencia con el fin de ilustrar cómo se genera y se evalua un ataque y también cómo detectarlo posteriormente.

3.2. Desarrollo

El propósito principal de este trabajo es estudiar la robustez de los sistemas de recomendación, tanto aquellos que usan ratings como los que usan datos de check-in con marcas temporales. Para poder realizar esta tarea se intenta hacer una comparativa de efectividad de los ataques y detecciones entre los sistemas de recomendación tradicionales y el nuevo sistema de recomendación con marcas temporales en el tiempo.

Durante el desarrollo de este trabajo se han estudiado las técnicas de ataques como RandomAttack, AverageAttack, BandwagonAttack e HybridAttack. Siendo RandomAttack la peor, AverageAttack el medio y BandwagonAttack el mejor. La técnica de HybridAttack es un comodín ya que no siempre funciona de la manera deseada. Más adelante en el siguiente capítulo se va a describir en detalle la

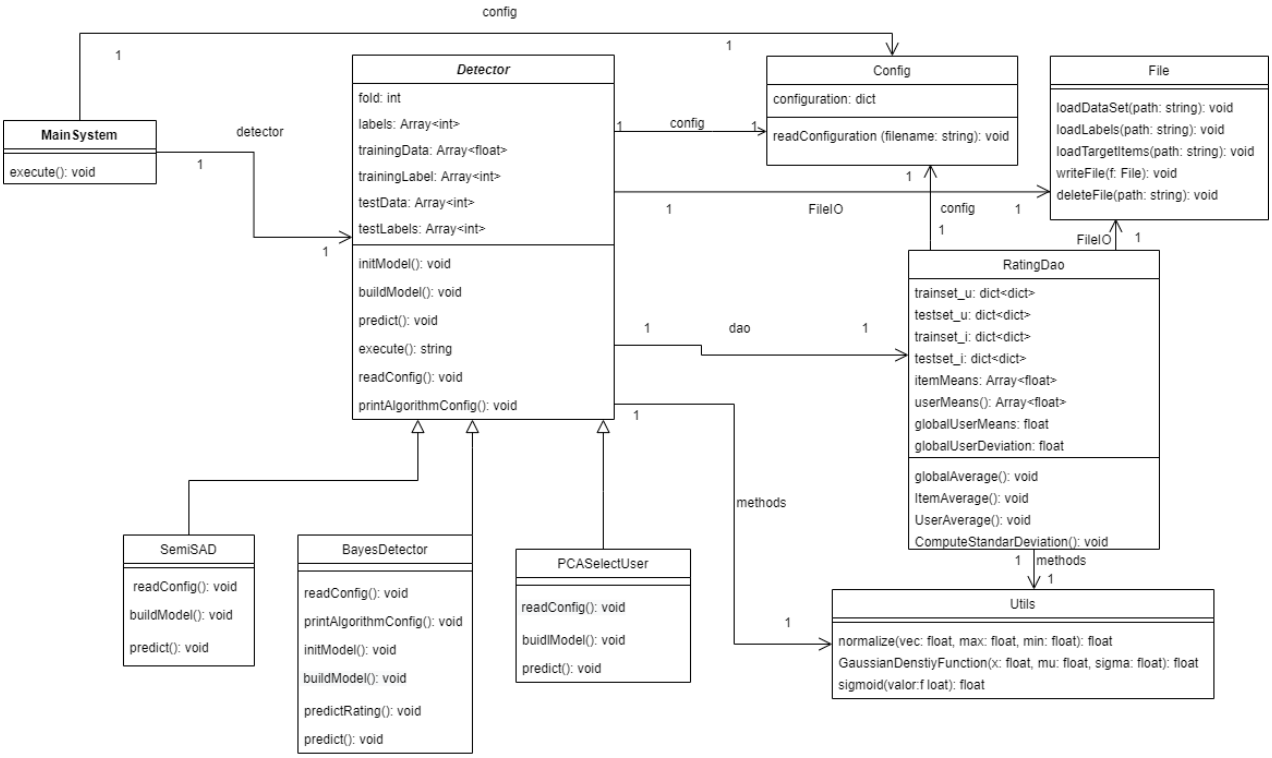


Figura 3.2: Diagrama de clase en detección

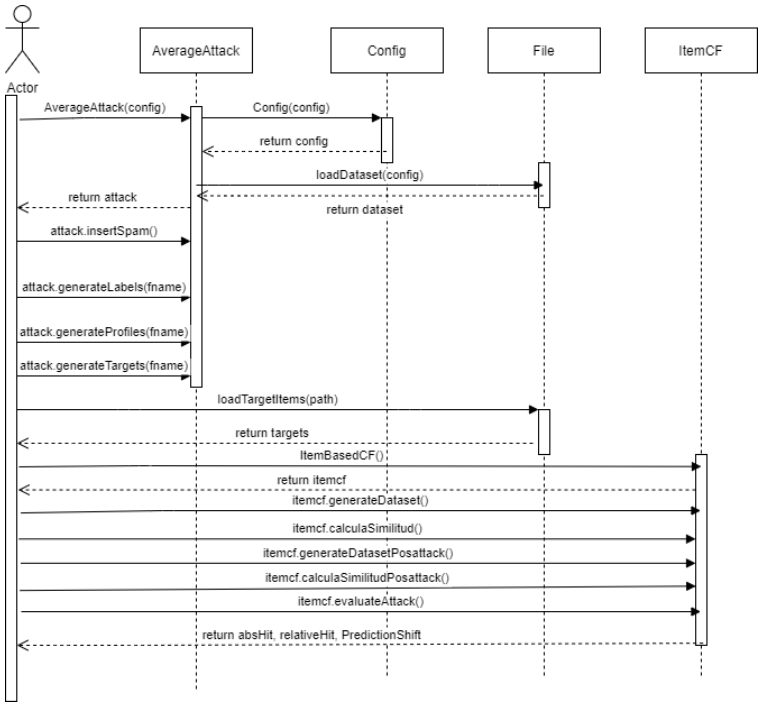


Figura 3.3: Diagrama de secuencia para realizar un ataque de tipo Average y evaluarlo.

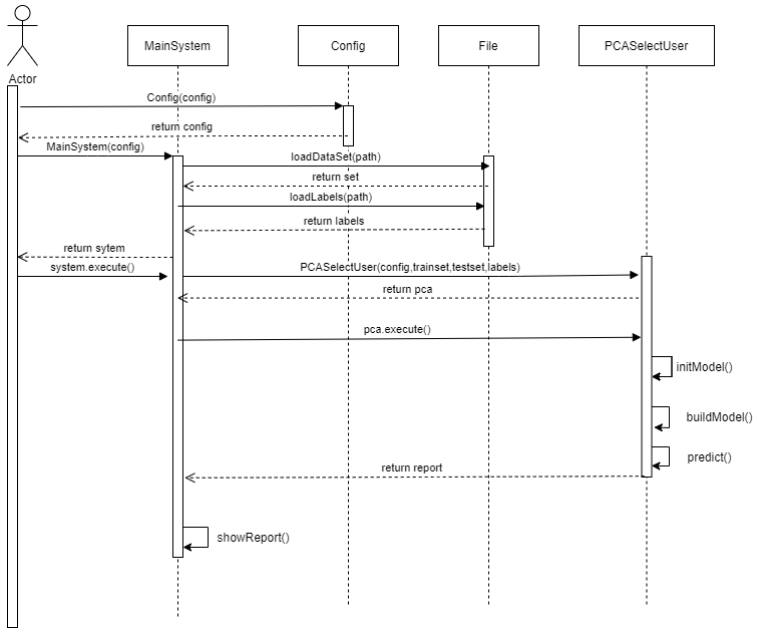


Figura 3.4: Diagrama de secuencia para detectar un ataque mediante PCASelectUser.

eficacia de los 4 ataques propuestos en este trabajo.

Al igual que con los ataques, se han escogido 3 modelos de detección siguiendo la filosofía de hacer una comparativa entre las técnicas de aprendizaje supervisado, semi-supervisado y no supervisado. Estos modelos son: BayesDetector (supervisado), SemiSAD (semi-supervisado) y PCASelectUser (no supervisado).

3.2.1. Pasos de implementación y decisiones de diseño

Los pasos de implementación que se han seguido para llevar a cabo este trabajo son:

Paso 1 Montar el generador de ataque. Como primer paso se ha hecho un estudio riguroso de qué tipo de ataque se va a implementar. Los ataques básicamente se dividen en dos grandes tipos: los estándares y los de confusión. Los de confusión se pueden entender como adiciones extras a los ataques estándares. Finalmente, se han elegido tres adecuados al contexto. Tras implementar estos tres (Random, Average y Bandwagon) surgió la idea de probar una combinación de ellos. De esta manera se comenzó la implementación de HybridAttack, este ataque recibe como parámetro de 0 a N ataques, siendo en nuestro caso N=3, mezclando los spammers.

Paso 2 Implementar el filtrado colaborativo tanto basado en ítem como basado en usuario. La primera versión del filtrado colaborativo contenía los ataques en sí. Pero para facilitar la prueba, se decidió aislar totalmente la clase de ataque con la clases de CF (Collaborative Filtering). De esta forma, el evaluar un ataque no siempre implica volver a lanzar el ataque. También está la ventaja de tener ataques guardados para poder usarlos luego en detección. Se han implementado las métricas de hit ratio absoluto (cuenta 1 cuando alguno de los targets es recomendado), hit ratio relativo (presenta una escala normalizada entre 0 y 1, es 1 cuando los top-K recomendados son todos ítems targets).

Paso 3 Construir los detectores. Este es el paso más decisivo, porque decidir qué detectores imple-

mentar es una tarea bastante complicada. Se tiene que hacer un estudio previo de viabilidad, estimar cómo va a funcionar y ver si encaja con el contexto. Finalmente se han elegido 3 técnicas de detección, uno por cada tipo de aprendizaje automático.

Paso 4 Probar el conjunto de ataques y detectores en movielens. Se hace una prueba de si realmente se ha implementado de forma correcta, puesto que es donde se tienen datos del estado del arte para comparar. Una vez terminado este paso, pasamos a operar con el conjunto de datos de foursquare.

Paso 5 Modificar los ataques y detectores para Foursquare (datos de Nueva York). Dado que la mayoría de estos detectores reciben como input una matriz numérica, se ha tenido que transformar en muchos casos los registros de checkin como frecuencias de visita de la siguiente forma: $rating = frequence/(frequence_{max} - frequance_{min})$ aunque esta transformación se puede hacer de otras muchas formas, como usando el z-score. Estas variantes se dejan para investigar en el futuro.

3.2.2. Otras decisiones de diseño

1. Utilizar el lenguaje Python. Al principio se decidió usar python debido a su librería sklearn, la cual contiene una alta cantidad de métodos de aprendizaje automático. Otro motivo sobre la elección de Python se basa en su estructura de datos. Python tiene potentes diccionarios, arrays de numpy y listas que son muy fáciles de utilizar.

2. Se ha hecho una reducción de datos en los datos de Nueva York. Dada la complejidad del algoritmo, operar con una base de datos tan densa y tan desbalanceada como esta supondría un alto coste. Hay demasiadas coordenadas geográficas (ítems), existen usuarios con más de 20 visitas en un sitio y los demás solamente 1. También existen usuarios que solamente han visitado un único lugar una vez, todo esto no ocurre en los datos de Movielens, ya que vienen filtrados de origen. Por lo tanto, se ha decido ignorar estos tipos de usuario como si estuvieran inactivos. Básicamente, se ha hecho un preprocesado de los datos y se han eliminado usuarios que tuvieran muy pocas visitas y los que las tienen extremadamente altas. El resultado de este procesado es que los usuarios resultantes son usuarios bastante relacionados entre ellos. Las consecuencias de esta decisión se explicarán en el siguiente capítulo.

3.2.3. Dificultades encontradas y su solución correspondiente

A la hora de implementar la técnica de detección SemiSAD, tras emplear las métricas correspondiente, entrenar el Gaussian Naive Bayes (GNB) y aplicar las técnicas de EM-λ, pasamos a calcular la Ω de la figura 2.6. Este Ω debería ser dependiente de usuario u, en caso negativo, sería un valor constante para cualquier usuario del dataset de test. Tal como está definido en el articulo [11], la técnica de SemiSAD genera un Ω constante y predice todo S(Spam User) o todo N(Normal User) para el conjunto test. Después de revisar varias veces el artículo, se entendió que esto era muy extraño, ya que implicaba predecir siempre igual, por lo que debería haber algún tipo de errata u omisión dicho algoritmo.

Para poder seguir usando SemiSAD se ha decido modificar la parte de Expectation Maximization sustituyendo las líneas 7 a 12 de la figura 2.6 por:

```
Repite hasta que len(X) sea mayor que k:
1. Predecir la probabilidad para una parte X de train
2. Obtener los top-k probabilidad según probabilidad a S o N
3. Si P(N) > P(S), hacer un entrenamiento parcial de los datos
   indicando que todo son N añadiendo un peso $\lambda$,
   en caso contrario, no se añade peso pero si se hace un
   entrenamiento parcial indicando que todo son S.
4. Termina la repetición
Una vez que todo está etiquetado, predict con el GNB.
```

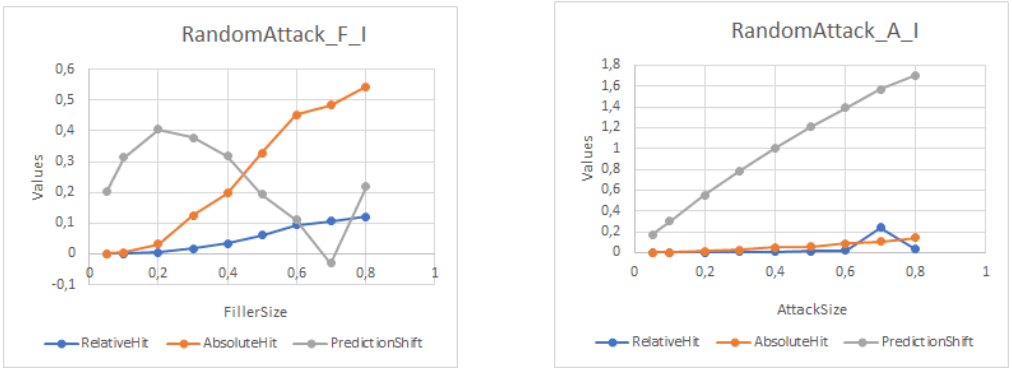
4

Pruebas y resultados

A lo largo de este capítulo se van a mostrar las pruebas realizadas para medir el rendimiento de los distintos métodos de ataque y detección, tanto para datos de Movielens como para datos de Foursquare. Posteriormente, se van a discutir los resultados en base a las pruebas realizadas. Todos los ataques realizados en esta sección son ataques de tipo push, es decir, $I_T = r_{max}$. Para realizar los estudios del factor fillerSize en los ataques propuesto se ha fijado el valor de attackSize=0.1. De manera equivalente, para los estudios del factor attackSize, se ha fijado el valor de fillerSize=0.1.

Mediante la variación de fillerSize y attackSize se intenta observar el grado en que el tamaño del ataque y del relleno influye en el desplazamiento predicho promedio y en la tasa de impacto del sistema original para diferentes modelos de ataque.

4.1. Ataques en Movielens

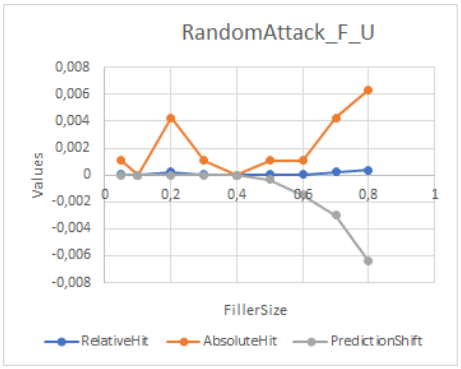


(a) RandomAttack en itemCF

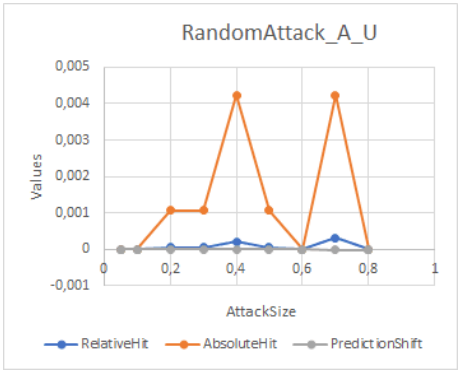
(b) RandomAttack en itemCF

Figura 4.1: Evaluación en itemCF realizada sobre RandomAttack. Variando el tamaño de ataque y de relleno.

Las métricas utilizadas para medir la eficacia son el hit ratio y prediction shift. Dentro del hit ratio se diferencia dos tipos de hit ratio: el hit ratio absoluto y el hit ratio relativo. El hit ratio absoluto es cuando acierta alguno de los targetItems en recomendación. En cambio, el hit relativo varía entre [0,1] siendo 0 ningún acierto en recomendación y 1 significa que se han acertado todos los top-K valores. El hit ratio se computa según la ecuación 2.8 y el prediction shift según 2.6.

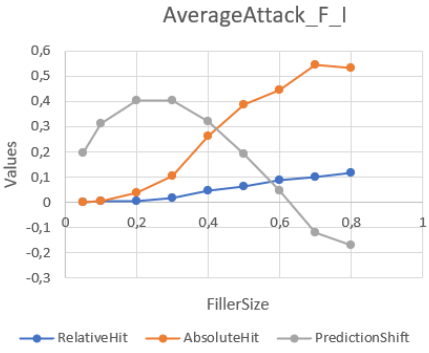


(a) RandomAttack en userCF

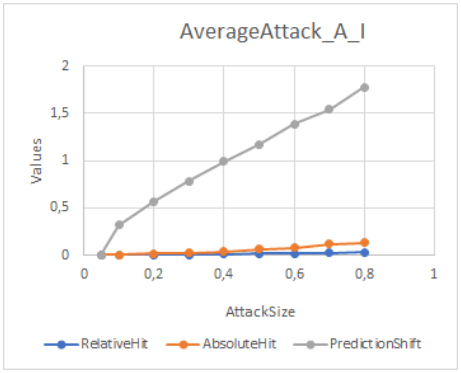


(b) RandomAttack en userCF

Figura 4.2: Evaluación en userCF realizada sobre RandomAttack. Variando el tamaño de ataque y de relleno.

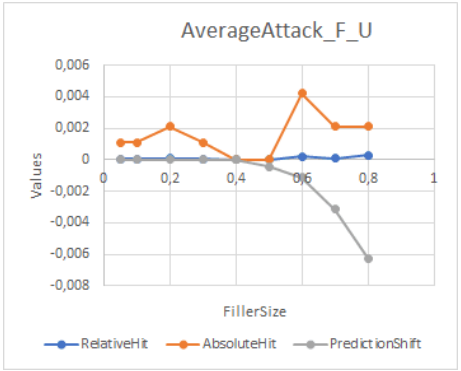


(a) AverageAttack en itemCF

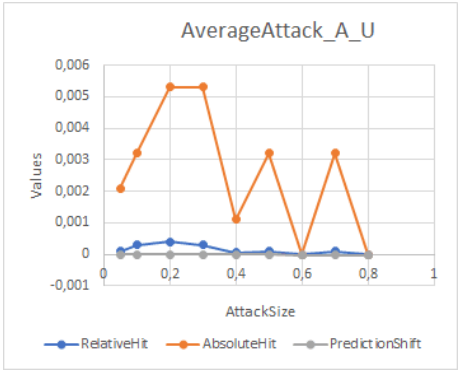


(b) AverageAttack en itemCF

Figura 4.3: Evaluación en itemCF realizada sobre AverageAttack. Variando el tamaño de ataque y de relleno.

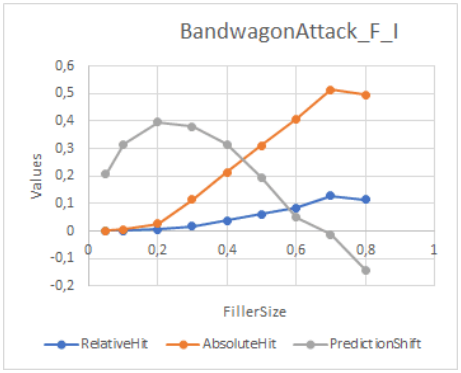


(a) AverageAttack en userCF

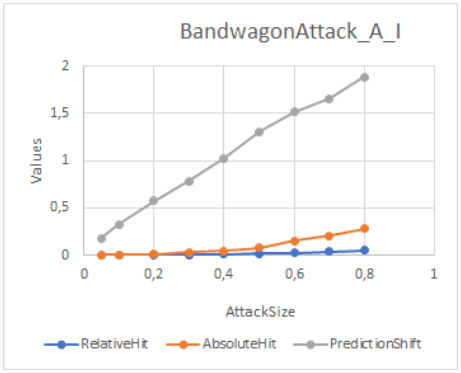


(b) AverageAttack en userCF

Figura 4.4: Evaluación en userCF realizada sobre AverageAttack. Variando el tamaño de ataque y de relleno.

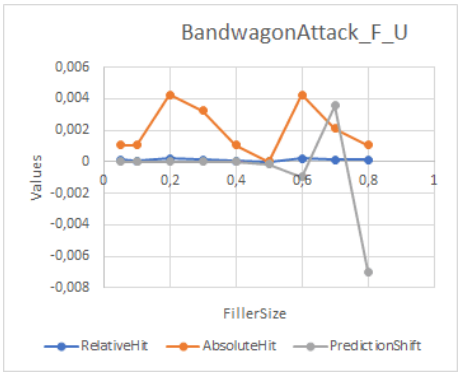


(a) BandwagonAttack en itemCF

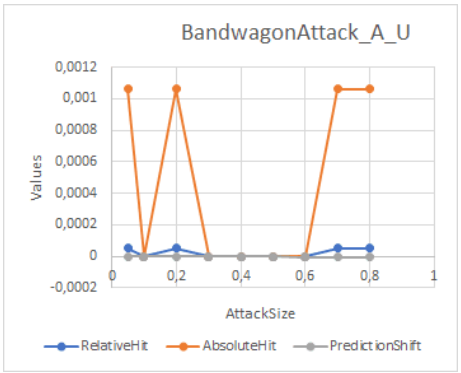


(b) BandwagonAttack en itemCF

Figura 4.5: Evaluación en itemCF realizada sobre BandwagonAttack. Variando el tamaño de ataque y de relleno.



(a) BandwagonAttack en userCF



(b) BandwagonAttack en userCF

Figura 4.6: Evaluación en userCF realizada sobre BandwagonAttack. Variando el tamaño de ataque y de relleno.

Se puede observar que al aumentar fillerSize, aumentan también los hit ratios. Desde otro punto de vista, al aumentar el attackSize se tiende a aumentar el prediction shift pero no los hit ratios. Este hecho nos hace pensar que no es siempre mejor tener un attackSize alto ya que la correlación entre los ítems también es un factor importante. Este es el principal motivo por el cual al aumentar el fillerSize aumente también los hit ratios.

En general, todas las evaluaciones realizadas sobre el filtrado colaborativo de usuario presenta variaciones bastante grandes. Esto se debe a que las técnicas de ataques implementadas se centran principalmente en tratar de ponderar los ratings de los ítems. Por lo cual, tiende a tener un mejor resultado en las mediciones realizadas en filtrado colaborativo basado en ítem.

Otra observación que cabe destacar aquí es que en BandwagonAttack_F_i a medida que se va aumentando el tamaño de relleno, los hit ratios empiezan a escalar progresivamente y disminuye el prediction shift del sistema. Esto se debe a que bandwagon tiene $I_F = random$, por lo que al aumentar fillerSize, mete mucho ruido provocando la bajada del prediction shift.

A través de este estudio empírico, se deduce que el bandwagon es la mejor técnica entre las tres implementadas. Tras estas observaciones, también llegamos a la conclusión de que el fillerSize y attackSize son dos valores relativos, se deben tomar en proporción a lo necesitado. Pero ante un filtrado colaborativo, el fillerSize parece un factor más importante que el attackSize.

4.1.1. Discusión sobre HybridAttack

%Fillersize	R-Hit	A-Hit	PredShift
0.05	0	0	0.12
0.1	0	0	0.2
0.2	0	0	0.31
0.3	0	0	0.37
0.4	0	0	0.40
0.5	0	0	0.40
0.6	0	0	0.39
0.7	0.001	0.002	0.35
0.8	0.005	0.101	0.32

(a) Ataque realizado en itemCF variando fillerSize.

%Fillersize	R-Hit	A-Hit	PredShift
0.05	0	0	0
0.1	0	0	-0.000001
0.2	0	0	-0.000001
0.3	0	0	-0.000001
0.4	0.0005	0.001	-0.000001
0.5	0.0005	0.001	-0.000001
0.6	0	0	-0.000001
0.7	0	0	-0.000001
0.8	0.00015	0.00.	-0.000001

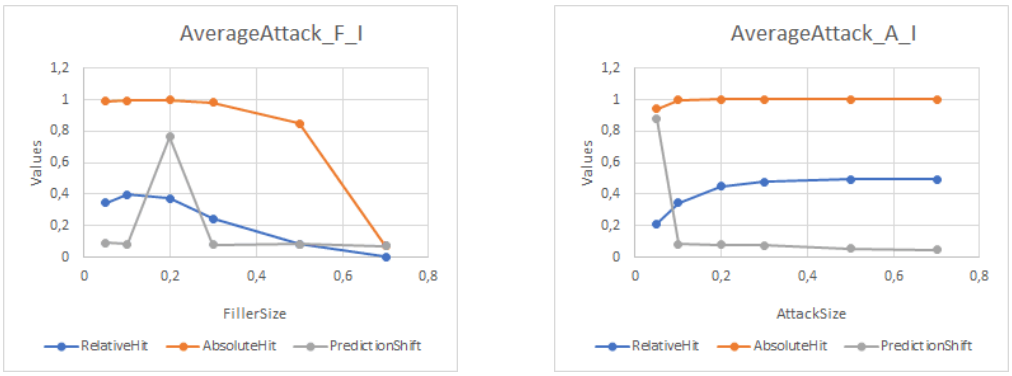
(b) Ataque realizado en userCF variando fillerSize.

Tabla 4.1: Datos de evaluación sobre ataque hybrido.

Este ataque híbrido surge de la combinación de la técnica Random y Bandwagon. Se ha querido combinar estas dos técnicas porque se pretendía hacer una comparativa con el AverageAttack, combinando la supuesta técnica peor y mejor. Sin embargo, tras realizar el ataque vemos que se comporta de manera pésima. No ha acertado ninguna de las veces ni tampoco ha provocado ningún desplazamiento de predicción en el sistema. En este caso, no se muestran los datos con attackSize porque son muy parecidos. En teoría, el hit ratio cuando tenemos un fillerSize alto debería subir también. Si con un fillerSize tan alto como el 80 % no ha conseguido acertar será casi imposible tener un hit ratio subiendo el attackSize.

4.2. Ataque en Foursquare

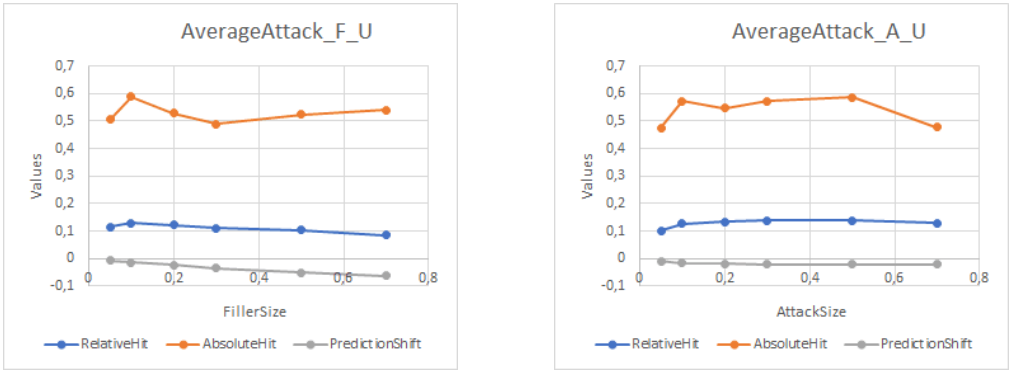
En esta parte se va a hablar de los ataques realizados en los datos de Foursquare. Los ataques realizados han sido: AverageAttack y Bandwagon Attack. Se ha descartado el RandomAttack porque básicamente es una implementación muy parecida al AverageAttack y funciona peor que el segundo. Debido al preprocesado de los datos de NuevaYork mencionado en el capítulo anterior, los datos de Nueva York están bastante correlacionados entre sí, es decir, tienen una similitud alta.



(a) AverageAttack en itemCF

(b) AverageAttack en itemCF

Figura 4.7: Evaluación en itemCF realizada sobre AverageAttack. Variando el tamaño de ataque y de relleno.

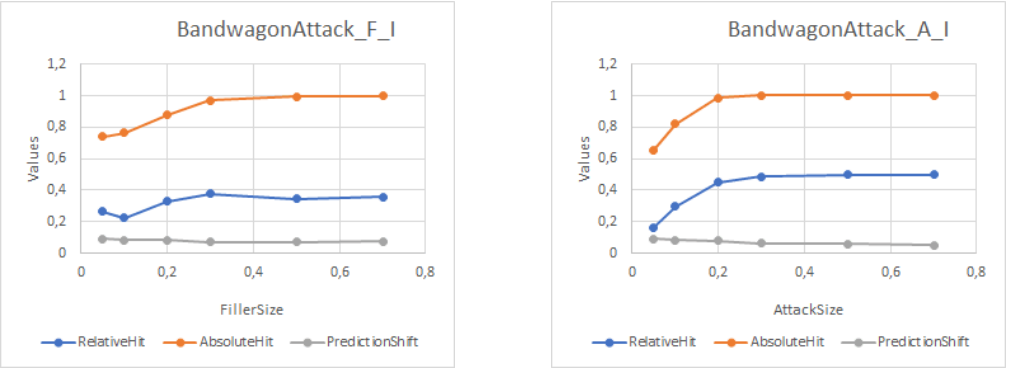


(a) AverageAttack en userCF

(b) AverageAttack en userCF

Figura 4.8: Evaluación en userCF realizada sobre AverageAttack. Variando el tamaño de ataque y de relleno.

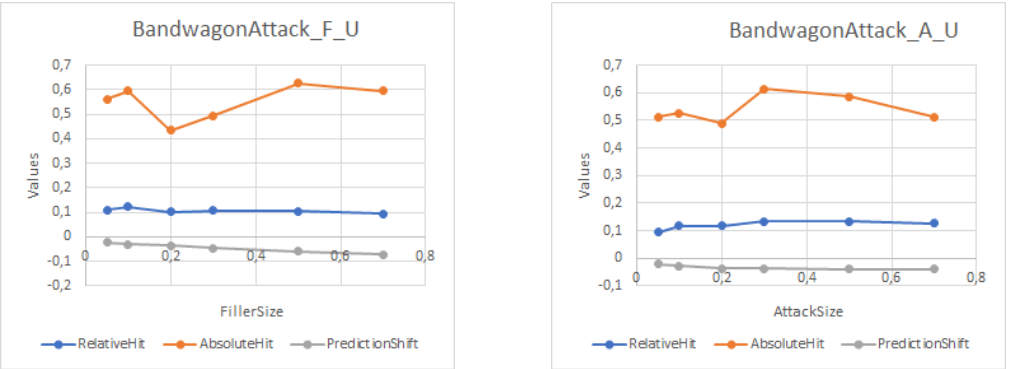
A partir de los datos podemos observar que el prediction shift en esto caso es prácticamente despreciable y apena sufre variación tras aumentar la magnitud de los ataques. Según las gráficas, AverageAttack_A_I y Bandwagon_A_I son los mejores y presentan la misma tendencia de crecimiento. Esto se debe a que uno usa la media para evaluar, cuando la frecuencia de la mayoría de los usuarios en el sistema es alta, la media también es alta. En cambio, Bandwagon cuenta los destinos más populares y a partir de estos destinos intenta establecer una mayor similitud con los demás usuarios. En este tipo de sistema, donde los perfiles son muy similares, es decir, con vector de similitud alto, el tamaño de ataque es un factor más importante que el fillerSize, ya que da igual dónde rellenes debido a que hay



(a) BandwagonAttack en itemCF

(b) BandwagonAttack en itemCF

Figura 4.9: Evaluación en itemCF realizada sobre BandwagonAttack. Variando el tamaño de ataque y de relleno.



(a) BandwagonAttack en userCF

(b) BandwagonAttack en userCF

Figura 4.10: Evaluación en userCF realizada sobre BandwagonAttack. Variando el tamaño de ataque y de relleno.

una gran posibilidad de valorar un ítem popular del sistema.

4.3. Detección en Movielens

Métrica	Modelo	0.05	0.1	0.2	0.3	0.5	0.7
Precision	BayesD	0.8341	0.8486	0.8456	0.8547	0.8245	0.8760
	PCASelectUser	0.8994	0.9053	0.9059	0.8297	0.8452	0.7
	SemiSAD	0.9952	1	0.9056	0.9181	0.9169	0.9130
Recall	Bayes	1	0.9521	1	0.9875	1	1
	PCASelectUser	0.9444	0.8970	0.4319	0.5609	0.9578	0.2318
	SemiSAD	0.7075	0.7382	1	1	1	1
F-measure	BayesD	0.9526	0.8944	0.9038	0.9232	0.9243	0.9530
	PCASelectUser	0.9213	0.9011	0.5923	0.6693	0.8980	0.3490
	SEMISAD	0.8270	0.8494	0.9449	0.9573	0.9567	0.9545

Tabla 4.2: En esta tabla se recogen los resultados de las detecciones realizadas. Donde los números = {0.05, 0.1, 0.2, 0.3, 0.5, 0.7} son % de fillersize

Para medir las distintas técnicas de detección se han utilizado las métricas enunciadas en la ecuación 2.21. El tipo de ataque seleccionado en este caso es el AverageAttack. A partir del estudio y el análisis de los ataques, sabemos que en los datos de Movielens la magnitud de fillerSize es un factor importante en la medición del hit ratio. Este es el motivo por el cual se ha decido estudiar la precisión, recall y f-measure de los tres modelos en función de fillerSize.

A partir de los datos obtenidos en la fase experimental, tenemos claramente un ganador (el SemiSAD ligeramente modificado, con la nueva filosofía descrita en la sección 3.2.3) en todos los aspectos, tanto en estabilidad como efectividad. La única desventaja que se observa es que para los fillerSize pequeños, el recall es bajo. Eso significa que hay elementos relevantes no devueltos. En muchos casos, P y R son contradictorios, una forma de considerar ambos factores es a través de F-measure. Si nos fijamos en F-measure vemos que a partir de fillerSize >0.2, el detector SemiSAD ya le saca ventaja frente a los demás.

4.4. Detección en Foursquare

Tras aplicar las técnicas de detección en los datos de Foursquare, observamos una vez más que el mejor método es SemiSAD modificado. Tanto P, R como F-measure son óptimos frente a los demás detectores. Este hecho a lo mejor se debe a que SemiSAD surge de la combinación de varias métricas (FMTD, LengthVar, DegSim, RDMA, H) y la técnica de EM (Expectation maximization).

Métrica	Modelo	0.05	0.1	0.2	0.3	0.5	0.7
Precision	BayesD	0.8766	0.8199	0.8345	0.9028	0.9110	0.9428
	PCASelectUser	0.9093	0.9973	0.9969	0.9982	0.9926	0.6803
	SemiSAD	0.9932	0.9833	0.9167	0.9146	0.9519	0.9130
Recall	Bayes	0.9521	0.9531	1	0.9624	0.9768	0.98
	PCASelectUser	0.9459	0.9851	0.8765	0.7729	0.5452	0.2259
	SemiSAD	0.9864	0.9866	0.9851	0.9967	1	1
F-measure	BayesD	0.9137	0.8964	0.8933	0.9040	0.9522	0.9623
	PCASelectUser	0.9273	0.9911	0.9328	0.8712	0.7038	0.3391
	SEMISAD	0.9898	0.9850	0.9449	0.9539	0.9754	0.9545

Tabla 4.3: En esta tabla se recogen los resultados de las detecciones realizadas. Donde los números = {0.05, 0.1, 0.2, 0.3, 0.5, 0.7} son % de fillersize

5 Conclusión

5.1. Conclusión general

Con el uso generalizado del comercio electrónico, en internet se puede encontrar todo tipo de información que la gente necesita. En muchos casos, la gente no sabe qué hacer con la información masiva de tal magnitud que se encuentra. Para solventar este tipo de problema, nacieron las técnicas de recomendación. Estos sistemas son capaces de capturar rápidamente la información de interés de los usuarios de acuerdo a sus preferencias específicas y realizar una recomendación personalizada a dichos usuarios.

En los últimos años, los sistemas de recomendación han ganado popularidad rápidamente y cada vez más se están aplicando a diferentes dominios. Sin embargo, debido a la apertura inherente del sistema de recomendación a los usuarios, es vulnerable a los ataques externos con fines maliciosos. Por lo tanto, cómo garantizar la calidad de recomendación y mejorar la seguridad del sistema de recomendación se ha convertido en una llamada de atención por parte de la comunidad científica.

La cuestión de la inyección de perfiles de usuarios maliciosos en sistemas de recomendación es, o debería ser, una línea de investigación prioritaria. Este documento se centra en estudiar los riesgos del sistema de filtrado colaborativo, con su respectivos perfiles de usuarios, ataques y modelos de detección. Se ha intentado analizar el funcionamiento de los algoritmos de detección de ataques clásicos en el dominio turístico.

Los principales resultados de este trabajo son los siguientes:

1. Estudio de los sistemas de recomendación con filtrado colaborativo y la implementación de dichos sistemas.

2. Implementación de tres modelos de ataque comunes (randomAttack, averageAttack, bandwagonAttack), junto con una propuesta sobre un algoritmo de ataque híbrido inspirado en la técnica de RandomForest. En base a los 4 modelos de ataque anteriores se construyó un mecanismo de generador de usuarios falsificados en el sistema de recomendación de filtrado colaborativo. Posteriormente se intentaron analizar los efectos de los ataques externos en el sistema de recomendación, evaluando la efectividad correspondiente.

3. Implementación de tres algoritmos de detección de ataque: SemiSAD, BayesDetector y PCASelectUsers. Durante la implementación del algoritmo SemiSAD, se ha hecho una pequeña modificación en la parte de EM-λ debido a que las definiciones de los pasos no estaban claras. Tras modificarla se

ha comprobado la eficacia de la nueva implementación y resulta que es más eficiente que los otros dos algoritmos.

4. Aplicación tanto de los ataques como de los modelos de detección en los nuevos datos de dominio turístico.

5.2. Tareas futuras

Durante este proyecto, nos hemos basado en los trabajos de investigación de otros célebres científicos para poder realizar este trabajo de investigación sobre ataques y detecciones aplicados al dominio turístico. En los estudios próximos relativos a mi trabajo se puede contemplar la posibilidad de:

1. Implementar un mecanismo de ataque híbrido que ajuste correctamente las proporcionalidades. Por ejemplo, un 10 % de ataque segment + 90 % de ataque bandwagon.

2. Continuar este estudio en el dominio turístico, e intentar diseñar un método de detección que incluya las coordenadas. Para ello se recomienda usar la distancia de Haversine.

3. Idear una técnica que preprocese los datos de Foursquare, tal como lo hace SemiSAD, para conseguir las mejoras demostradas con esta técnica.

Bibliografía

[1] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Introduction and challenges," in *Recommender Systems Handbook* (F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 1–34, Springer, 2015.

[2] D. Ciruelo, "Introducción a sistemas de recomendación." Accedido en Mayo 2020: `https://blog.bi-geek.com/introduccion-a-sistemas-de-recomendacion/`, 2020.

[3] M. de Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro, "Semantics-aware content-based recommender systems," in *Recommender Systems Handbook* (F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 119–159, Springer, 2015.

[4] D. Goldberg, D. A. Nichols, B. M. Oki, and D. B. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[5] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001* (V. Y. Shen, N. Saito, M. R. Lyu, and M. E. Zurko, eds.), pp. 285–295, ACM, 2001.

[6] F. Yang, M. Gao, J. Yu, Y. Song, and X. Wang, "Detection of shilling attack based on bayesian model and user embedding," in *IEEE 30th International Conference on Tools with Artificial Intelligence, ICTAI 2018, 5-7 November 2018, Volos, Greece* (L. H. Tsoukalas, É. Grégoire, and M. Alamaniotis, eds.), pp. 639–646, IEEE, 2018.

[7] J. Cao, Z. Wu, B. Mao, and Y. Zhang, "Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system," *World Wide Web*, vol. 16, no. 5-6, pp. 729–748, 2013.

[8] P. Chirita, W. Nejdl, and C. Zamfir, "Preventing shilling attacks in online recommender systems," in *Seventh ACM International Workshop on Web Information and Data Management (WIDM 2005), Bremen, Germany, November 4, 2005* (A. Bonifati and D. Lee, eds.), pp. 67–74, ACM, 2005.

[9] R. D. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Classification features for attack detection in collaborative recommender systems," in *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006* (T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, eds.), pp. 542–547, ACM, 2006.

[10] R. Burke, M. P. O'Mahony, and N. J. Hurley, "Robust collaborative recommendation," in *Recommender Systems Handbook* (F. Ricci, L. Rokach, and B. Shapira, eds.), pp. 961–995, Springer, 2015.

[11] B. Mehta and W. Nejdl, "Unsupervised strategies for shilling detection and robust collaborative filtering," *User Model. User Adapt. Interact.*, vol. 19, no. 1-2, pp. 65–97, 2009.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID