Máster en Ingeniería Computacional y Matemática

Trabajo Final de Máster

Tecnologías que mejoran la privacidad en los sistemas de recomendación.

Estudiante: Francisco Gil Mayo

Tutor: Javier Parra-Arnau.

Profesor responsable asignatura: Juan Alberto Rodríguez Velázquez.

Junio 2017.

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Tecnologías que mejoran la privacidad en los sistemas de recomendación.*
Nombre del autor:	*Francisco Gil Mayo*
Nombre del consultor/a:	*Javier Parra-Arnau*
Nombre del PRA:	*Juan Alberto Rodríguez Velázquez*
Fecha de entrega (mm/aaaa):	06/2017
Titulación::	*Máster en Ingeniería Computacional y Matemática*
Área del Trabajo Final:	*MO.525 TFM-Adhoc*
Idioma del trabajo:	*Español*
Palabras clave	*Sistemas de recomendación, privacidad, mecanismos de perturbación.*

Resumen del Trabajo:

El desarrollo de Internet, de la telefonía móvil y la aparición de numerosas aplicaciones están modificando rápidamente nuestra sociedad.

Muchas aplicaciones disponen de sistemas de recomendación personalizada para analizar las preferencias de cada cliente y predecir el interés que éstos tendrán por un determinado ítem.

Aunque las compañías se esfuerzan en proteger los datos de los usuarios con diferentes técnicas: encriptación, firewalls, etc., se ha demostrado que no son medidas totalmente seguras y los usuarios no confían en la protección y el anonimato que los sistemas de recomendación les pueden ofrecer.

Una posible solución para proteger la privacidad de los usuarios en los sistemas de recomendación son los métodos perturbativos, su objetivo es modificar los ítems y sus respectivas puntuaciones para no ser reconocidos.

Es evidente que estos cambios afectarán a la calidad de la información con la que trabaja el sistema de recomendación, por lo que debe existir un compromiso entre la privacidad y la precisión en la recomendación. Es aquí donde nuestro trabajo cobra sentido.

Existen dos estrategias habituales: "forgery" donde falsificamos la puntuación de los ítems de manera que el usuario no muestra sus intereses reales y "suppression" donde se elimina la puntuación que el ítem tenía.

El propósito de este trabajo es el de evaluar el impacto de estas dos estrategias en utilidades reales de un sistema de recomendación como pueden ser MAE y RMSE.

Abstract:

The development of Internet, mobiles and the appearance of numerous applications are quickly changing our society.

Many applications have personalized recommendation systems to analyze the preferences of each customer and to predict the interest they will have for a particular item.

Although companies try to protect user data with different techniques: encryption, firewalls, etc., it has been shown that they are not completely safe measures and users do not trust the protection and anonymity that recommendation systems can offer.

A possible solution to protect the privacy of users in the recommendation

systems are the perturbative methods, their objective is to modify the items and their respective scores for not to be recognized.

Those changes will affect the quality of the information and accuracy of recommendation systems, so there must be a compromise between privacy and accuracy in the recommendation.

There are two usual strategies: "forgery" where we change the items rating for user doesn't show his real interests and "suppression" where the items rating is eliminated.

The purpose of this paper is to evaluate the impact of these two strategies on real utilities of a recommendation system such as MAE and RMSE.

INDICE

Introducción:

1.1 Contexto y justificación del trabajo.

En los últimos años la expansión de Internet, de la telefonía móvil y la aparición de numerosas aplicaciones están transformando rápidamente nuestros hábitos de compra y ocio.

Estamos acostumbrados a confiar y adquirir artículos que las aplicaciones web nos recomiendan: compramos libros en Amazon[1] que aparecen en nuestra lista de sugerencias, vemos series en Netflix[2] de las que no hemos oído hablar nunca o descubrimos un nuevo grupo en Spotity[3] que nos recuerda ligeramente a nuestra banda favorita.

Todas estas aplicaciones y muchas más (Youtube, Facebook, LinkedIn, etc.) utilizan los sistemas de recomendación personalizada para analizar las preferencias de cada cliente y predecir el interés que éstos tendrán por un determinado ítem (película, libro, vídeo, cantante, etc.).

Se produce un doble beneficio, por un lado, los usuarios disponen de sugerencias y consejos que mejoran su experiencia de navegación y, por otro, las empresas aumentan sus ingresos, segmentan mejor a sus clientes y obtienen una mayor fidelización.

Pero, ¿cómo funcionan estas aplicaciones para sugerirnos un libro o una película y acertar en su predicción?

La mayoría utilizan un tipo de sistemas de recomendación llamado filtro colaborativo o "Collaborative Filtering" (CF), no es necesario conocer las características del usuario o ítem, solo las opiniones que los usuarios han realizado sobre los artículos. Hay muchas formas de capturar esta información, pero la manera más habitual de obtenerla consiste en que los usuarios puntúen los ítems según su parecer.

A partir de estos datos se crean perfiles de usuario que representan las preferencias e intereses de cada uno de ellos. Estos perfiles se agrupan en matrices con las que el sistema de recomendación, aplicando un algoritmo determinado o una combinación de ellos, generará una predicción de los intereses que el usuario tendrá por aquellos ítems que no ha valorado e indicará al usuario qué artículo o elemento podría ser de su agrado

A pesar de las múltiples ventajas que los sistemas de recomendación aportan también existen algunos inconvenientes, por ejemplo, la necesidad de colaboración hace que muchos usuarios desconfíen del tratamiento que se

[1] https://www.amazon.es/
[2] https://www.netflix.com/
[3] https://www.spotify.com/es/

puede realizar con ellos y les preocupa que sus perfiles puedan revelar información confidencial respecto a temas tan sensibles como la salud, la afiliación política o la religión.

Un caso paradigmático de lo que comentamos es lo que ocurrió con el Premio Netflix[4]. En el año 2006, la compañía organizó un concurso para que los participantes mejoraran la precisión de predicción de sus sistemas de recomendación. Los concursantes trabajaron con un dataset que contenía las calificaciones de un gran número de películas enviadas por medio millón de usuarios de forma anónimos. La competición fue un éxito, la empresa consiguió mejorar sus sistemas de recomendación pero, a cambio, se observó que, enriqueciendo y cruzando la información obtenida con datos externos o provenientes de redes sociales, era posible descubrir datos de uso privado como eran la identidad, la tendencia política o la orientación sexual, Este inconveniente propició que la empresa no convocara una segunda edición del concurso.

Aunque las compañías se esfuerzan en proteger los datos de los usuarios con diferentes técnicas: encriptación, firewalls, etc., se ha demostrado que no son medidas totalmente seguras y los usuarios no confían en la protección y anonimato que los sistemas de recomendación les pueden ofrecer.

Esta desconfianza provoca un cambio en el comportamiento de los usuarios al navegar por la red. De hecho existen estudios[5] que indican que el 24% de los usuarios dan información falsa para proteger su privacidad.

Es evidente que estos cambios afectarán a la calidad de la información con la que trabaja el sistema de recomendación, por lo que debe existir un compromiso entre la privacidad y la precisión en la recomendación. Es aquí donde nuestro trabajo cobra sentido.

De todo el abanico de soluciones que hay para proteger la privacidad en los sistemas de recomendación existe una categoría llamada métodos perturbativos que tienen como objetivo modificar los ítems y sus respectivas puntuaciones. Como el resto de soluciones de privacidad también plantea un compromiso entre la privacidad que obtendremos y el grado de precisión en la recomendación.

En estos métodos existen dos estrategias habituales: "forgery" donde falsificamos la puntuación de los ítems de manera que el usuario no muestra

[4] Joonseok Lee, Minxuang Sun, Guy Lebanon."A Comparative Study of Collaborative Filtering Algorithm",

[5] J. Parra-Arnau, D. Rebollo-Monedero, J. Forné, “Optimal Forgery and Suppression of Ratings for Privacy Enhancement in Recommendation Systems”

sus intereses reales y "suppression" que elimina la nota del artículo o producto y así el usuario se abstiene de calificar el ítem.

Este trabajo se basa en el documento "*Optimal Forgery and Suppression of Ratings for Privacy Enhancement in Recommendation Systems*" [6] que fue el primero en estudiar la combinación de ambas estrategias desde un punto de vista de optimización ingenieril.

En el artículo, se define la privacidad como la divergencia entre el perfil de un usuario y el de la población mientras que la utilidad se mide como el porcentaje de ítems eliminados o falsificados.

Estos porcentajes de ítems eliminados o falsificados son, sin embargo, medidas de utilidad simples, escogidas en beneficio de la tratabilidad matemática, y que pueden no ser representativas de métricas estándares de sistemas de recomendación .

El propósito de este trabajo es el de evaluar el impacto que estas técnicas optimizadas de privacidad tienen en utilidades reales de un sistema de recomendación como pueden ser: MAE (*Mean Absolute Error*), RMSE (*Root Mean Square Error*), etc.

En nuestro análisis de estos mecanismos, también consideraremos el caso en el que estos mecanismos no están optimizados, los llamados random.

1.2 Objetivos del trabajo

Las líneas generales del trabajo son, por un lado, analizar diferentes mecanismos de algoritmos de perturbación aplicando distintas estrategias prácticas para usar "suppression" y "forgery" y, por otro lado, escoger distintos datasets, adaptarlos a las necesidades del proyecto, aplicar en ellos los algoritmos de perturbación y las estrategias anteriores y ver como varía el comportamiento con diferentes sistemas de recomendación.

De forma más concreta, los objetivos particulares que se abordan en este trabajo son los siguientes:

- Aplicar estrategias de "suppression" y "forgery" tanto en mecanismos optimizados y no optimizados.
- Diseñar e implementar métodos de selección de ítems par aplicar las estrategias comentadas en el punto anterior.
- Conocer y aplicar distintos algoritmos de sistemas de recomendación y las métricas más utilizadas.

[6] J. Parra-Arnau, D. Rebollo-Monedero, J. Forné, “Optimal Forgery and Suppression of Ratings for Privacy Enhancement in Recommendation Systems"

- Adaptar el programa PREA[7], que nos permite comparar distintos sistemas de recomendación, a las necesidades del diseño.
- Analizar el comportamiento de dos datasets: movieLens-100k y yahoo a través de diferentes gráficas: variación métricas vs porcentaje ítems perturbados, variación riesgo de privacidad vs pérdida de utilidad, métricas vs tipo de algoritmo de recomendación, etc.

1.3 Enfoque y método utilizado.

El enfoque es totalmente práctico, hemos creado diferentes scripts para implementar los algoritmos de perturbación, sus diferentes estrategias y observar los resultados a través de gráficas.

Partimos de una documentación y unas funciones en MATLAB suministrados por el tutor del proyecto. Además teníamos que trabajar con el toolkit PREA que, aunque está programado en JAVA, dispone de un interfaz en MATLAB por lo que decidimos desde un primer momento trabajar en este lenguaje.

Por último, hemos analizado con el tutor un amplio abanico de sistemas de recomendación, trabajado con diferentes métricas que medían la calidad de la precisión y varios datasets para ir poco a poco descartando aquellos algoritmos, métricas o datasets que no aportaban información relevante o necesitaban un elevado tiempo de computación que superaban los límites del proyecto.

1.4 Productos.

Los productos obtenidos son los ficheros de información generados a partir de los datasets utilizados, los distintos scripts que implementan los algoritmos de perturbación, estrategias prácticas, interfaz con PREA, etc. y las gráficas generadas para analizar los datos:

- **Ficheros de información** llamados "ratings.csv" y "genre.csv" que se obtienen a partir de los datasets "movieLens100k" y "yahoo". Era necesario extraer la información útil y modificarla para adaptarla a nuestro diseño.
- **Scripts,** programados en MATLAB, implementan tanto los algoritmos random como los optimizados, las estrategias prácticas para seleccionar ítems a los que aplicar "suppression" y "forgery", la comunicación con el toolkit PREA , capturar la información y diseñar las gráficas.
- **Gráficas**, cada una de ellas se han guardado en formato FIG. Como hemos comentado anteriormente, se han creado un gran número de figuras que analizan las métricas, el riesgo de privacidad, las estrategias prácticas para cada uno de los sistemas de recomendación elegidos.

[7] http://www.prea.gatech.edu/index.html

1.5 Otros capítulos.

Aparte de la introducción, los otros capítulos importantes que forman esta memoria son:

- **Conceptos básicos**, donde se explican aspectos teóricos y herramientas que necesitamos para desarrollar nuestro trabajo: sistemas de recomendación, filtros colaborativos, PREA, etc.
- **Mecanismos y metodología**, es el capítulo más importante del trabajo, se explican de forma detallada todas las decisiones de diseño usadas, los algoritmos teóricos y prácticos aplicados, estrategias de selección de ítems, etc. En todos los casos hemos usado un pequeño modelo de tres usuarios y cinco ítems, a modo de ejemplo para facilitar la comprensión del diseño.
- **Resultados y conclusiones**, recogemos todas las gráficas realizadas para observar el comportamiento de los distintos sistemas de recomendación, la evolución de la privacidad, estudio de las métricas, etc.

2. Conceptos básicos.

2.1 Sistemas de recomendación basados en filtros colaborativos:

En general los sistemas de recomendación se basan en que los usuarios con actividades o gustos parecidos comparten preferencias en el futuro y su objetivo principal es hacer recomendaciones de productos o servicios a las personas. Existe una gran variedad de algoritmos que difieren en su aproximación analítica y que harán que unos se comporten mejor que otros dependiendo de las características del problema a tratar.

Los hay basados en contenido donde se utilizan las características del usuario o del ítem para hacer las recomendaciones, otros basados en filtros colaborativos que usan las opiniones de los usuarios para recomendar y, por último, los híbridos, que combinan las dos opciones anteriores.

Nosotros estamos interesados en la segunda opción los basados en filtros colaborativos. En ellos, no se conocen las características del usuario o del artículo. Para realizar las recomendaciones utilizan las opiniones que realizan los usuarios sobre los productos. La información se dispone en una matriz de ratings, donde los usuarios son las filas y los ítems las columnas y cada celda contiene una nota que indica el grado de gusto o preferencia como vemos en la siguiente figura.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Figura 1. Matriz de un sistema de recomendación filtro colaborativo.

Suelen ser matrices muy dispersas y que requieren de muchos datos para poder usarse. Podemos clasificarlos en dos subgrupos:

- **Basados en memoria**, estos algoritmos cargan todos los datos en la memoria y hacen predicciones basadas en usuarios o ítems similares al usuario o ítem objetivo. No construyen explícitamente un modelo, en su lugar, utilizan los datos para cada consulta. Es decir, no disponen de tiempo de aprendizaje pero tardan más en realizar las predicciones. En esta categoría utilizaremos el "user-based collaborative filtering" y el "item-based collaborative filtering".
- **Basados en modelo**, en este caso se construye un modelo durante el proceso de aprendizaje a partir del reconocimiento de patrones en el conjunto de training set. A continuación, utilizan el modelo para realizar las predicciones. Las últimas mejoras realizadas en la precisión de la predicción se han conseguido con este tipo de filtros colaborativos, especialmente con los métodos basados en factorización. De esta clase utilizaremos "Regularized SVD", "Non-negative Matrix Factorization" y "Probabilistic Matrix Factorization"

2.2 Sistemas de recomendación utilizados.

Como hemos visto en el apartado anterior los algoritmos de sistemas de recomendación que usaremos de la clase basada en memoria son:

- **Userbased**, donde la puntuación de un ítem se realiza en función de las calificaciones que otros usuarios han realizado en ese ítem. Para ello, se tiene en cuenta como un factor de ponderación la similitud entre usuarios. Es decir, las puntuaciones de los usuarios que tiene un interés similar al del usuario objetivo tienen una mayor influencia para realizar la estimación. Es un algoritmo muy utilizado, rápido y logra una precisión razonable.
- **Itembased**, primero analiza la matriz de puntuaciones para identificar relaciones entre los ítems. A continuación, utiliza estas relaciones para

predecir las puntuaciones de los ítems no vistos. Puede trabajar con datos más grandes que userbased, y logra una mejor precisión.

Respecto a los basados en modelo utilizaremos tres tipos basados en la factorización que consiste en dividir la matriz ratings en dos matrices más sencillas: la de perfil de usuario y la de perfil de los ítems. Los tipos son:

- **Regularized SVD**, donde SVD significa "Singular Value Descomposition", este algortimo minimiza el error cuadrático entre las puntuaciones y las estimaciones previstas para ello utiliza el cálculo del gradiente. Para reducir el overfitting añade términos de regularización tanto para los usuarios como para los ítems. Es un algoritmo que consigue una buena predicción.
- **Non-negative Matrix Factorization (NMF**), divide como el resto la matriz de ratings en dos matrices: la de perfil de usuario y la de perfil de ítems, pero añade una restricción más, los valores de estas matrices deben ser positivos. Utiliza la divergencia Kullback-Leibler para minimizar la distancia euclídea entre puntuaciones reales y estimaciones.
- **Probabilistic Matrix Factorization (PMF**) adopta un modelo lineal probabilístico con ruido de observación gaussiano para representar características latentes tanto para usuarios como para ítems.

2.3. Toolkit PREA.

PREA (Personalized Recommendation Algorithms Toolkit) es un software open source programado en JAVA que permite realizar una fácil comparación entre sistemas de recomendación basados en filtros colaborativos.

Permite trabajar con una gran variedad de algoritmos de recomendación, desde básicos como: constant, user average, item average, etc. como basados en memoria: userbased, itembased pasando por los basados en factorización: regsvd, NMF, PMF, BPMF, NLPMF, etc.

Permite que los usuarios utilicen una gran cantidad de métricas para evaluarlos: RMSE, MAE; NMAE, ASYMM, HLU, NDCGG, Kendall, Spearman, etc.

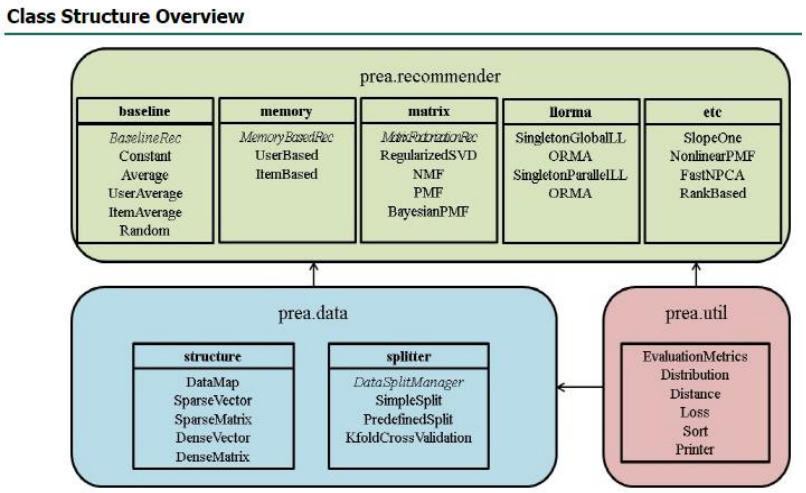


Figura 2. Estructura y clases de PREA.

Se ejecuta con la siguiente orden:

"java prea/main/Prea -f [*fichero.arff]* -a [*algoritm*o]"

donde fichero.arff es un archivo en formato arff que contiene las puntuaciones realizadas por los usuarios y algoritmo es el tipo de sistema de recomendación que queremos evaluar.

Además permite que le indiquemos el porcentaje de datos que serán de training set y los que pertenecerán a test set. En nuestro caso vamos un poco más allá ya que debemos decirle incluso qué usuarios y qué items en concreto pertenecen a cada conjunto. Lo haremos creando un fichero de texto con los datos que corresponden al test set añadiendo la opción "-s pred [*nombre fichero*]'.

Por último, la aplicación dispone de dos scripts en MATLAB (prea y matlab2arff) que permiten llamar al código JAVA de PREA desde MATLAB y convertir los ficheros de puntuación en formato arff respectivamente.

2.4.Métricas utilizadas:

Los scripts están realizados de forma que podemos disponer de todas las métricas que calcula PREA pero nos hemos decantado por mostrar solamente las más habituales y conocidas (MAE y RMSE) ya que nos permiten visualizar claramente cómo se comporta el sistema de recomendación al aumentar las estrategias de perturbación: "suppression" y "forgery":

- **MAE** (Mean Absolute Error) mide la diferencia entre la predicción (p) y el valor real (r) de la siguiente forma:

$$MAE = \frac{1}{n}\sum_{i,j}|p_{i,j} - r_{i,j}|$$

- **RMSE**(Root of the Mean Square Error), mide la diferencia entre la predicción (p) y el valor real (r) como en el caso anterior pero elevando al cuadrado los factores antes de realizar la suma y finalmente, calculando la raíz cuadrada.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i,j}(p_{i,j} - r_{i,j})^2}$$

2.5 Formato ARFF.

PREA utiliza como entrada un fichero que contiene las puntuaciones en formato ARFF (Attribute-Relation File Format). Es el formato de archivo típico utilizado por Weka[8], un toolkit muy reconocido de machine learning. Es muy útil cuando tenemos matrices dispersas.

El formato comienza con una declaración de relación:

@RELATION [relation name]

A continuación tenemos dos secciones: atributos y datos. En atributos definimos los nombres de los campos o columnas y en datos introducimos los valores de la matriz distintos de cero.

Así si tenemos la matriz:

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 3 & 0 \end{pmatrix}$$

En la parte de atributos declararemos el nombre de las dos columnas y el tipo de dato:

@ATTRIBUTE title1 NUMERIC

@ATTRIBUTE title2 NUMERIC

Y en la parte de datos ponemos para cada fila el ID del ítem y la puntuación, si la puntuación es cero es que no hay valor:

@DATA

{2 1}

{1 1, 2 1}

[8] http://www.cs.waikato.ac.nz/ml/weka/

{1 3}

3.Mecanismos y metodología.

3.1 Algoritmos teóricos y prácticos.

Hemos planteado que los algoritmos de perturbación estén divididos en dos partes para facilitar su implementación: los algoritmos teóricos y los algoritmos prácticos.

Los teóricos se encargan principalmente de comparar los perfiles de cada usuario con el perfil de la población. A partir de la divergencia que hay entre estos perfiles calculamos la privacidad y según el nivel de perturbación que introduzcamos (σ, para "suppression", o ρ, para "forgery") tendremos un array o tupla por usuario que nos indicará qué porcentaje de ítems se deben eliminar o falsificar por categoría para tener esa privacidad.

Hemos implementado cinco tipos de algoritmos teóricos, dos que funcionan de forma no optimizada y tres optimizados:

- "**Random Suppression**" es un algoritmo no optimizado que solo utiliza la estrategia de "suppression". Por lo que ρ es siempre cero y σ hemos decidido que trabaje en el intervalo [0,1] con incrementos de 0.1.
- "**Random Forgery**", es otro algoritmo no optimizado pero que ahora trabaja solamente con la estrategia de "forgery", es decir σ=0 mientras que ρ trabajará en el intervalo [0,1] con incrementos de 0.1.
- **"Optimized Suppression**", es un algoritmo optimizado que solo trabaja con la estrategia "suppression". El intervalo de trabajo de σ es [0,1] con incrementos de 0.1 y ρ=0.
- **"Optimized Forgery**", otro algoritmo optimizado pero que trabaja solamente con la estrategia "forgery". El intervalo de trabajo es ρ=[0,1] mientras que σ=0.
- **"Optimized Forgery Suppression**", también es un algoritmo optimizado pero donde ahora se unen las dos estrategias "suppression" y "forgery", Los intervalos de trabajo son ρ=[0,1] y σ=[0,1].

En la siguiente tabla tenemos una clasificación de los distintos algoritmos teóricos (TA) y prácticos(PA).

Algoritmos de Perturbación			
Algoritmos Teóricos (TA)		Algoritmos Prácticos (PA)	
No optimizados	Optimizados	Suppression	Forgery
Random Suppression	Optimized Suppression	Random	Random
Random Forgery	Optimized Forgery	Least Popular	Most Popular
	Optimized Forgery Suppression	Least voted	Most voted

Figura 3.Algoritmos teóricos y prácticos.

Los algoritmos prácticos (PA) recogen la información que reciben de los algoritmos teóricos (TA), conocemos la cantidad de ítems que queremos suprimir o falsificar por cada categoría pero no sabemos qué elementos tenemos que elegir. Los PA's harán esta función, se encargarán de seleccionar los artículos que se van a modificar siguiendo las estrategias que hemos definido según el tipo de perturbación.

Si estamos en una estrategia de "suppression", es decir, eliminamos la puntuación de algún ítem, tenemos los siguientes métodos:

- **Random**, la elección del ítem es totalmente aleatoria, no existe ninguna preferencia.
- **Least Popular**, escogemos los ítems que tienen de media menor puntuación, de esta forma la estrategia de "suppression" afectará menos a la precisión del sistema de recomendación. No será lo mismo suprimir una valoración de un ítem que valga 5 que si su valor es 1.
- **Least voted**, en este caso escogeremos los artículos que han sido menos votados, sin importarnos la puntuación dada solo la cantidad de votos. Es decir, miraremos cuantos usuarios han votado ese ítem. Escogeremos en primer lugar los que tengan un valor más pequeño ya que afectará menos a la calidad de la precisión que si escogemos ítems muy votados.

Para la estrategia de "forgery" también tenemos tres métodos. En este caso la estrategia consiste en darle una puntuación a un ítem que no tiene :

- **Random**, que como en el caso anterior la elección de los artículos es totalmente aleatoria.
- **Most popular**, ahora escogeremos los ítems que tienen de media una mayor puntuación. Nos interesa, al contrario que en "suppression" seleccionar los artículos que tienen una mejor puntuación para que la precisión no se vea muy afectada.
- **Most voted**, en este caso seleccionamos los elementos que han sido más votados sin importarnos la puntuación dada, solo la cantidad de votos. Nos interesa como en "Most popular" elegir los ítems con mayor

número de votos para que al puntuar la película la precisión se vea lo menos afectada.

3.2 Forma de puntuar.

En la estrategia de "suppression" no hay confusión, sólo podemos eliminar la puntuación que tenía el ítem. En cambio en la estrategia de "forgery" podemos plantearnos diferentes maneras de puntuar los artículos, por ejemplo: dar el valor medio de las puntuaciones del ítem, el valor que más se repite, es decir, la moda, totalmente aleatorio, etc.

Al final decidimos que el valor a asignar al ítem sería totalmente aleatorio de 1 a 5, ya que consideramos que era el método más cercano a la realidad.

3.3 Implementación:

Para facilitar el diseño hemos dividido las tareas en dos partes que contienen distintos bloques como se muestran en la imagen inferior.

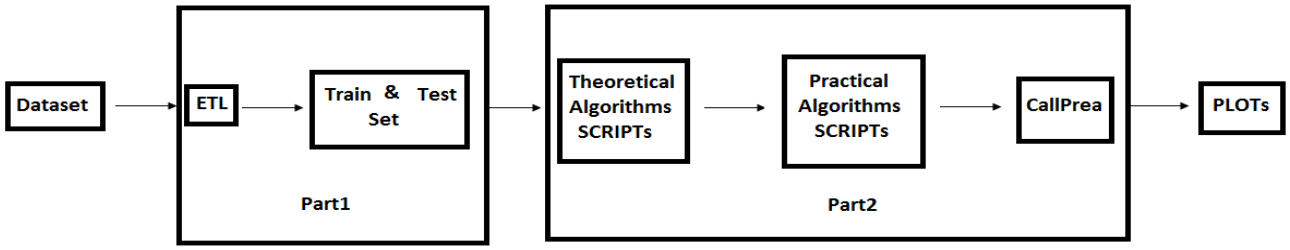


Figura 4.Diagrama de bloques

En la primera parte, nos descargamos los datasets que contienen la información con la que vamos a trabajar, realizamos un proceso ETL (Extract, Transform & Load) en el que extraemos los datos que nos interesan de los datasets y los transformamos para poderlos cargar en los scripts de MATLAB en formato csv.

El primer script "Train & Test Set " divide para cada usuario los ítems puntuados en dos conjuntos: training set y test set para que después el script PREA pueda realizar el cálculo de las métricas. También nos proporciona el valor de los perfiles tanto de usuario como el de la población y genera la matriz de ratings con las puntuaciones de todos los usuarios.

En la segunda parte tenemos los algoritmos teóricos (TA), los prácticos(PA) y la llamada al toolkit PREA. Los algoritmos TA contienen las funciones que comparan los perfiles de usuario y población. Según el valor de ρ o σ, que hayamos introducido, tendremos un array por usuario que nos indicará el porcentaje de ítems por categoría en los que debemos aplicar "forgery" o "suppression".

Los scripts prácticos (PA) implementan las estrategias de selección de los ítems en los que se aplican los mecanismos de perturbación. Según el tipo de estrategia seleccionada elegirán unos ítems u otros hasta llegar al porcentaje que el algoritmo TA haya marcado y modificaremos la matriz de ratings, eliminando la puntuación que había en el ítem si se aplica "suppression" o añadiendo una puntuación de forma aleatoria si se aplica "forgery".

A continuación, elegimos un sistema de recomendación del grupo a analizar y aplicamos PREA que nos calcula una seria de métricas (MAE y RMSE) para indicar la calidad de nuestro sistema de recomendación.

Por último, el script plot realizará las gráficas para ver los resultados.

Uno de los mayores inconvenientes del trabajo es la gran cantidad de información con la que hay que trabajar. Se realizan muchos cálculos para cada usuario: división del dataset en training y test set, creación de perfiles para cada usuario, cinco algoritmos teóricos, tres tipos de estrategias prácticas por cada tipo de perturbación, uso de PREA para diferentes sistemas de recomendación, etc.

A esto hay que añadirle que se deben repetir los cálculos para diferentes valores de rho y sigma y, por último, aplicar "crossvalidation".

Por lo que era necesario intentar optimizar todo lo posible el código y guardar la información en distintos archivos para no tener que repetir desde el principio todos los procesos, lo que convertiría el trabajo en inviable.

3.3.1. Selección de datasets.

Es posible encontrar en Internet una gran variedad de datasets utilizados por los sistemas de recomendación, el único inconveniente que nos limitaba la selección era que los ítems puntuados debían estar clasificados por categorías para poder crear posteriormente los perfiles.

Esta condición nos redujo mucho las opciones disponibles ya que no es una característica común en todos ellos. Es decir, si queríamos usar un dataset de películas, y entre ellas se encontraba "Toy Story" necesitaríamos tener un fichero en donde los usuarios puntuaran esta película y otro fichero que nos dijera que es una película infantil y de aventuras.

Desde un primer momento pensamos en MovieLens[9] ya que cumplía con esta condición. En su web disponen de diferentes datasets, elegimos por optimización y por tamaño el "movielens 100k" que contiene 100.000 ratings para 943 usuarios y 1682 películas, lo que representa una densidad de puntuación del 6,3%. Lo que nos indica la gran dispersión de las matrices.

[9] https://grouplens.org/datasets/movielens/

Cada dataset contiene gran cantidad de ficheros pero a nosotros sólo nos interesan dos, en el caso de movieLens: u.data y u.item. El primero almacena en columnas: el identificador de usuario, el identificador de la película y la puntuación y un campo de timestamp como podemos ver en la figura inferior.

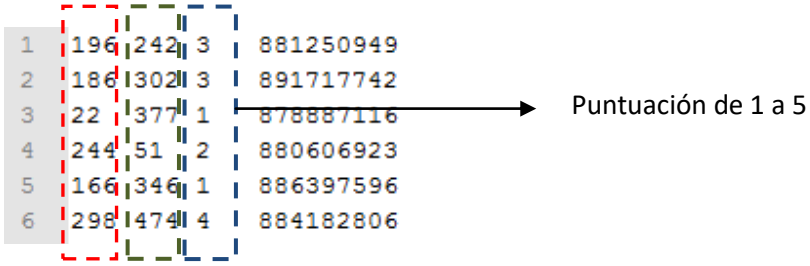


Figura 5. Extracto fichero u.data de MovieLens-100k

En el fichero u.item tenemos el nombre de la película, un enlace a la web imdb[10] y un array indicando a qué genero corresponde la película.

```
1|Toy Story (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)|0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0|0
2|GoldenEye (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?GoldenEye%20(1995)|0|1|1|0|0|0|0|0|0|0|0|0|0|0|1|0|0
```

Figura 6. Extracto fichero u.item de MovieLens-100k

Los valores 0 y 1 del array indican si la película pertenece a un determinado género (valor 1) o si no pertenece (valor 0). Cada columna está vinculada a una categoría o género, como muestra la figura inferior.

```
unknown|0
Action|1
Adventure|2
Animation|3
Children's|4
Comedy|5
Crime|6
Documentary|7
Drama|8
Fantasy|9
Film-Noir|10
Horror|11
Musical|12
Mystery|13
Romance|14
Sci-Fi|15
Thriller|16
War|17
Western|18
```

Figura 7.Fichero u.genre

[10] http://www.imdb.com/

También hemos usado el dataset de yahoo[11], relacionado con canciones y que cumplía el requisito comentado anteriormente donde los ítems están clasificados por género.

La forma de extraer la información es muy parecida al caso de MovieLens ya que también disponemos de ficheros que contienen las puntuaciones y otros que almacenan la clasificación de las canciones por género. La gran diferencia con el caso anterior es que no se puede descargar los datos automáticamente, necesitas una aprobación que tarda unos días en llegar y lo más importante es que el dataset es enorme (136.000 canciones con 1.8 millones de usuarios) lo que hacía imposible tratarlo en un tiempo razonable con nuestro diseño.

Decidimos reducirlo para poder aplicar nuestro programa. Al final, creamos un dataset reducido con 356 usuarios,136.000 canciones y con 150.000 ratings, lo que nos daba una densidad de 0,.31%, muchísimo más baja que en el caso anterior.

Otro inconveniente que tuvimos que tratar es que los datos que nos interesaban estaban divididos en archivos de 200.000 usuarios y en grupos de train y test. La solución elegida fue unir los dos conjuntos y extraer 356 usuarios.

3.3.2.Parte 1.Extracción de datos

Una vez descargados los datasets de movieLens (películas) y Yahoo (canciones) realizamos un pequeño proceso de ETL (Extract,Transform & Load) para incorporar los datos a los scripts de MATLAB.

Para tal tarea creamos un pequeño script en R para obtener dos ficheros con extensión csv: ratings.csv y genero.csv.

En "ratings.csv" solo extraemos las tres columnas de información que nos interesan (ID usuario, ID ítem y puntuación) y lo guardamos en el formato escogido que MATLAB lee fácilmente. Los archivos se guardan en un directorio con el nombre del dataset: yahoo o ml-100k.

Código script en R para crear ratings.csv:

```
# work folder
setwd("C:/TFM/ml-100k")
# read u.data without header
ratings<-read.csv("u.data", sep="\t",header=FALSE,stringsAsFactors =
FALSE)
# create dataframe
ratings<-as.data.frame(ratings)
#select column 1, 2  and 3
ratings2<-as.data.frame(ratings[,c(1,2,3)])
#write data in ratings.csv
```

[11] https://webscope.sandbox.yahoo.com/

```
write.table(ratings2, file =
"ratings.csv",row.names=FALSE,col.names=FALSE, sep="\t")
```

El fichero "genre.csv" necesitamos guardar una matriz con la clasificación de los ítems por sus categorías. Las filas serán los ítems y las columnas los géneros, como podemos ver en la siguiente figura.

1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
4	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figura 8. Matriz de géneros.

Este script, también realizado en R, es un poco más complicado que el anterior ya que no era sencillo extraer la información del fichero como estaba guardada. Además, necesitábamos crear un bucle que mirase cada ítem para generar la matriz de géneros.

Código script en R para crear genre.csv:

```
library(data.table)
setwd("C:/TFM/ml-100k")

#Read data

genre<-read.csv("u.item", sep=";",header=FALSE,stringsAsFactors =
FALSE)

genre2<-as.data.frame(genre[,3])
genres2 <- as.data.frame(tstrsplit(genre[,3], '[|]',
type.convert=TRUE), stringsAsFactors=FALSE)
colnames(genres2) <- c(1:6)
#List of genres
genre_list <- c("Action", "Adventure", "Animation", "Children",
"Comedy", "Crime","Documentary", "Drama", "Fantasy","Film-Noir",
"Horror", "Musical", "Mystery","Romance","Sci-Fi", "Thriller", "War",
"Western")

#empty matrix
genre_matrix <- matrix(0,943,18)
#set first row to genre list
genre_matrix[1,] <- genre_list
#set column names to genre list
colnames(genre_matrix) <- genre_list

#iterate matrix
for (i in 1:nrow(genres2)) {
  for (c in 1:ncol(genres2)) {
    genmat_col = which(genre_matrix[1,] == genres2[i,c])
    genre_matrix[i+1,genmat_col] <- 1
  }
}
```

```
#convert into dataframe
genre_matrix2 <- as.data.frame(genre_matrix[-1,],
stringsAsFactors=FALSE)
#remove first row, which was the genre list
for (c in 1:ncol(genre_matrix2)) {
  genre_matrix2[,c] <- as.integer(genre_matrix2[,c])
}
#write data to genero.csv
write.table(genre_matrix2, file = "genero.csv",
row.names=FALSE,sep="\t")
```

Una vez creados los dos ficheros csv, sólo debemos guardarlos en un carpeta con el nombre del dataset: movielens100k o yahoo.

3.3.3 Parte 1.Script Train &Test Set.

Lo hemos nombrado de esta forma porque es un paso muy importante pero realmente hace muchas más cosas que solamente dividir los ítems del usuario en training y test set.

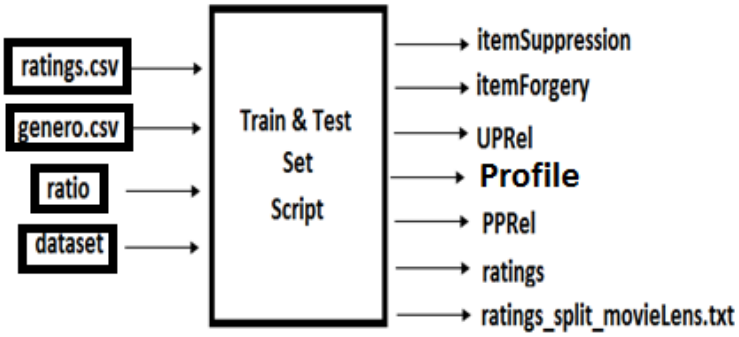


Figura 9. Bloque script Train &Test Set

Las entradas al script son:

- El fichero **"ratings.csv**" que contiene el ID del usuario, el ID de ítem y la puntuación.
- El fichero **"genero.csv**" que guarda una matriz de géneros donde cada artículo está clasificado según su clase.
- **dataset** indica si los datos pertencen a yahoo o a movieLens100k.
- **ratio**, almacena el porcentaje de ítems puntuados que irán al training set. En las simulaciones hemos usado el 50%.

Las salidas son:

- **itemSuppression,** es un array de celdas que nos guarda todos los ítems para cada usuario que pertenecen al training set y a los que se les puede aplicar la estrategia de "suppression". Es decir, son aquellos

ítems de cada usuario que pertenecen al conjunto de training y tienen puntuación.

- **itemForgery**, es también un array de celdas que contiene todos los ítems para cada usuario que pertenecen al conjunto de training y a los que se les puede aplicar la estrategia de "forgery", es decir, aquellos ítems de cada usuario que no están puntuados.
- **profile,** matriz de perfiles absolutos.
- **UPRel**, es el perfil relativo de cada usuario.
- **PPRe**l, es el perfil relativo de todos los usuarios.
- **ratings**, es la matriz inicial de puntuaciones extraída a partir de la información del fichero ratings.csv.
- **ratings_split_movieLens.txt**, es un fichero que guarda los ítems de cada usuario que pertenecen al test set y que usará más tarde el script PREA.

Para mostrar las diferentes fases de la implementación usaremos un pequeño conjunto de datos que nos sirvan de guía y que facilite la comprensión de lo que hace cada procedimiento.

Este modelo tendrá cinco películas con las siguientes categorías:

		ID género				
ID película	**Título**	**0**	**1**	**2**	**3**	**4**
1	**Toy Story**	0	0	1	1	1
2	**Golden Eye**	0	0	1	0	0
3	**Star Wars**	0	1	1	0	1
4	**Citizen Kane**	0	0	0	1	0
5	**TaxiDriver**	0	1	0	0	0

Figura 10. Tabla de películas y categorías.

Así que el fichero genre.csv guardará la siguiente matriz:

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Tendremos tres usuarios: John, Michael y Peter a los que les asignamos el ID 1,2,3 respectivamente.

- **Creación de ratings:**

A partir de los datos guardados en el fichero ratings.csv podemos crear su matriz de ratings. El ID de usuario y el ID de ítem nos dirán en que celda de la matriz ponemos la puntuación, en la figura 10 vemos cómo se hace.

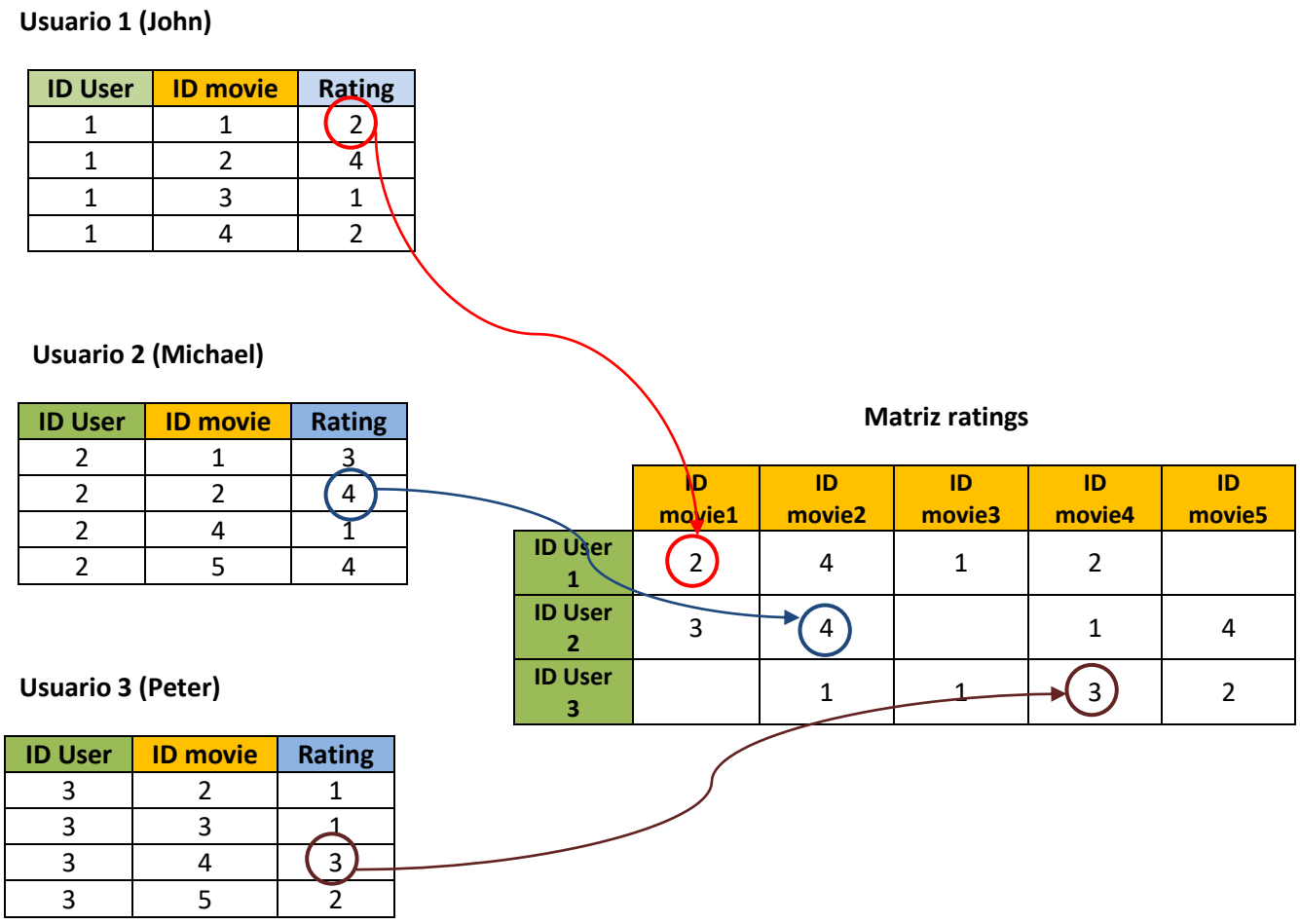


Figura 11. Creación de ratings.

Vemos que John (ID usuario=1) ha puntuado la película "Toy Story" (ID película=1) con un 2 por lo que esta puntuación está en la celda (1,1) de la matriz.

La matriz ratings quedará finalmente:

$$\begin{pmatrix} 2 & 4 & 1 & 2 & 0 \\ 3 & 4 & 0 & 1 & 4 \\ 0 & 1 & 1 & 3 & 2 \end{pmatrix}$$

Aquí debemos tener en cuenta que cero no significa que se puntúe con valor cero sino que el usuario no ha puntuado, es un matiz importante, no es lo mismo no puntuar que asignar un valor cero a la puntuación. Para nuestro diseño, un cero significará no puntuado y no supone ningún problema ya que el formato arff que guardará más adelante las matrices ratings trata los ceros como si no asignáramos valor a la celda. Además nuestras puntuaciones van de 1 a 5.

Código MATLAB para matriz ratings:

```
%========================================
% Create ratings matrix
%========================================

ratings=zeros(length(users),length(items));

for i=1:length(data(:,1))
    ratings(data(i,1),data(i,2))=data(i,3);
end;
```

- **Creación de training y test set:**

A partir de la matriz ratings debemos escoger qué ítems, para cada usuario, van a pertenecer al training set y al test set. Recordemos que a los ítems puntuados solo se les puede aplicar "suppression" y a los que no lo están, "forgery".

La variable ratio es la que guarda el porcentaje, por usuario, de artículos puntuados que van a pertenecer a training set, por lo que el resto de ítems puntuados pertenecerán al test set. En nuestros cálculos hemos usado habitualmente un 50%, lo que no supone que el 50% de los todos los ítems pertenezcan a training set y la otra mitad a test set. Quizás con el ejemplo se vea más claro, recuperemos la matriz de ratings para nuestros tres usuarios:

	IDmovie1	ID movie2	ID movie3	ID movie4	ID movie5
ID User 1	2	4	1	2	
ID User 2	3	4		1	4
ID User 3		1	1	3	2

Figura 12. Tabla de ratings.

Para el usuario ID 1 tenemos 4 películas puntuadas y nuestro ratio es del 0.5, por lo que elegimos aleatoriamente dos de las cuatro películas para que formen parte del training set y las otras dos sean del test set. Por ejemplo, las películas ID1 e ID3 forman parte del primer conjunto y se guardarían en la variable

itemSuppression{ID user}. y las películas ID2 e ID4 serán parte del test set y se guardarán en la variable testSet{ID user}

Tenemos que tener en cuenta que las estrategias de "forgery" y de "suppression" solamente se pueden aplicar en el training set y que el test set no puede contener ítems no puntuados.

Vemos que para el usuario ID1 tenemos una película no valorada, todos aquellos ítems que no lo están los guardamos en la variable itemForgery{ID user}

Así que para el usuario ID1 tenemos:

- itemSuppression {1} = [1,3]
- itemForgery {1} = [5]
- testSet {1} = [2,4]

El conjunto de training set estará formado por las películas de itemSuppression y de itemForgery , en este caso 3 películas, y tenemos otras dos que pertenecerán al test set. Es decir, no tenemos un 50% de training set como podríamos imaginar al inicio con el parámetro ratio, sino que realmente tenemos un 60% de películas que pertenecen al training y un 40% al test set.

Para el resto de usuarios podemos tener los siguientes valores, son datos aleatorios, si hiciéramos una nueva selección, los conjuntos de training y test set no contendrían las mismas películas.

	ID user 1	ID user 2	ID user 3
Training set	itemSuppression = [1,3] itemForgery = 5	itemSuppression = [2,5] itemForgery = 3	itemSuppression = [2,4] itemForgery = 1
Test set	testSet = [2,4]	testSet = [1,4]	testSet = [3,5]

Figura 13. Tabla resumen training y test set

- **Creación de perfiles:**

Los sistemas de recomendación colaborativos necesitan trabajar con perfiles. Los crearemos de dos tipos, un perfil para cada usuario y otro para toda la población. Hay que tener en cuenta que deben ser perfiles relativos.

Los perfiles de usuario relativos, UPRel se calculan a partir de una nueva matriz que llamaremos "profile" y que se confecciona a partir de los ítems puntuados que pertenecen al conjunto de training, es decir el conjunto de itemSuppression y de la matriz de géneros.

Aunque itemForgery también pertenece al mismo conjunto de training solo almacena ítems no valorados por lo que no nos ayuda a generar el perfil.

Así para el primer usuario tenemos que los ítems puntuados que pertenecen a training son [1,3]. Vamos a la matriz de géneros para ver cómo está categorizada cada película.

ID user 1:

itemSuppression = [1,3] → $\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$

y sumamos las filas elegidas:

$$\begin{array}{r} 0\ \ 0\ \ 1\ \ 1\ \ 1 \\ +\ 0\ \ 1\ \ 1\ \ 0\ \ 1 \\ \hline 0\ \ 1\ \ 2\ \ 1\ \ 2 \end{array}$$

El array [0,1,2,1,2] será el perfil con valores absolutos del usuario 1. Si hacemos lo mismo para el resto de usuarios obtenemos la matriz de profile.

$$\begin{pmatrix} 0 & 1 & 2 & 1 & 2 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Ahora para obtener los valores relativos de los perfiles por usuario dividimos cada puntuación por la suma total por usuario.

$\begin{pmatrix} 0 & 1 & 2 & 1 & 2 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$ → $UPRel_{ID1} = \frac{(0,1,2,1,2)}{(0+1+2+1+2)} = \frac{(0,1,2,1,2)}{6} = [0, 0.16, 0.33, 0.16, 0.33]$

Realizando los mismos cálculos para el resto de usuarios tenemos el resto de perfiles:

$$UPRel_{ID2} = \frac{(0,1,1,0,0)}{(0+1+1+0+0)} = \frac{(0,1,1,0,0)}{2} = [0, 0.5, 0.5, 0, 0]$$

$$UPRel_{ID3} = \frac{(0,0,1,1,0)}{(0+0+1+1+0)} = \frac{(0,0,1,1,0)}{2} = [0, 0, 0.5, 0.5, 0]$$

Para calcular el perfil relativo de toda la población, PPRel, debemos sumar todas las columnas de profile y el array resultante dividirlo por la suma como en el caso anterior. Vemos el cálculo:

$$\begin{array}{rccccc} & 0 & 1 & 2 & 1 & 2 \\ + & 0 & 1 & 1 & 0 & 0 \\ & 0 & 0 & 1 & 1 & 0 \\ \hline & 0 & 2 & 4 & 2 & 2 \end{array} \longrightarrow \quad PPRel = \frac{(0,2,4,2,2)}{(0+2+4+2+2)} = (0,0.2,0.4,0.2,0.2)$$

Si en nuestros perfiles relativos, tanto de usuario como de población, aparecen valores nulos debemos modificarlos ya que los scripts que calculan las tuplas para aplicar las estrategias de "suppression" o de "forgery" no permiten valores cero debido a que las funciones de probabilidad tienen que ser estrictamente positivas. Así que usaremos un pequeño truco y le daremos un valor mínimo, 10^{-9}.

Código MATLAB para la creación de training, test set, UPRel y PPRel:

```
%=======================================
% Create Training Set, User Profile & Population Profile
%=======================================
itemSuppression=cell(1,length(users));
itemForgery=cell(1,length(users));
indexTest=cell(1,length(users));

profile=zeros(length(users),length(genre(1,:)));
UPRel=zeros(length(users),length(genre(1,:)));

for i=1:length(users)
     b=items(ratings(i,:)~=0);
     k=round(ratio*length(b));
     data=datasample(b,k,2,'Replace',false);
     %Train
     itemSuppression{i}=data;
     itemForgery{i}=items(ratings(i,:)==0);
     % Test
     indexTest{i}=b(~ismember(b,data));
```

```
      % Profile matrix (absolute values)
      profile(i,:)=sum(genre(itemSuppression{i},:));
      % User profile
      UPRel(i,:) = profile(i,:)/sum(profile(i,:));
end;
% Population profile
PPAbs=sum(profile,1);
PPRel=PPAbs/sum(PPAbs);

PPRel(PPRel==0)= 1e-9;
UPRel(UPRel==0)= 1e-9;
```

- **Fichero que almacena el Test Set.**

Necesitamos que PREA funcione con el mismo conjunto de training y test set para comparar resultados. Es necesario enviarle la información de algún modo, en PREA la solución es crear un fichero de texto que guarde los test set. La puntuación de los ítems se la enviamos a través de la matriz de ratings.

El formato de este fichero es "ID usuario ID ítem".Si nos fijamos en la figura 12, donde están los ítems que pertenecen al test set, el archivo de nuestro modelo quedaría:

ID usuario	ID ítem
1	2
1	4
2	1
2	4
3	3
3	5

Se guardan con el nombre "ratings_split_[*nombre dataset*].txt" en la carpeta raíz.

Código MATLAB para crear fichero que guarda los test set.

```
%=========================================
% Create rating split file with test set items
%=========================================

if exist(['ratings_split' dataset '.txt'], 'file')==2
  delete(['ratings_split' dataset '.txt']);
end;
for i=1:length(UPRel(:,1))
    A=repmat(i,length(indexTest{i}),1);
    B=indexTest{i}';

    % Save data training in file txt
    fid=fopen(['ratings_split' dataset '.txt'],'at');
    fprintf(fid, '%d\t%d\n', [A B]');
    fclose(fid);
end;
```

3.3.4 Parte 2. Scripts Algoritmos teóricos:

Como hemos visto en el apartado 3.1 tenemos cinco algoritmos teóricos (TA) distintos, cada uno de ellos internamente llama a una función que se corresponde con una estrategia de suppression y forgery distinta, optimizada o no.

Cada algoritmo TA tiene su propio script:

- **RSTA** (Random Suppression Theoretical Algorithm). Aplicamos la estrategia de "suppression" de forma aleatoria, por lo tanto sólo usamos σ. En nuestras simulaciones hemos trabajado con intervalo de [0,1] con incrementos de 0.1. El algoritmo llama a la función "RandomSuppression".
- **OSTA** (Optimized Suppression Theoretical Algorithm) igual que en el caso anterior pero ahora el algoritmo busca los porcentajes de forma optimizada. El algoritmo llama a la función "ForgerySuppression".
- RFTA(Random Forgery Theoretical Algortihm). Aplicamos la estrategia de "forgery" de manera aleatoria. Usamos solamente ρ en el intervalo [0,1] con incrementos de 0.1. El algoritmo llama a la función "RandomForgery".
- **OFTA** (Optimized Forgery Theoretical Algortihm). En este caso aplicamos solo la estrategia "forgery" de forma optimizada y como en el punto anterior ρ trabaja en el intervalo [0,1] con incrementos de 0.1. El algoritmo llama a la función "ForgerySuppression"
- **OFSTA** (Optimized Forgery Suppression Algorithm). Por último tenemos el caso en que trabajamos con valores de σ y ρ a la vez. Los intervalos de los dos parámetros trabajan entre [0,1] con incrementos de 0.1 como anteriormente. El algoritmo llama a la función "ForgerySuppression".

Es importante tener en cuenta que los porcentajes que indican la cantidad de ítems que tenemos que perturbar se aplican a la suma total del perfil absoluto de cada usuario y no del perfil relativo, más adelante trataremos el tema en mayor profundidad.

Además, aparecerá un error, al que hemos llamado error teórico (TA Error) provocado al calcular el porcentaje de ítems que queremos perturbar por categoría. Como es lógico si estos porcentajes dan resultados con decimales debemos redondear su valor ya que no podemos suprimir o falsificar parte de un ítem, esta diferencia entre el resultado exacto de ítems a perturbar y el redondeo es nuestro error teórico. Los datos se guardarán en un fichero de texto para poderlos representar en un histograma más tarde.

Además cada función nos calcula un valor de riegso de privacidad para cada usuario y cada valor de σ o ρ. Se crean grandes matrices de datos que

debemos tratar para conseguir un riesgo de privacidad relativo para cada valor de σ o ρ. Estos datos también los guardaremos en un fichero de texto.

Por último, las salidas que nos interesan de estos algoritmos teóricos y que serán utilizados por los algoritmos prácticos son:

- Las tuplas por usuario (catSuppression y catForgery) que indican el número de ítems que debemos eliminar por categoría.
- El error teórico (TA Error)
- El riesgo de privacidad (riskprivacy) que se produce para cada valor de σ y ρ.

1. RSTA (Random Suppression Theoretical Algorithm):

Como muestra la figura inferior los parámetros de entrada de este script son las variables de la izquierda que están en un recuadro:

- **Sigma**, parámetro de supresión que va desde 0 a 1.
- **Percentage**: Número de usuarios que queremos perturbar
- **TA (**Theoretical Algorithm), nombre del algoritmo teórico.

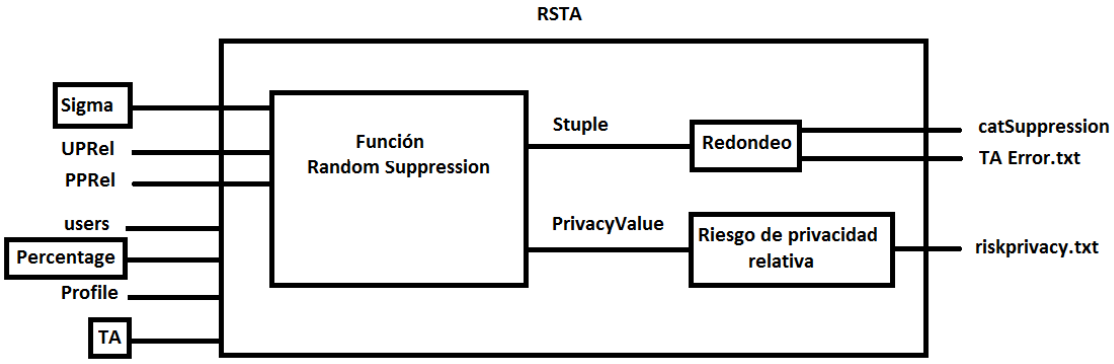


Figura 14. Bloques script RSTA.

El resto de variables (UPRel, PPrel, users. profile) las utiliza internamente el algoritmo teórico y provienen del script Train & Test.

- **Número usuarios que se perturban.**

El parámetro percentage, nos indica qué cantidad de usuarios queremos suprimir o falsificar. Normalmente hemos seleccionado el 100% de los usuarios para realizar nuestras simulaciones pero es un valor que nos permite ver como empeora la precisión de los algoritmos de recomendación al aumentar el número de usuarios perturbados.

Su uso es muy sencillo, recuperando nuestro ejemplo de guía en el que teníamos tres usuarios: John, Michael y Peter supongamos que queremos perturbar sólo el 60% de los usuarios.

usuarios =3

porcentaje perturbación=60% $\longrightarrow$ $round\left(\frac{60}{100}\ 3\ usuarios\right) = 2$

Calcularíamos el número de usuarios que queremos modificar (1.8 usuarios) y al ser un valor decimal lo debemos redondear con lo que tenemos que perturbar 2 usuarios. La selección de los usuarios se realiza empezando desde el primer usuario hasta llegar al número indicado. Así en nuestro caso serían los dos primeros: John y Michael.

Código MATLAB redondeo percentage:

```
N=round(percentage*length(users));
```

- **STuple:**

Para calcular el porcentaje de ítems que se deben, en este caso suprimir, llamamos a la función "Function_RandomSuppression", podemos ver el bloque de esta función en la figura 14. Debemos llamarla para cada usuario y para cada valor de sigma.

La función compara los perfiles relativos del usuario (UPRel) y el de la población (PPRel) y a partir del valor de sigma nos da una tupla llamada Stuple que indica el porcentaje de ítems que se debe modificar y un valor de privacidad PrivacyValue.

Recordemos que los valores profile, UPRel y PPRel de nuestro modelo eran:

Matriz profile:

$$\begin{pmatrix} 0 & 1 & 2 & 1 & 2 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

UPRel:

$$\begin{pmatrix} 0 & 0.16 & 0.33 & 0.16 & 0.33 \\ 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 \end{pmatrix}$$

PPRel:

$$(0 \quad 0.2 \quad 0.4 \quad 0.2 \quad 0.2)$$

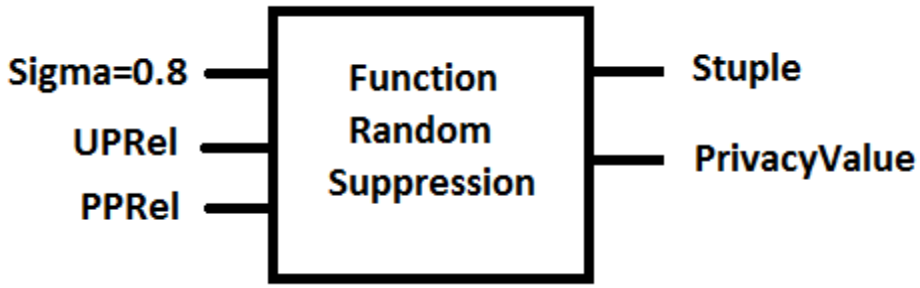


Figura 15. Bloque Función RandomSuppression

Si ejecutamos la función RandomSuppression con MATLAB para σ=0.8 y los valores de UPRel y PPRel obtenemos unas tuplas con el porcentaje de ítems a eliminar por categoría. Para los tres usuarios obtendríamos la siguiente matriz:

$$Stuple = \begin{pmatrix} 0 & 0.13 & 0.26 & 0.13 & 0.26 \\ 0 & 0.4 & 0.4 & 0 & 0 \\ 0 & 0 & 0.4 & 0.4 & 0 \end{pmatrix}$$

Como decíamos en la introducción de los algoritmos teóricos este porcentaje se calcula a partir de la suma de la matriz de los perfiles absolutos de usuario, con el ejemplo se verá más claro:

Matriz Profile

$$\begin{pmatrix} 0 & 1 & 2 & 1 & 2 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \longrightarrow$$

Suma de los componentes de la fila

$$\begin{pmatrix} (0+1+2+1+2)=6 \\ (0+1+1+0+0)=2 \\ (0+0+1+1+0)=2 \end{pmatrix}$$

Por ejemplo, el usuario ID 1, multiplicaría cada porcentaje de su Stuple con el valor de la suma de los componentes de la primera fila de profile:

$$6 \cdot (0 \quad 0.13 \quad 0.26 \quad 0.13 \quad 0.26) = (0 \quad 0.78 \quad 1.56 \quad 0.78 \quad 1.56)$$

Así por ejemplo si en la columna 3 que corresponde al valor 1.56 perteneciera al género aventuras deberíamos suprimir o falsificar (en este caso suprimir ya que estamos con un algoritmo de "suppression") 1.56 películas de éste género, pero esto no es posible ya que no podemos dividir los ítems, así que decidimos redondear los valores obtenidos y lo guardamos en la variable catSuppression[{ID user}.

Si redondeamos quedaría:

$$round(0 \quad 0.78 \quad 1.56 \quad 0.78 \quad 1.56) = ((0 \quad 1 \quad 2 \quad 1 \quad 2))$$

Por lo que en el caso anterior deberíamos eliminar finalmente 2 películas de ese género, Otra cosa es qué ítems en concreto serán, que lo decidiremos con los algoritmos prácticos.

- **Error teórico.**

Esta diferencia entre el valor real asignado por nuestra función "random suppression" y el redondeo es lo que llamaremos el error teórico TA Error. Para calcular el error hemos utilizado MSE (Mean Squared Error), previamente hay que aplicar valor absoluto a la diferencia.

Para el primer usuario tendremos un TA Error :

$$TA\ Error = abs[(0 \quad 1 \quad 2 \quad 1 \quad 2) - (0 \quad 0.78 \quad 1.56 \quad 0.78 \quad 1.56)]$$
$$= (0 \quad 0.22 \quad 0.44 \quad 0.22 \quad 0.44)$$

Para calcular el MSE elevamos cada componente del array anterior al cuadrado, los sumamos y los dividimos por la longitud del array para sacar la media.

$$TA\ Error\ MSE = \frac{0 + 0.22^2 + 0.44^2 + 0.22^2 + 0.44^2}{5} = 0.0968$$

Si aplicamos los mismos cálculos para el resto de usuarios tenemos:

Usuario ID2:

$$2 \cdot (0 \quad 0.4 \quad 0.4 \quad 0 \quad 0) = (0 \quad 0.8 \quad 0.8 \quad 0 \quad 0)$$

$$round(0 \quad 0.8 \quad 0.8 \quad 0 \quad 0) = ((0 \quad 1 \quad 1 \quad 0 \quad 0))$$

$$TA\ Error = abs[(0 \quad 1 \quad 1 \quad 0 \quad 0) - (0 \quad 0.8 \quad 0.8 \quad 0 \quad 0)]$$
$$= (0 \quad 0.2 \quad 0.2 \quad 0 \quad 0)$$

$$TA\ Error\ MSE = \frac{0 + 0.2^2 + 0.2^2 + 0 + 0}{5} = 0.016$$

Usuario ID3:

$$2 \cdot (0 \quad 0 \quad 0.4 \quad 0.4 \quad 0) = (0 \quad 0 \quad 0.8 \quad 0.8 \quad 0)$$

$$round(0 \quad 0 \quad 0.8 \quad 0.8 \quad 0) = ((0 \quad 0 \quad 1 \quad 1 \quad 0))$$

$$TA\ Error = abs[(0 \quad 0 \quad 1 \quad 1 \quad 0) - (0 \quad 0 \quad 0.8 \quad 0.8 \quad 0)]$$
$$= (0 \quad 0 \quad 0.2 \quad 0.2 \quad 0)$$

$$TA\ Error\ MSE = \frac{0+0 + 0.2^2 + 0.2^2 + 0}{5} = 0.016$$

En resumen, para σ=0.8 tenemos un TA Error, en MSE, para cada usuario de:

$$\begin{pmatrix} 0.0968 \\ 0.016 \\ 0.016 \end{pmatrix}$$

- **Riesgo de privacidad en Random Suppression:**

Habría que repetir este proceso con todos los valores de σ que pusiéramos, como hemos comentado al principio vamos a trabajar con el intervalo [0,1]. El resultado se guarda en un fichero llamado "TA Error.txt" y que se almacena en la carpeta "data\ *[nombre dataset]* \ [*nombre TA algoritmo*] \error".

Por último, la función RandomSuppression calcula también a variable,PrivacyValue, que guarda el valor del riesgo de privacidad que se produce para cada usuario y cada valor de σ. En este caso el aumento de sigma ni empeora ni mejora el riesgo de privacidad ya que estamos asignando una parte proporcional, σ, a cada componente del perfil UPRel para aplicar la estrategia de suppression.

Así que al calcular el riesgo de privacidad relativa nos da cero para todo σ ya que el valor de privacyValue no varía.

Código MATLAB para buscar STuple, TAError y PrivacyValue

```
for i=1:length(sigma)
    for iUsers=1:N
        %STuple for each user and sigma
        [Stuple,~,PrivacyValue] = Function_RandomSuppression2
                                  (UPRel(iUsers,:),PPRel,sigma(i));
        a=Stuple'*sum(profile(iUsers,:));
        b=round(a);
        catSuppression(iUsers,:,i)=b;
        %Error TA
        TAError=abs(b-a);
        TAErrorMSE(iUsers,i) = mean((TAError.^2),2);
        c(iUsers,i)=PrivacyValue;
        d(iUsers,i)=((c(iUsers,i)-c(iUsers,1))/c(iUsers,1));
    end;
end;
%privacy
privacyMatrix=mean(d);
```

2. OSTA (Optimized Suppression Theoretical Algorithm)

Este algoritmo teórico es similar al anterior pero esta vez trabajamos con la función ForgerySuppression" que calcula de forma optimizada las distintas tuplas que contendrán el porcentaje de ítems que se deben eliminar.

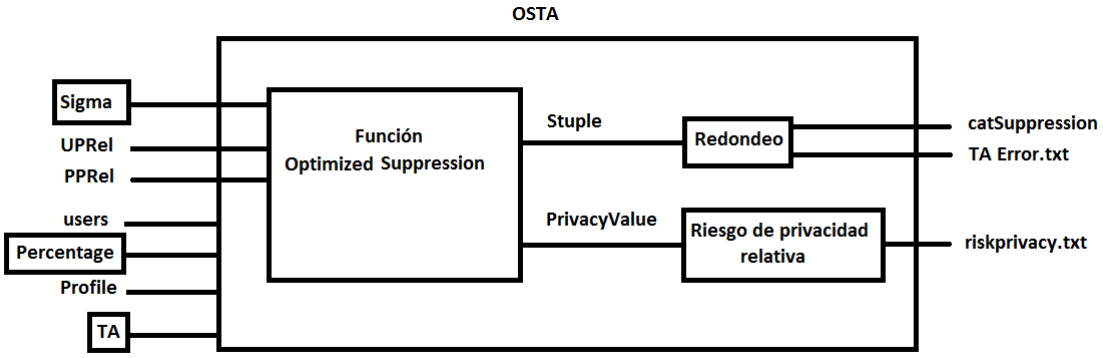


Figura 16. Bloque script OSTA

En este caso tenemos las mismas entradas que en al algoritmo anterior (sigma, percentage y TA) que ya hemos explicado.

Para poder trabajar solamente con la estrategia "suppression" con la función "ForgerySuppression." debemos poner a cero el parámetro ρ.

El cálculo de las tuplas stuple y los errores teóricos son idénticos al caso anterior, la única diferencia es en el tratamiento de la variable privacyvalue y como se realiza el cálculo del riesgo de la privacidad relativa.

- **Riesgo de privacidad en Optimized Suppression:**

Al aumentar el valor de σ desde 0 a 1 el valor de PrivacyValue para cada usuario va decreciendo. Si calculamos los valores, por ejemplo, desde σ de 0 a 0.3 con los datos de nuestros tres usuarios obtenemos que los valores son:

PrivacyValue:

	σ=0	σ=0.1	σ=0.2	σ=0.3
ID user 1	0.0438	-0.0.161	-0.0348	-0.0348
ID user 2	0.8219	0.7753	0.7425	0.7370
ID user 3	0.8219	0.7753	0.7425	0.7370

Figura 17. Tabla de privacidad OSTA

Vemos que en los casos de optimización hay una reducción del riesgo de privacidad al aumentar la "suppression".

Hay que fijarse que el término PrivacyValue(i,1) es siempre mayor que el resto de valores de riesgo de privacidad por lo que, para mantener siempre valores positivos, lo pondremos siempre a la izquierda en la resta.

Los valores relativos se calculan para cada fila de la siguiente forma:

$$Risk\ PrivacyValue\ Relative\ (i,j) = \frac{PrivacyValue(i,1) - PrivacyValue(i,j)}{PrivacyValue(i,1)}$$

Risk PrivacyValue Relative per user:

	σ=0	σ=0.1	σ=0.2	σ=0.3
ID user 1	0	1.3675	1.7945	1.7945
ID user 2	0	0.057	0.096	0.1033
ID user 3	0	0.057	0.096	0.1033

Figura 18. Tabla de riesgo de privacidad relativa por usuario OSTA

Para calcular el valor relativo de Risk PrivacyValue para cada σ debemos calcular la media entre todos los usuarios, es decir para cada columna sumamos los valores y lo dividimos por el total de usuarios. Si lo multiplicamos por 100 lo tenemos en porcentaje

Reduction Relative Risk PrivacyValue:

	σ=0	σ=0.1	σ=0.2	σ=0.3
Reducción Relativa PrivacyValue	0	0.49	0.65	0.67

Figura 19. Tabla de reducción relativa de riesgo de privacidad por usuario OSTA

Como vemos los datos nos indican que al aumentar σ el riesgo de privacidad disminuye.

Los datos los guardamos en un fichero llamado riskprivacy.txt en "data\[*dataset*]\ [*algoritmo teórico*]\"

Código MATLAB para buscar STuple, TAError y PrivacyValue

```
for i=1:length(sigma)
    for iUsers=1:N
        %STuple for each user and sigma
        [~,Stuple,~,PrivacyValue]=Function_ForgerySuppression
                                  (UPRel(iUsers,:),PPRel,0,sigma(i));
        a=Stuple'*sum(profile(iUsers,:));
        b=round(a);
        catSuppression(iUsers,:,i)=b;
        %Error TA
        TAError=abs(b-a);
```

```
        TAErrorMSE(iUsers,i) = mean((TAError.^2),2);
        c(iUsers,i)=PrivacyValue;
        d(iUsers,i)=((c(iUsers,1)-c(iUsers,i))/c(iUsers,1));
    end;
end;
privacyMatrix=mean(d);
```

3. **RFTA(Random Forgery Theoretical Algorithm).**

Este algoritmo utiliza la función "randomForgery", ahora no aplicamos "suppression" como en los dos casos anteriores sino que nos interesa utilizar solamente "forgery". Es un algoritmo teórico no optimizado.

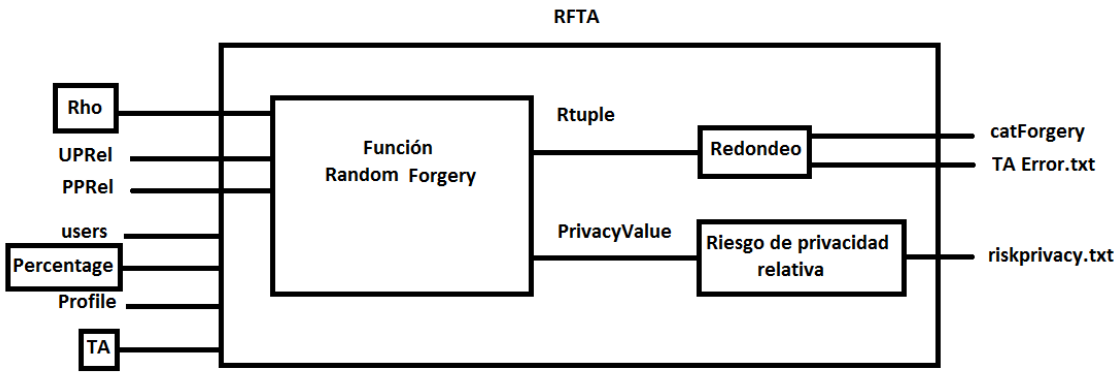


Figura 20. Bloque script RFTA

La forma de obtención de las tuplas y de los errores teóricos son idénticos a los casos anteriores, la única diferencia es que se guardan los arrays en la variable Rtuple.

- **Riesgo de privacidad en Random Forgery.**

El punto más importante en este algoritmo es el tratamiento del riesgo de privacidad, que es distinto al de los dos casos anteriores. Ahora, aunque estemos en un algoritmo que no optimiza los resultados, el riesgo de privacidad va aumentando a medida que incrementamos la falsificación, cosa totalmente distinta en el caso de la optimización que va disminuyendo.

La forma de calcular el riesgo de privacidad relativa es también diferente, ahora cambiaremos el orden de los factores en la resta para que el resultado sea siempre positivo.

PrivacyValue:

	ρ=0	ρ=0.1	ρ=0.2	ρ=0.3
ID user 1	0.0438	0.4518	0.8211	1.1445
ID user 2	0.8219	1.0133	1.2636	1.5027
ID user 3	0.8219	1.0133	1.2636	1.5027

Figura 21. Tabla de privacidad RFTA

Ahora vemos que los valores de riesgo de privacidad van aumentando a medida que incrementamos ρ, por lo que los valores relativos se calculan para cada fila de la siguiente forma:

$$RiskPrivacyValue\ Relative\ (i,j) = \frac{PrivacyValue(i,j) - PrivacyValue(i,1)}{PrivacyValue(i,1)}$$

Risk PrivacyValue Relative per user

	ρ=0	ρ=0.1	ρ=0.2	ρ=0.3
ID user 1	0	9.310	17.75	25.13
ID user 2	0	0.378	0.537	0.828
ID user 3	0	0.378	0.537	0.828

Figura 22. Tabla de riesgo de privacidad relativa RFTA

Increase Relative Risk PrivacyValue:

	ρ=0	ρ=0.1	ρ=0.2	ρ=0.3
Incremento Relativo PrivacyValue	0	3.35	6.27	8.93

Figura 23. Tabla de incremento relativo riesgo de privacidad RFTA

Como vemos los datos nos indican que al aumentar ρ el riesgo de privacidad aumenta rápidamente.

Los datos los guardamos en un fichero llamado riskprivacy.txt en "data\[*dataset*]\ [*algoritmo teórico*]\"

Código MATLAB para buscar RTuple, TAError y PrivacyValue

```
for i=1:length(rho)
   for iUsers=1:N
       [Rtuple,~,PrivacyValue] = Function_RandomForgery
                                 (UPRel(iUsers,:),PPRel,rho(i));
       a=Rtuple*sum(profile(iUsers,:));
       b=round(a);
```

```
        catForgery(iUsers,:,i)=b;
        TAError=abs(b-a);
        TAErrorMSE(iUsers,i) = mean((TAError.^2),2);
        c(iUsers,i)=PrivacyValue;
        d(iUsers,i)=((c(iUsers,i)-c(iUsers,1))/c(iUsers,1));
    end
end
privacyMatrix=mean(d);
```

4.OFTA(Optimized Forgery Theoretical Algorithm).

Este algoritmo teórico utiliza la función "forgery suppresssion" para calcular los porcentajes de forma optimizada, en este caso nos interesa solamente la estrategia de forgery.

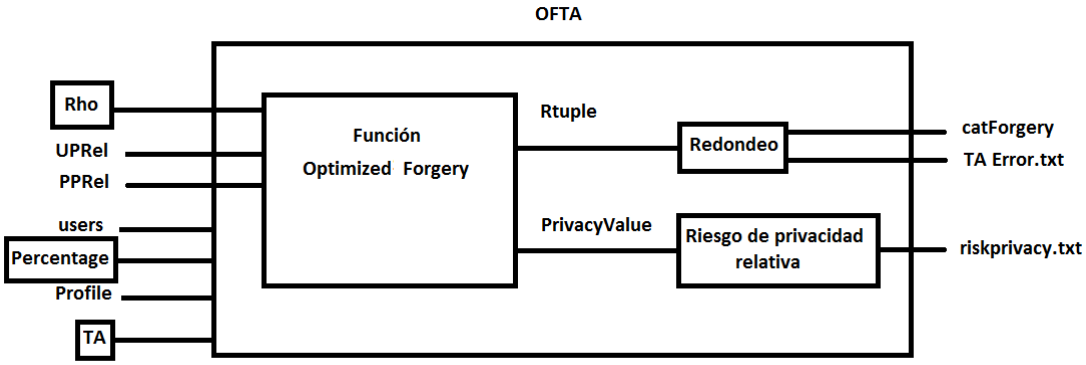


Figura 24.Bloques script OFTA

Para poder trabajar solamente con la estrategia "forgery" con la función "ForgerySuppression." debemos poner a cero el parámetro σ.

El tratamiento de los datos y los resultados son similares a los vistos con anterioridad, destacar que al trabajar con un método optimizado el riesgo de privacidad irá disminuyendo a medida que aumente ρ como sucedía en el caso de OSTA, en la "suppression".

Código MATLAB para buscar RTuple, TAError y PrivacyValue

```
for i=1:length(rho)
    for iUsers=1:N

[Rtuple,~,~,PrivacyValue]=Function_ForgerySuppression(UPRel(iUsers,:),
                          PPRel,rho(i),0);
        a=Rtuple'*sum(profile(iUsers,:));
        b=round(a);
        catForgery(iUsers,:,i)=b;
        TAError=abs(b-a);
        TAErrorMSE(iUsers,i) = mean((TAError.^2),2);
        c(iUsers,i)=PrivacyValue;
        d(iUsers,i)=((c(iUsers,1)-c(iUsers,i))/c(iUsers,1));
    end
end
privacyMatrix=mean(d)
```

5. OFSTA (Optimized Forgery Suppression Algorithm)

Aquí también trabajaremos con la función "forgery suppresssion" pero en este caso no mantendremos uno de los dos parámetros, σ o ρ, a cero sino que tendrán valores desde [0,1] en los dos casos lo que aumentará la cantidad de datos obtenidos y su tratamiento.

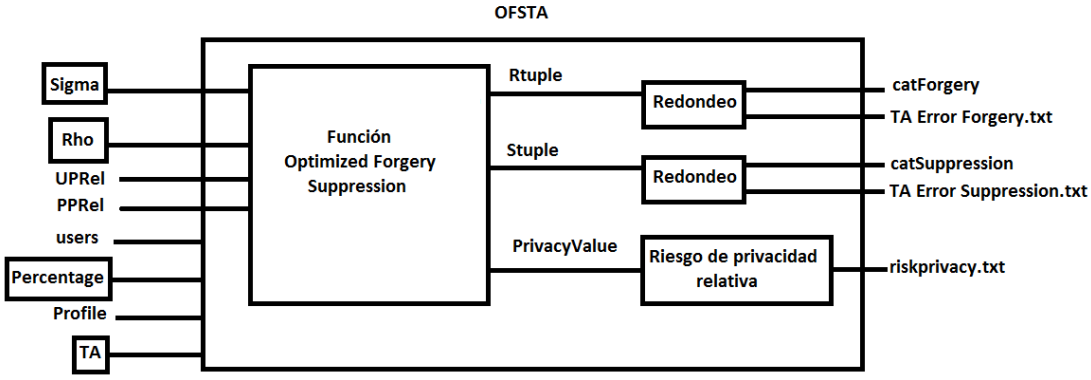


Figura 25.Bloques script OFSTA

Por ejemplo, si en el caso del dataset movielens hemos trabajado con 943 usuarios para los 11 casos distintos de σ o ρ. Ahora tenemos los mismos usuarios pero para 121 casos!!!.

El cálculo del error teórico es similar a los casos anteriores pero en este caso tenemos dos tipos de errores teóricos: los que se generan en la diferencia del redondeo de STuple (TA error suppression) y los que se producen con la diferencia del redondeo de Rtuple (TA error forgery).

El formato del riesgo de privacidad también se modifica, en los casos anteriores al trabajar con un solo parámetro, σ o ρ, teníamos una sola fila donde sus valores aumentaban o disminuían según el tipo de estrategia y algoritmo utilizado.

Ahora al tener dos parámetros tendremos una matriz de dimensiones m y n donde m es el número de valores de sigma y n el número de valores de n. Por ejemplo si utilizamos en nuestro caso de tres usuarios una σ de [0,0.3] con 0.1 de incremento y una ρ igual, tendríamos una matriz 4x4. y la guardaremos como tal en nuestro fichero txt.

Código MATLAB para buscar RTuple,STuple, TAError Suppression, TAError Forgery y PrivacyValue

```
for i=1:length(sigma)
    for j=1:length(rho)
         for iUsers=1:N
          [Rtuple,Stuple,~,PrivacyValue] =Function_ForgerySuppression

            (UPRel(iUsers,:),PPRel,rho(j),sigma(i));
```

```
            aSup=Stuple'*sum(profile(iUsers,:));
            bSup=round(aSup);
            aux1(iUsers,:)=bSup;

            TAErrorSup=abs(bSup-aSup);
            aux4(iUsers,:)= mean((TAErrorSup.^2),2);

            aFor=Rtuple'*sum(profile(iUsers,:));
            bFor=round(aFor);
            aux2(iUsers,:)=bFor;

            TAErrorFor=abs(bFor-aFor);
            aux5(iUsers,:)= mean((TAErrorFor.^2),2);


            aux3(iUsers)=PrivacyValue;
        end;
        catSuppression{i,j}=aux1;
        catForgery{i,j}=aux2;
        privacy{i,j}=aux3;
        TAErrorSupMSE{i,j}=aux4;
        TAErrorForMSE{i,j}=aux5;
    end;
end;

for i=1:length(sigma)
    for j=1:length(rho)
        for iUsers=1:N
            privacyRel{i,j}(iUsers)=(privacy{1,1}(iUsers)-
privacy{i,j}(iUsers))/privacy{1,1}(iUsers);
        end
        privacyRelMean{i,j}=mean(privacyRel{i,j});
    end
end
privacyMatrix=cell2mat(privacyRelMean);
TAErrorSup=cell2mat(reshape(TAErrorSupMSE,1,[]));
TAErrorFor=cell2mat(reshape(TAErrorForMSE,1,[]));
```

3.3.5 Parte 2. Scripts algoritmos prácticos:

Una vez tenemos la cantidad de ítems que queremos suprimir o falsificar por categoría necesitamos usar las estrategias indicadas en el apartado 3.1 para elegir qué ítems escogemos.

Recordemos que para la estrategia de "suppression" usaremos: random, least popular y least voted. El primer método, Random, es totalmente aleatoria, no existe ninguna preferencia en la elección de los ítems.

En la segunda opción, Least popular, escogemos las películas que tienen de media menor puntuación, teníamos dudas si para hacer la media debíamos escoger a todos los usuarios o sólo aquellos que realmente habían puntuado, al final escogimos la primera opción, para verlo más claramente analizamos el caso en que dos usuarios puntúan dos ítems de la siguiente forma

	Item ID1	Item ID2
Usuario ID1	2	2
Usuario ID2	1	No puntuado

Figura 26.Tabla ejemplo puntuación

Si solo consideramos valores puntuados, el ítem ID1 tiene una puntuación media de 1.5 y el ítem ID2 de 2 así que el primer artículo sería el elemento a eliminar pero no parecía una forma muy ajustada de analizar los ítems.

En el caso extremo en que un solo usuario puntúe a un ítem con un 5 este artículo nunca se eliminaría o sería uno de los últimos en hacerlo. Si escogemos a todos los usuarios para hacer la media, el ítem ID1 tendría una media de 1.5 y el ítem ID2 de 1 y sería este último producto el que suprimiríamos.

Recuperemos la matriz de ratings de nuestro modelo de tres usuarios y hagamos los cálculos:

Sumamos las columnas de la matriz:

$$\begin{pmatrix} 2 & 4 & 1 & 2 & 0 \\ 3 & 4 & 0 & 1 & 4 \\ 0 & 1 & 1 & 3 & 2 \end{pmatrix} \downarrow \quad (5 \quad 9 \quad 2 \quad 6 \quad 6)$$

Si hacemos la media para los tres usuarios tenemos:

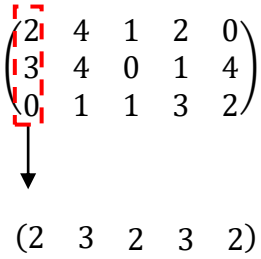
$$(1.66 \quad 3 \quad 0.66 \quad 2 \quad 2)$$

Este resultado nos da un orden de preferencia. Para la estrategia de "suppression" escogemos en primer lugar los valores más bajos. El orden sería:

$$(ítem\ 3 \quad ítem\ 1 \quad ítem4 \quad ítem5 \quad ítem2)$$

En el caso de empate como sucede con el ítem 4 y 5 es indiferente el artículo que se escoja primero.

En el tercer método, Least voted, la estrategia es elegir las películas que han sido menos votadas sin importar el valor de la puntuación, es decir, contar el número de usuarios que las han votado. Volviendo a nuestra matriz de ratings hacemos cálculos:



$$(2 \quad 3 \quad 2 \quad 3 \quad 2)$$

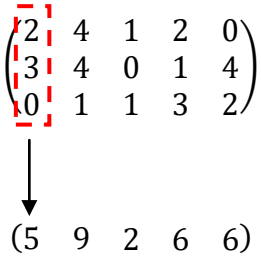
Vemos que en este caso el orden de elección de ítems podría ser:

$$(\acute{\imath}tem\ 3 \quad \acute{\imath}tem\ 1 \quad \acute{\imath}tem5 \quad \acute{\imath}tem4 \quad \acute{\imath}tem2)$$

En este caso hay varias repeticiones en los valores obtenidos y las combinaciones posibles en el orden de selección son diversas. Esto se debe a que es un ejemplo muy pequeño, en un dataset con un gran número de usuarios hay menos opciones. En nuestro código el programa escoge el primer valor que encuentra.

Este ejemplo no es válido ya que las puntuaciones son totalmente inventadas, pero en las simulaciones realizadas con datasets reales se observa una correlación entre los órdenes de películas menos populares y las menos votadas.

En el caso de "forgery" también usaremos tres estrategias. La primera, Random, como en el caso de "suppression" es totalmente aleatoria, la segunda, Most popular, busca los artículos que tienen mayor puntuación de media. En nuestro ejemplo habitual tendríamos:



$$(5 \quad 9 \quad 2 \quad 6 \quad 6)$$

El orden ahora quedaría:

$$(\acute{\imath}tem\ 2 \quad \acute{\imath}tem\ 4 \quad \acute{\imath}tem5 \quad \acute{\imath}tem1 \quad \acute{\imath}tem3)$$

Para el tercero método, Most voted, esta estrategia selecciona el orden de los ítems que han recibido más votos, es decir, elegirá primero aquel artículo que más usuarios han votado sin tener en cuenta el valor de la puntuación.

Analizando la matriz de ratings tenemos:

$$\begin{pmatrix} 2 & 4 & 1 & 2 & 0 \\ 3 & 4 & 0 & 1 & 4 \\ 0 & 1 & 1 & 3 & 2 \end{pmatrix} \rightarrow (2 \quad 3 \quad 2 \quad 3 \quad 2)$$

El orden ahora quedaría:

$$(\acute{\imath}tem\ 2 \quad \acute{\imath}tem\ 4 \quad \acute{\imath}tem5 \quad \acute{\imath}tem1 \quad \acute{\imath}tem3)$$

Al haber varios valores repetidos existen diferentes alternativas para escoger el orden. En nuestro código el programa escoge el primer valor que encuentra como en el caso de "suppression".

A veces el conjunto de ítems seleccionados para aplicar perturbación no alcanzan los objetivos indicados por los algoritmos teóricos. Esto se debe principalmente a que hay casos en que los ítems están clasificados en diferentes categorías a la vez lo que reduce la cantidad de opciones en la selección. Esta diferencia entre el objetivo y la cantidad de artículos que al final hemos seleccionado es lo que llamamos error práctico (PA Error)

En el caso que los ítems pertenezcan a un solo género este error es nulo o muy pequeños ya que el sistema es menos rígido y podemos hacer más combinaciones. Esto nos sucede con el de dataset de yahoo, donde cada canción pertenece a un sólo género, en cambio con el dataset de movielens, al pertenecer las películas a varias categorías a la vez, el error práctico es mayor.

Hemos diseñado tres scripts que implementan las diferentes estrategias de "suppression" y de "forgery":

- **Arffsuppression**, implementa las estrategias de "suppression": Random, Least popular y Least voted.
- **ArffForgery** implementa las estrategias de "forgery·: Random, Most popular y Most voted.
- **ArffForSup**, que aplicamos en el caso especial del algoritmo teórico "optimized Forgery Suppression", OFSTA, que trabaja tanto con σ y con ρ para que implemente todas las estrategias prácticas.

1. Script Arffsuppression:

Implementa las estrategias prácticas de "suppression" para algoritmos teóricos random y optimizado. Los parámetros de entrada del script son:

- **PASup**, indica la estrategia del algoritmo práctico en "suppression": random, least popular o least voted.

- **catSuppression**, guarda el número de ítems a suprimir por categoría de cada usuario.
- **sigma**, indica el grado de "suppression" con el que vamos a trabajar, $\sigma \in [0, 1]$
- **percentage**, el porcentaje de usuarios que se van a perturbar.
- **TA**, el algoritmo teórico con el que se trabaja.

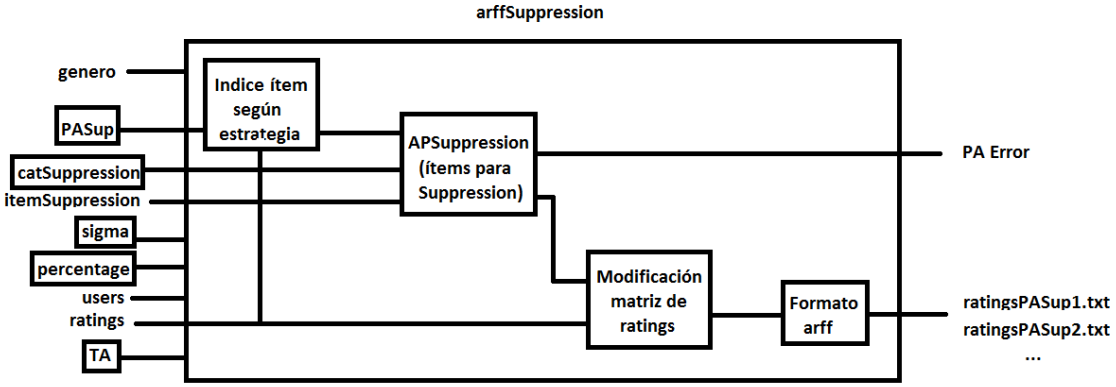


Figura 27. Bloques scripts arffSuppression

Las otras variables que necesita el script (itemSuppression, users, genero y ratings) se usan de forma interna. Las más importante serían itemSuppression que guarda, para cada usuario, los ítems del training que son candidatos a ser suprimidos, ratings que almacena la matriz de puntuaciones que han realizado los usuarios antes de que se elimine alguna valoración y genero que es la matriz que identifica cada ítem con su categoría.

El bloque de indice recoge el valor de la estrategia que tiene el parámetro de entrada PASup y realiza los cálculos sobre la matriz ratings para conocer el orden de los artículos a seleccionar en cada método práctico. El orden de cada ítem lo guardamos en una columna que añadimos a la matriz genero.

Si volvemos a nuestro ejemplo clásico de tres usuarios y cinco películas tenemos que para random no incorporamos ninguna columna. Así la matriz genero queda igual:

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Para least popular añadimos los valores de las medias de las puntuaciones:

$$(1.66 \quad 3 \quad 0.66 \quad 2 \quad 2) \longrightarrow \begin{pmatrix} 1.66 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 1 & 0 & 0 \\ 0.66 & 0 & 1 & 1 & 0 & 1 \\ 2 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Y para least voted hacemos los mismo con su índice:

$$(2 \quad 3 \quad 2 \quad 3 \quad 2) \longrightarrow \begin{pmatrix} 2 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 & 1 \\ 3 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Código MATLAB para crear los índices de PASup

```
%=========================================
% Create AP Suppresion index
%=========================================
if strcmp(PASup,'Least popular')
    indexAPSuppression=mean(ratings,1);
elseif strcmp(PASup,'Least voted')
    indexAPSuppression=sum(ratings~=0,1);
else
    indexAPSuppression=zeros(1,length(ratings(1,:)));
end;
```

El bloque interno APSuppression es el que se encarga de realizar la tarea de seleccionar qué ítem se escoge para cumplir con el objetivo de "suppression". El diagrama de flujo de este bloque sería:

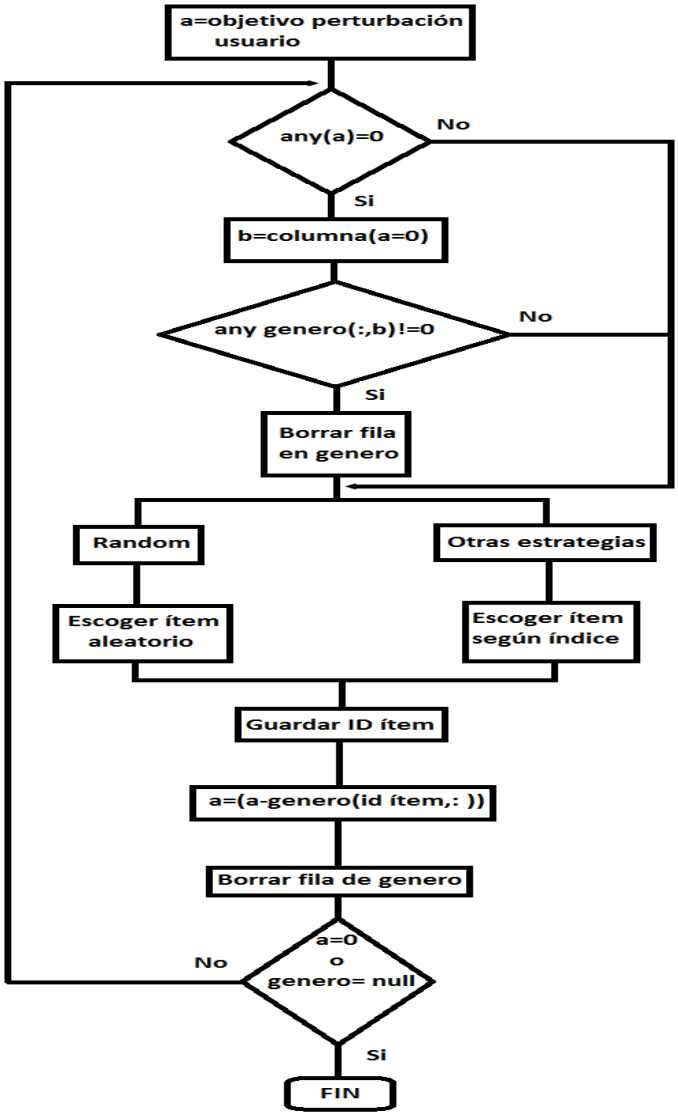


Figura 28, Diagrama de flujo de APSuppression

Veamos con nuestro modelo de tres usuarios como funciona, recuperamos los valores de Stuple para cada usuario y sigma=0.8 para el algoritmo teórico Random Suppression:

$$STuple = \begin{pmatrix} 0 & 1 & 2 & 1 & 2 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Esta matriz nos indica el número de películas que debemos anular por categoría y usuario.

- **Usuario 1 para Random:**

Ahora vamos a trabajar con la estrategia random Escojamos para ello la fila del primer usuario y la matriz de género para tal estrategia:

Array objetivo ítems a suprimir por categorias del id user 1:

Matriz de género:

$$\begin{pmatrix} 0 & 1 & 2 & 1 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Siguiendo el diagrama de flujo anterior, el array objetivo lo guardamos en la variable a, miramos si hay alguna celda a cero, en este caso la primera es nula por lo que comprobamos si en la primera columna de la matriz de género hay algún valor distinto de cero para anular la fila. Como no lo hay podemos elegir una fila aleatoriamente, por ejemplo la primera fila de la matriz género que corresponde al ID de película 1:

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Pasos:

1. Guardamos el id movie 1 para suppression.

2. Restamos la fila seleecionada al array a y lo volvemos a guardar en a:

a=(0 1 2 1 2)-(0 0 1 1 1)=(0 1 1 0 1)

3. Eliminamos la fila de la matriz género.

Ahora volvemos a repetir los pasos desde el principio:

Array objetivo a:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

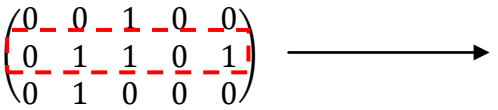
$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Fila eliminada

Ahora tenemos dos celdas a cero en el array objetivo,(variable a), la primera y la cuarta columnas. En la primera no hay ningún valor distinto de cero y en la

cuarta vemos que en la tercera fila de la matriz género hay un valor distinto por lo que esta fila no puede ser seleccionada y la eliminamos

Solo podemos elegir al azar la primera, segunda y cuarta fila, supongamos que elegimos la segunda fila que contiene los valores (0 1 1 0 1)



Pasos:

1. Guardamos el id movie 3 para suppression.

2. Restamos la fila seleccionada al array a y lo volvemos a guardar en a:

a=(0 1 1 0 1)-(0 1 1 0 1)=(0 0 0 0 0)

3. Eliminamos la fila de la matriz género.

Como ahora la variable a es cero damos el bucle por terminado y guardamos los ID de las películas elegidas :(1,3)

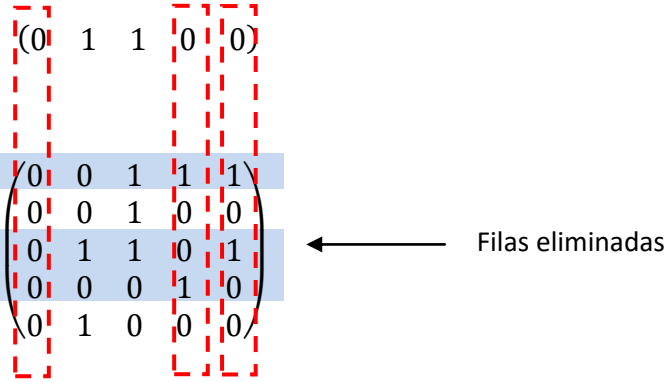
Vemos que hemos conseguido nuestro objetivo (a=0) por lo que el usuario ID1 no tiene error práctico.

- **Usuario 2 para Random:**

Hagamos el caso para el usuario ID2, nuestro array objetivo es (0 1 1 0 0) y la matriz de géneros es la misma del caso anterior.

Array objetivo ítems a suprimir por categorías del ID usuario 2:

Matriz de género:



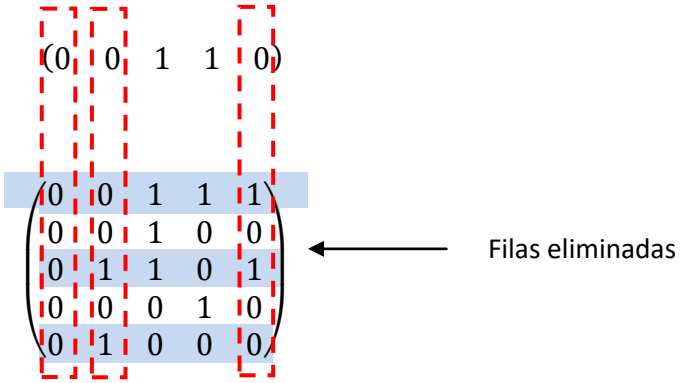
Vemos que hay tres celdas nulas en el array objetivo, anulamos las filas de la matriz género que contienen valores en esas columnas y nos quedamos con dos filas que vemos a simple vista que son las que necesitamos para logar el objetivo marcado y sin ningún error práctico. Así para el usuario ID 2 los ítems de las películas que debemos guardar para suprimir son (2,5)

- **Usuario 3 para Random:**

Para el ID usuario 3 haríamos lo mismo:

Array objetivo:

Matriz de género:



Sucede como en el caso anterior que tenemos que anular las filas primera, tercera y quinta de la matriz género por tener valores no nulos en las columnas nulas del array objetivo. Así que solo disponemos de dos filas que vemos que su suma forman el objetivo buscado, por lo que guardamos los ID de las películas para su posterior "suppression": (2,4)

Código MATLAB APSuppression

```
while not(isequal(D,arrayZero))

    itemFilter2=itemFilter(:,3:end);

    [row,~]=find(itemFilter2(:,D==0)~=0);
    itemFilter=removerows(itemFilter,row);

    if isempty(itemFilter)
        break;
    else
        if strcmp(APSup,'Least popular')
            [~,idx]=min(itemFilter(:,1));
            selectItem=itemFilter(idx,:);
        elseif strcmp(APSup,'Least voted')
            [~,idx]=min(itemFilter(:,1));
            selectItem=itemFilter(idx,:);
        else
            selectItem=datasample(itemFilter,1,1);
        end;
        rowItem=ismember(itemFilter,selectItem,'rows');
        itemFilter=removerows(itemFilter,rowItem);
        item(k)=selectItem(2);
        k=k+1;
        D=D-selectItem(3:end);
    end
end
```

Ahora que sabemos qué películas debemos suprimir sólo nos queda modificar nuestra matriz de ratings:

Películas para "suppression" con σ=0.8 TA=RSTA y PA=Random	
ID usuario	**ID Ítem**
1	1,3
2	2,5
3	2,4

Figura 29. Tabla resumen usuarios, ítems elegidos.

Por último, sólo tenemos que modificar nuestra matriz ratings en la celda que nos indica el ID usuario y el ID ítem y le ponemos valor cero, que por convención, representa que no hay puntuación realizada.

$$\begin{pmatrix} 2 & 4 & 1 & 2 & 0 \\ 3 & 4 & 0 & 1 & 4 \\ 0 & 1 & 1 & 3 & 2 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 & 4 & 0 & 2 & 0 \\ 3 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 2 \end{pmatrix}$$

Para las estrategias Least popular y Least voted necesitamos, como comentamos al principio, un índice de las ítems que nos indique en qué orden seleccionarlos. En cada paso elegiremos el ID del artículo que tenga un índice más pequeño. Haremos el primer usuario para los dos casos:

- **Usuario 1 para Least Popular:**

Para least popular teníamos esta matriz de géneros con el índice incorporado en una columna a la izquierda:

$$\begin{pmatrix} 1.66 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 1 & 0 & 0 \\ 0.66 & 0 & 1 & 1 & 0 & 1 \\ 2 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Aplicando el diagrama de flujo para encontrar que películas vamos a escoger tenemos para el primer usuario:

Array objetivo:

Matriz género:

$$(0 \quad 1 \quad 2 \quad 1 \quad 2)$$

$$\begin{pmatrix} 1.66 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 1 & 0 & 0 \\ 0.66 & 0 & 1 & 1 & 0 & 1 \\ 2 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Vemos que no eliminamos ninguna fila porque no hay ningún valor distinto de cero en la segunda columna de la matriz género. Ahora no podemos elegir al azar la fila de la matriz sino que tenemos que buscar el valor mínimo en la primera columna. En este caso corresponde a la tercera fila con valor 0.66

$$\begin{pmatrix} 1.66 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 1 & 0 & 0 \\ 0.66 & 0 & 1 & 1 & 0 & 1 \\ 2 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Pasos:

1. Guardamos el id movie 3 para suppression.
2. Restamos la fila seleccionada al array a y lo volvemos a guardar en a:

   a=(0 1 2 1 2)-(0 1 1 0 1)=(0 0 1 1 1)

3. Eliminamos la fila de la matriz género.

Volvemos a repetir los pasos con el nuevo objetivo y la matriz de géneros modificada.

Array objetivo:

Matriz géneros:

$$(0 \quad 0 \quad 1 \quad 1 \quad 1)$$

$$\begin{pmatrix} 1.66 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Fila a eliminar

Vemos que ahora tenemos en la segunda columna un valor no nulo en la matriz género por lo que debemos eliminar la última fila. De las otras tres filas

elegimos la que tiene menor índice en la primera columna, es decir, 1.66 que corresponde a la primera fila.

Visualmente se comprueba que la primera fila de género y el array objetivo actual son iguales por lo que seleccionando este ID de película terminamos el bucle. Las películas elegidas para suprimir serían (3,1)

- **Usuario 1 para Least voted**

Para Least voted la matriz de géneros es distinta ya que usa unos valores de índice diferentes a Least popular. En este caso tenemos la matriz género:

$$\begin{pmatrix} 2 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 & 1 \\ 3 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

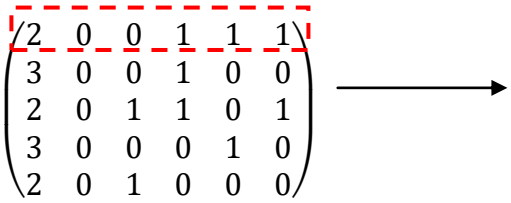
El array objetivo es:

Matriz géneros:

$$\begin{matrix} (0 & 1 & 2 & 1 & 2) \\ \\ \begin{pmatrix} 2 & 0 & 0 & 1 & 1 & 1 \\ 3 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 & 1 \\ 3 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{matrix}$$

Vemos, como en los casos anteriores, que en el primer paso no anulamos ninguna fila porque no hay valores no nulos en las columnas donde hay ceros en el array objetivo.

A continuación elegimos la fila con el índice para least voted que sea menor. En este caso tenemos tres opciones que se pueden elegir ya que el valor más pequeño es dos. En situaciones así nuestro script elige el primero que se encuentra en la matriz, en este caso sería la primera fila.



Pasos:

1. Guardamos el id movie 1 para suppression.

2. Restamos la fila seleccionada al array a y lo volvemos a guardar en a:

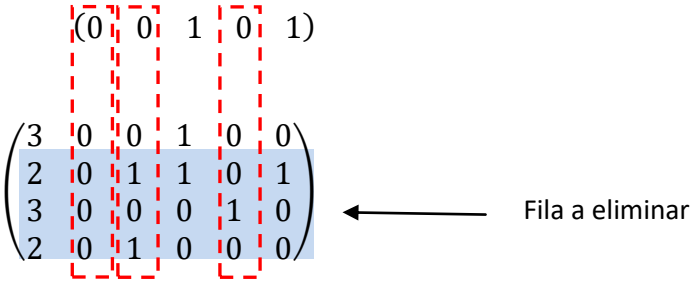
a=(0 1 2 1 2)-(0 0 1 1 1)=(0 0 1 0 1)

3. Eliminamos la fila de la matriz género.

Volvemos a repetir los pasos:

Array objetivo:

Matriz géneros:



Vemos que anulamos tres filas porque tienen valores diferentes de cero en las columnas donde el array objetivo es cero y nos quedamos solamente con una fila. Aquí se acabará el bucle porque ya no tenemos más filas en la matriz géneros.

También se observa que no alcanzamos el array objetivo marcado inicialmente. En este caso nos quedará un ítem en la quinta categoría sin eliminar, esta diferencia se convertirá en un error práctico que explicaremos con detalla un poco más adelante.

Las películas elegidas para suprimir en least voted serían (1,2)

- **Error práctico (PA Error):**

Como hemos explicado el error práctico se debe a que algunas veces es imposible llegar al objetivo indicado por los algoritmos teóricos. Se puede generar porque nos hemos quedado sin ítems que cumplan con las condiciones necesarias o porque los ítems que quedan no permiten ninguna combinación para seguir reduciendo la diferencia que existe con el objetivo indicado.

Lo trataremos igual que el error teórico que encontramos anteriormente, calcularemos el MSE (Mean Square Error) y lo guardaremos en un fichero de texto.

En nuestro modelo de guía hemos visto que para σ=0.8 y algoritmo práctico Least voted el usuario ID1 no llegaba al objetivo marcado y había una diferencia de (0 0 0 0 1). Supongamos que para el resto de usuarios tenemos los siguientes ítems sin poder tampoco suprimir:

ID user	Categoría 1	Categoría 2	Categoría 3	Categoría 4	Categoría 5
1	0	0	0	0	1
2	0	2	1	0	2
3	0	3	0	2	2

Figura 30. Tabla ejemplo con ítems sin eliminar

Calculamos los valores del error práctico (MSE):

usuario ID 1: $MSE\ PA\ Error = \frac{\sum x_i^2}{n} = \frac{1}{5} = 0.2$

usuario ID 2: $MSE\ PA\ Error = \frac{\sum x_i^2}{n} = \frac{9}{5} = 1.8$

usuario ID 3: $MSE\ PA\ Error = \frac{\sum x_i^2}{n} = \frac{17}{5} = 3.4$

Los ficheros de los errores prácticos guardan cada error producido para cada usuario y el valor de σ o ρ utilizados.

En nuestras simulaciones realizadas, para "suppression!, trabajamos con valores de σ=[0,1], es decir tendremos 11 columnas, una para cada valor de sigma. La ruta donde se almacenan es "data/[*dataset*]/[*TA algorithm*]/Error".

Código PA Error y función arff:

```
PAErrorMSE(iUsers,i) = mean((PAError.^2),2);

matlab2arff(ratings2,path);
```

Por último, nos queda transformar la matriz ratings modificada en formato arff para que los scripts que trabajan con PREA la puedan leer. La utilidad PREA ya nos proporciona un pequeño script que hace esta transformación llamado matlab2arff.

Guardaremos cada rating modificado en formato arff en un fichero texto llamado ratings[*PASup*][*sigma*].txt.

2. Script ArffForgery:

Este script implementa las estrategias utilizadas para "forgery" tanto cuando se aplica random como optimized.

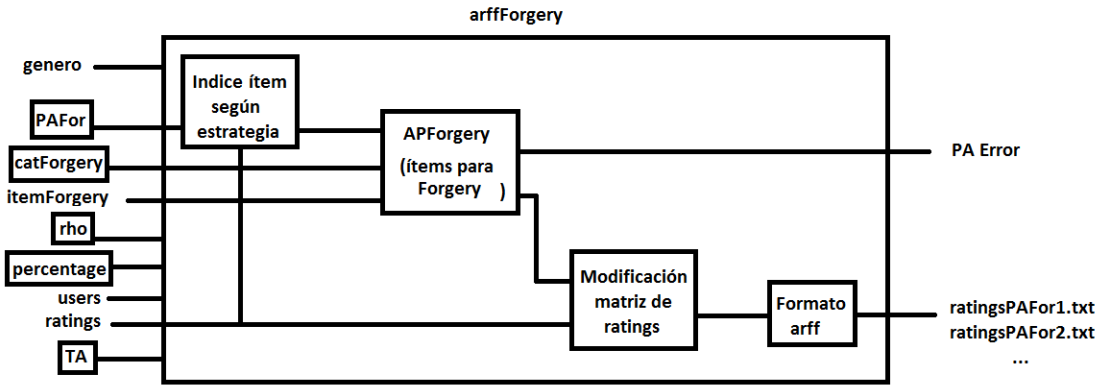


Figura 31. Bloques script arffForgery.

Como muestra la figura anterior los parámetros de entrada son:

- PAFor, indica el tipo de estrategia a utilizar en "forgery": random, most popular y most voted
- catForgery, almacena el número de ítems a falsificar por categoría.
- rho, indica el grado de "forgery", trabajamos con intervalo [0,1].
- percentage, el porcentaje de usuarios que se van a perturbar
- TA, el algoritmo teórico que se aplica.

El comportamiento de este script es muy similar al caso anterior donde se implementaba las estrategias de "suppression", ahora implementamos otras tres métodos de selección de ítems: random, most popular y most voted que ya hemos descrito en la introducción de los algoritmos prácticos.

Resumidamente, random elige el ítem a falsificar de forma aleatoria, most popular utiliza el mismo índice creado para least popular pero en vez de seleccionar el mínimo elige el máximo valor y most voted hace algo parecido, usa el mismo índice que least voted pero selecciona también el valor mayor.

El conjunto de ítems a elegir por cada usuario están guardados en la variable itemForgery en vez de itemSuppression y utilizamos el mismo algoritmo explicado anteriormente con la única diferencia que el array objetivo proviene de Rtuple en vez de Stuple y que en vez de buscar el valor mínimo buscamos el valor máximo para los métodos most popular y most voted.

El tratamiento del error también es idéntico que en el caso de "suppression". Los datos los guardaremos en la ruta 'data\[*dataset*]\[*TA algorithm*]\error'

- **Forma de puntuar:**

La gran diferencia con respecto al script ArffSuppression es la modificación de la matriz ratings, mientras que en el caso anterior eliminábamos puntuaciones en este caso hacemos lo contrario y puntuamos ítems que antes no tenían valoración.

La estrategia de puntuación elegida, como ya se ha explicado, es elegir un valor aleatorio entre 1 y 5.

Lo transformaremos en formato arff y lo guardaremos como un fichero texto llamado ratings[*PAFor*][*rho*].txt.

Código para puntuar:

```
for iItem=1:length(itemsToFor)
      ratings2(iUsers,itemsToFor(iItem))=randi([1 5]);
end;
```

3. Script ArffForSup:

Este script lo usamos cuando utilizamos el algoritmo teórico "Optimized Forgery Suppression". En este caso tenemos la combinación de las tres estrategias de "suppression" con las tres estrategias de "forgery", así que tenemos en total nueve posibles estrategias: Random Random, Random Most voted, Random Most Popular, Least Popular Random, etc.

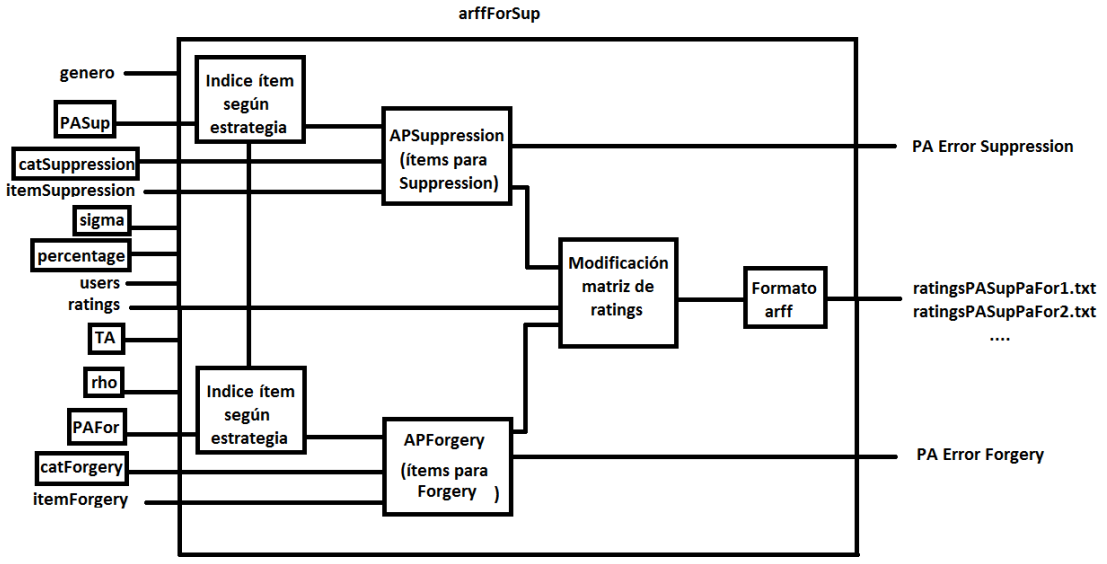


Figura 32. Bloques script arffForSup.

En primer lugar aplicamos las estrategias de "supression" y modificamos la matriz ratings con la eliminación de ítems que nos indica la función APSuppression. A continuación, hacemos lo mismo con las estrategias de "forgery" . Es interesante observar que podemos aplicarlas de forma independiente ya que nunca habrá que suprimir y falsificar ítems que pertenezcan a una misma categoría. Por ejemplo si aplicamos la función "forgerySuppression" con valores de σ=0.3 y ρ=0.1 para nuestro primer usuario tenemos que el porcentaje de ítems a suprimir es:

$$STuple = (0 \quad 0 \quad 0 \quad 0.1 \quad 0)$$

y para falsificar:

$$RTuple = (0 \quad 0.15 \quad 0 \quad 0 \quad 0.15)$$

Como decíamos Stuple suprime puntuación de ítems de la categoría 4 mientras que Rtuple falsificará ítems de categorías distintas: 2 y 5.

Otra característica diferente a los algoritmos prácticos anteriores es que, aunque el tratamiento para calcular los errores prácticos es idéntico, tenemos dos tipos distintos, el provocado al buscar ítems para suprimir (PA Error Suppression) y el generado al buscar los ítems para falsificar (PA Error Forgery).

Los ficheros también se guardarán en la ruta 'data\[*dataset*]\[*TA algorithm*]\error'

Por último, los transformaremos en formato arff y los guardaremos como un fichero texto llamado ratings[*PASup][PAFor*][*Sigma][rho*].txt.

3.3.6 Parte 2.Scripts callPREA:

En esta etapa vamos a medir la precisión de los sistemas de recomendación para ello usaremos el script PREA que permite a través de diferentes métricas ver cómo se comportan los diferentes algoritmos de recomendación. Hemos creado un script distinto para cada estrategia de perturbación:

- **callPREASup**, dedicado a la estrategia de "suppression".
- **callPREAFor**, se encarga de la estrategia de "forgery".
- **callPREAForSup**, en el caso que tengamos "suppression" y "forgery" a la vez.

El script PREA es un interfaz realizado en MATLAB que permite comunicar nuestros scripts de MATLAB con el programa PREA codificado en JAVA.

El problema principal que nos hemos encontrado en esta etapa era capturar los datos que nos interesaban ya que sólo se mostraban en pantalla y adaptar el

script PREA de MATLAB a nuestras necesidades ya que no estaba preparado para trabajar a partir de un fichero que contuviera los ítems de test set.

Una vez se ejecuta PREA los resultados aparecen en pantalla como muestra la figura inferior:

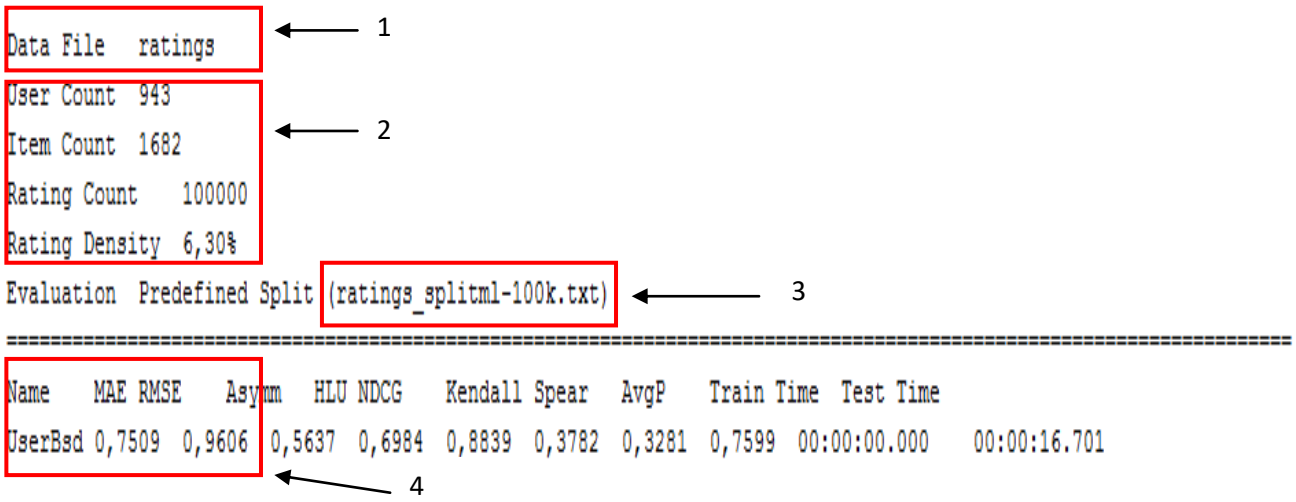


Figura 33. Pantalla con los datos obtenidos con PREA

La información que contiene el número 1 es el nombre del fichero de la matriz ratings en formato arff. El número 2 muestra la cantidad de usuarios y de ítems con los que trabajamos, el número de ratings que contiene y la densidad de ítems puntuados respecto al total de celdas. En este ejemplo hemos trabajado con el dataset ml-100k que contiene 943 usuarios, 1682 ítems y 100.000 ratings, 6.3% de densidad.

En la etiqueta número 3 aparece el nombre del fichero que contiene los ítems de cada usuario que pertenecen al test set y, por último, en el número 4 vemos a la izquierda el nombre del algoritmo de recomendación utilizado userbased y los valores de las métricas: MAE, RMSE; etc.

Los resultados obtenidos por pantalla los guardamos en un fichero txt llamado measures.txt.

1. Script callPREASup:

Este script lo usamos cuando trabajamos con los algoritmos teóricos de "suppression": Random Suppression y Optimized Suppression.

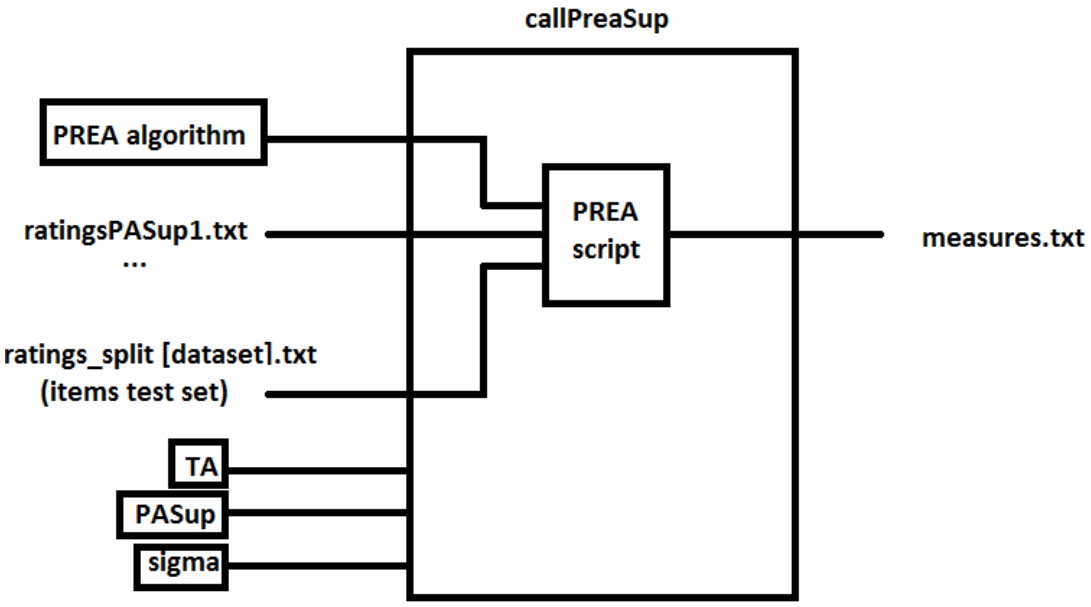


Figura 34 Bloques script callPREASup

En la figura superior vemos que el script tiene como parámetros de entrada:

- **PREA Algorithm**, el nombre del sistema de recomendación a analizar.
- **TA,** nombre del algoritmo teórico.
- **PASup**, nombre de la estrategía para seleccionar ítems para suprimir
- **sigma**, parámetro que indica el grado de "suppression".

Este tipo de scripts utiliza dos ficheros que son necesarios para calcular las métricas:

- EL fichero **ratings[*PaSup][sigma*].txt** que guarda la matriz de ratings ya perturbada para cada estrategia y cada valor de sigma en formato arff
- El fichero **ratings_split[*dataset*].txt** que guarda los ítems que pertenecen al test set.

Código llamada a PREA y captura de pantalla:

```
for i=1:length(sigma)
        file=[path '/ratings' PA num2str(i) '.arff'];
        copyfile(file,'ratings.arff','f');
        diary screen.txt
        prea(zeros(1,1),'ratings', algorithm, ['ratings_split' dataset
            '.txt']);
        diary off;
        delete('ratings.arff');
end;
```

2.Script callPREAFor:

En cambio este script trabajará con los algoritmos teóricos de "forgery": random forgery y optimized Forgery.

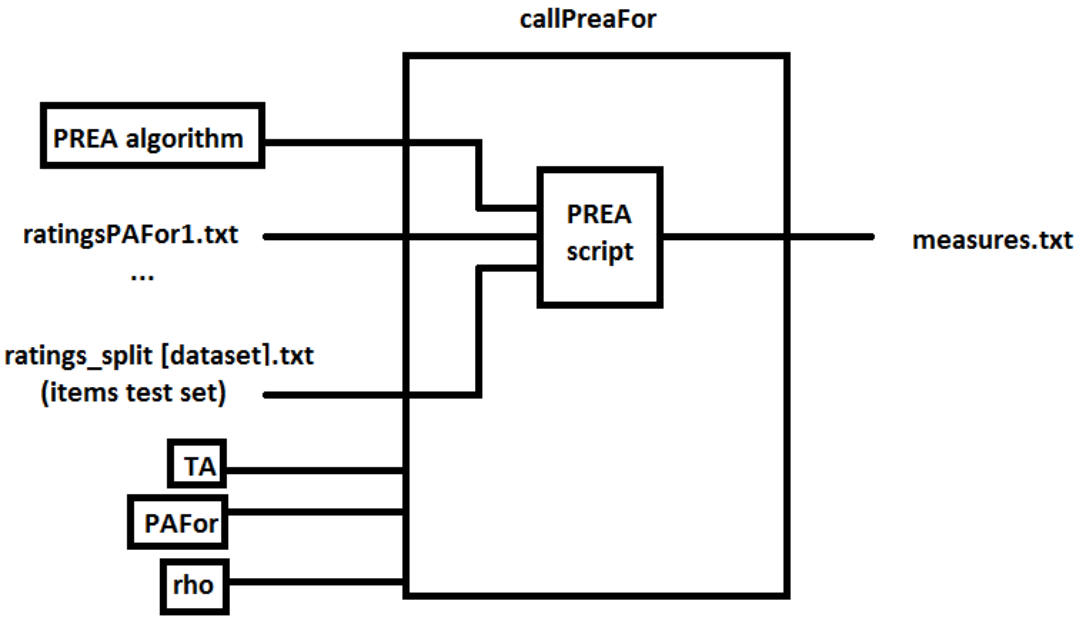


Figura 35. Bloques script callPREAFor

Es similar al script anterior, las únicas diferencias es que ahora trabajaremos con el parámetro rho en vez de sigma y con las matrices de ratings que han sido falsificadas.

3. Script callPreaForSup:

Por último, este script lo usaremos solamente cuando queramos aplicar estrategias de "suppression" y "forgery" a la vez.

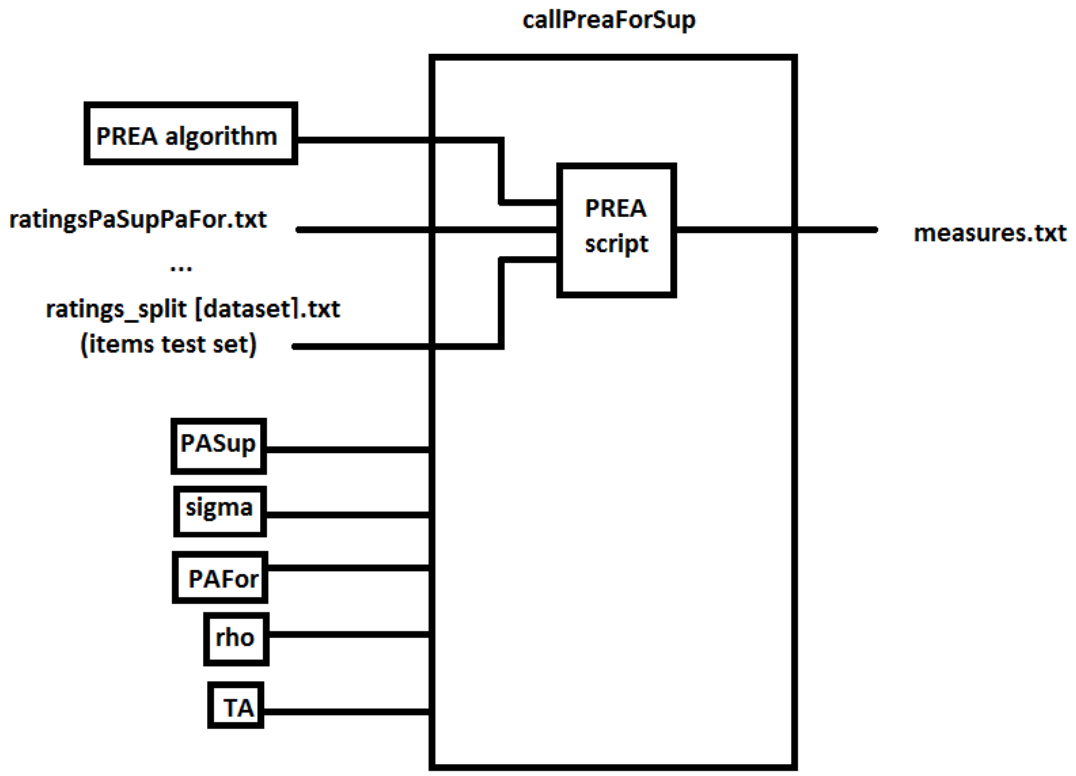


Figura 36. Bloques script callPREAForSup

El funcionamiento es muy similar a los casos anteriores las únicas diferencias es que ahora trabajamos con los dos parámetros: σ y ρ y con muchos más datos.

3.3.7 PLOT:

En la última etapa crearemos las gráficas. Como muestra la figura inferior en primer lugar haremos crossvalidation para garantizar que los resultados obtenidos son independientes de la partición realizada entre los datos de training y los de set y después calcularemos los valores relativos respecto al valor obtenido para σ=0 o $\rho = 0$

Los parámetros de entrada del script son:

- **dataset**, indicamos con qué dataset trabajamos: yahoo o ml-100k.
- **PREA algorithm**, indicamos el algoritmo de recomendación que vamos a usar.
- **sigma**, parámetro que indica el grado de "suppression".
- **rho,** parámetro que indica el grado de "forgery".
- **metricsWeWant,** las métricas con las que vamos a trabajar para ver sus gráficas, en este caso: MAE y RMSE.

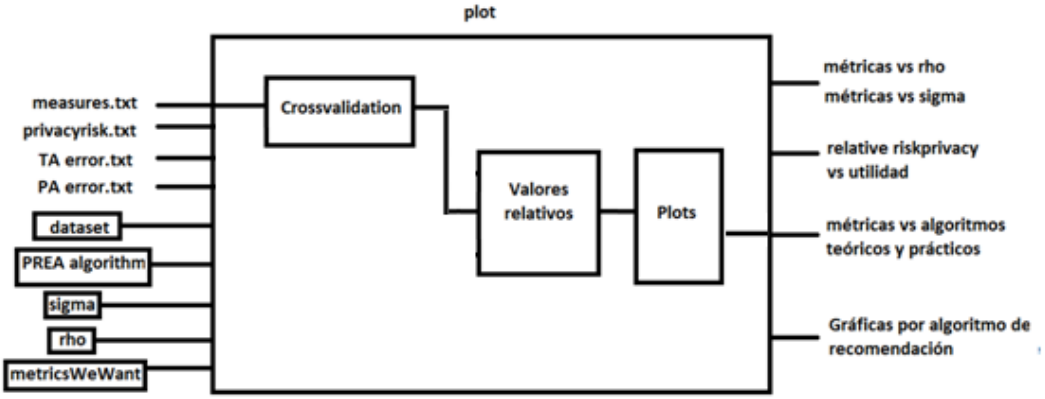


Figura 37. Bloques script plot

El funcionamiento de crossvalidation será sencillo, repetiremos todos los pasos 3 veces para cada uno de los algoritmos teóricos, prácticos y algoritmos de recomendación.

De esta forma obtendremos tres conjuntos distintos de ficheros para métricas, riesgo de privacidad, errores lógicos y prácticos. Asignaremos a cada grupo un número 1,2 y 3 y a continuación, haremos la media entre los ficheros del mismo tipo.

El siguiente bloque es calcular los valores relativos para los valores de las métricas, en el caso del riesgo de privacidad no es necesario ya que ya lo hicimos cuando implementamos los algoritmos teóricos.

Hemos creado diferentes tipos de gráficas para cada uno de los dos datasets y para diferentes sistemas de recomendación:

- **Por métricas**, para ver cómo se comportan MAE y RMSE al aumentar el nivel de σ y ρ.
- **Por tipo de algoritmo práctico**, para estudiar las diferencias en la precisión al aplicar un método práctico u otro.
- **Riesgo de privacidad relativa vs pérdida de utilidad**, para analizar como el tipo de algoritmo teórico, random u optimizado, afecta a la evolución del riesgo de privacidad
- **Histogramas** de errores teóricos y prácticos.

Código de crossvalidation para las métricas

```
TA='Random Suppression';
algorithm='userbased';
a=zeros(11,8);
aux1='Random abs';
path1=['C:\prea\data\ml-100k\' TA '\' algorithm];
for i=1:3
    filename=[aux1 num2str(i) '.txt'];
```

```
    pathFile = fullfile(path1, filename);
            aux2=dlmread(pathFile,'',1,0);
            a=a+aux2;
end

aux3=a/3;
```

4. RESULTADOS

Hemos simulado y analizado las gráficas comentadas en el punto anterior de un amplio abanico de datasets, sistemas de recomendación y algoritmos teóricos y prácticos:

- Dos datasets:movieLens-100k y yahoo
- Cinco algoritmos teóricos: RSTA,OSTA,RFTA,OFTA, OFSTA.
- Tres métodos para la estrategia de "suppression": random, least popular, least voted.
- Tres métodos para la estrategia de "forgery": random , most popular, most voted.
- Dos métricas: MAE y RMSE.
- Errores teóricos y prácticos.

No vamos a representar todas ellas ya que, en algunos casos, los resultados obtenidos son muy parecidos entre sí y no aportan información significativa a destacar. Por tanto, mostraremos solo aquellas que son sencillas de visualizar y que aportan un valor añadido al proyecto.

Todos los casos están simulados con un ratio del 50%, es decir, la mitad de los ítems puntuados están en training set y la otra mitad en test set y todos los usuarios aplican estrategias de perturbación (percentage=100%).

4.1 Dataset MovieLens-100k.

RS: Userbased y TA:OSTA (Optimized Suppression)

En la figura 38 vemos la evolución de la métrica MAE al aumentar σ, los porcentajes obtenidos son valores relativos respecto al valor de MAE sin perturbar (σ=0) y que vale en valor absoluto 0.7560.

Al aumentar la "suppression" se observa un empeoramiento de la precisión del sistema de recomendación, ya que cada vez MAE es mayor, llegamos a tener hasta un 32% para el caso de mayor perturbación (σ=1).

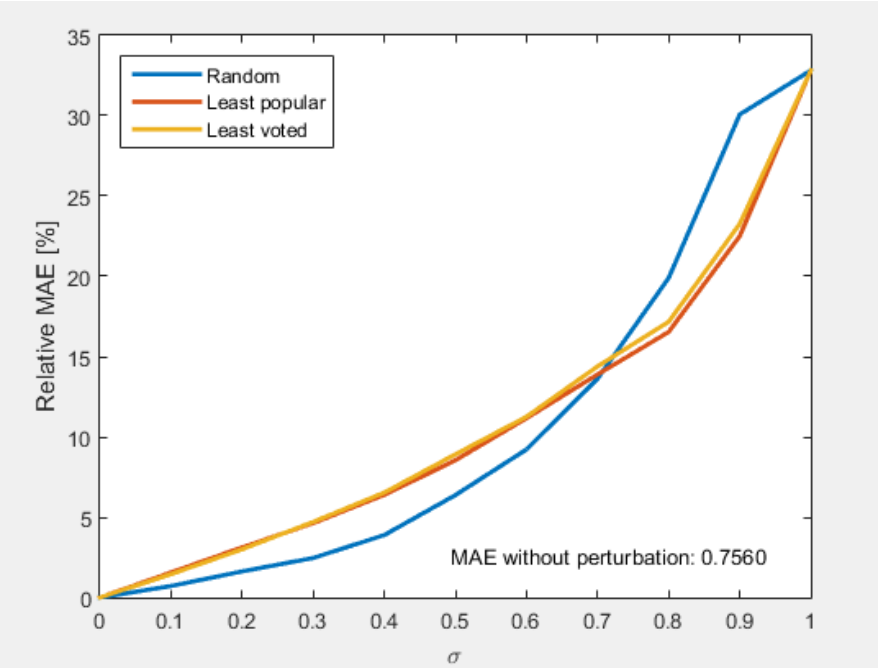


Figura 38. Evolución MAE relativa vs sigma para OSTA y Userbased

Es lo que esperábamos al perturbar los ítems las métricas que miden el error aumentan. Otra característica importante es el comportamiento similar que tienen los métodos Least popular y Least voted, sus curvas casi se superponen. Es interesante ver que hasta σ=0.8 son casi lineales.

Además la estrategia random tiene un mejor comportamiento que los otros dos métodos hasta σ=0.7 donde, a partir de este punto, Least popular y Least voted actúan mejor.

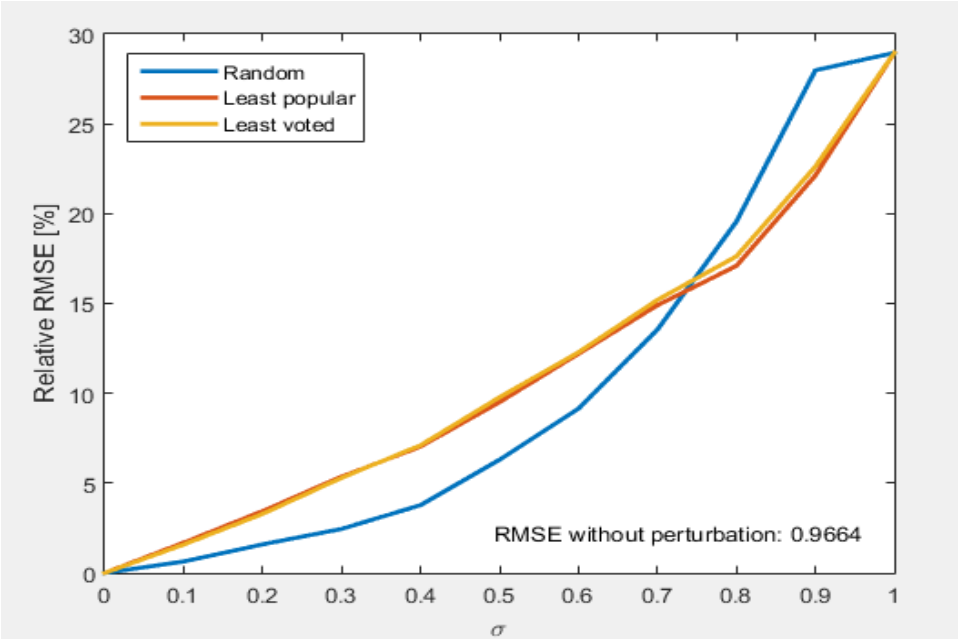


Figura 39. Evolución RMSE relativa vs sigma para OSTA y Userbased

En la figura 39 observamos que la métrica RMSE que es muy similar a MAE, aunque en valores absolutos RMSE es mayor, fijémonos en el valor de RMSE sin perturbar, 0.9664.

Si analizamos las métricas desde el punto de vista de los algoritmos prácticos (PA), figuras 35,36 y 37,vemos que tanto MAE y RMSE relativas tienen un comportamiento parecido.

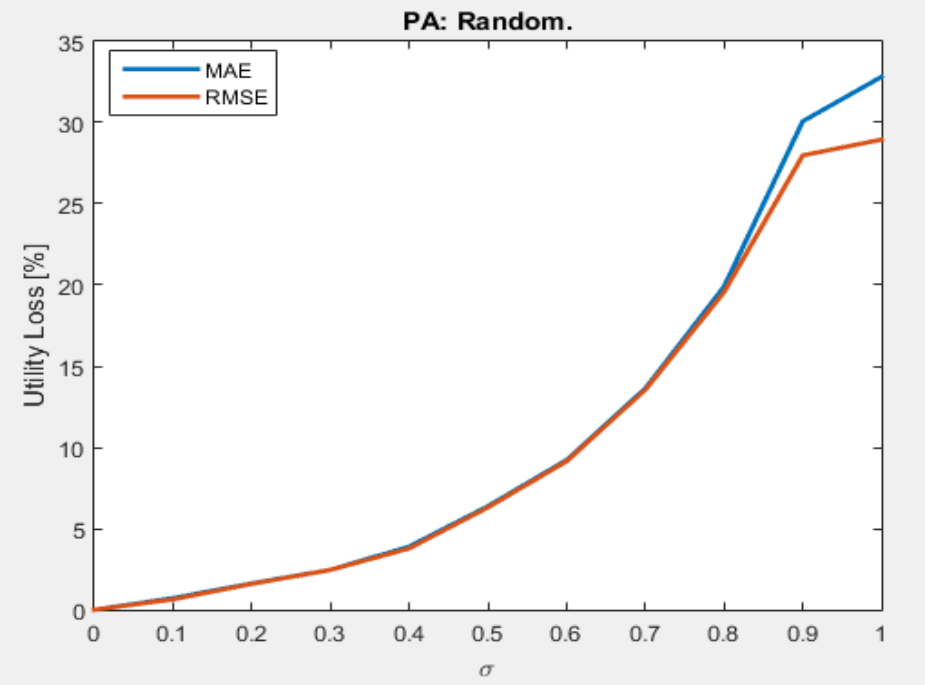


Figura 40. Evolución métricas vs sigma para OSTA, Random y Userbased

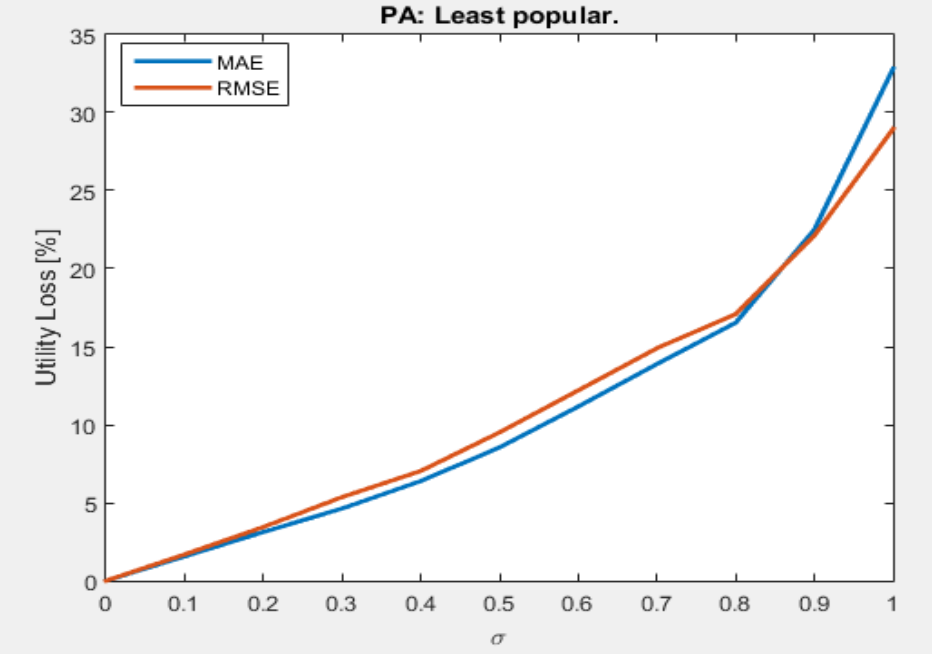


Figura 41. Evolución métricas vs sigma para OSTA, Least Popular y Userbased

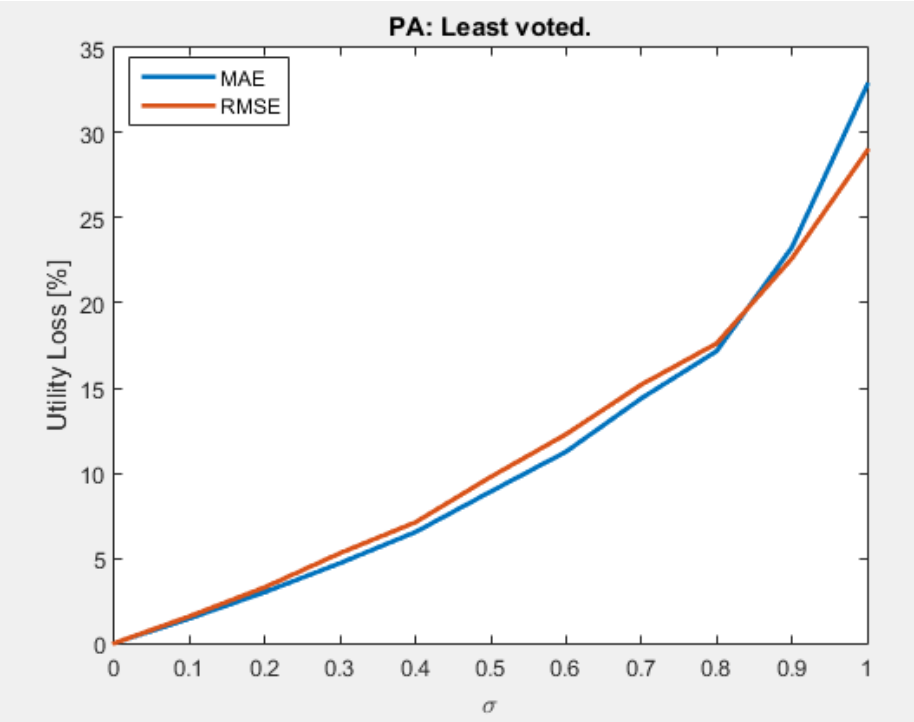


Figura 42 Evolución métricas vs sigma para OSTA, Least voted y Userbased

Para random en forma de exponencial suave creciente y para Least voted y Least Popular más lineal.

En las figuras 43 y 44 vemos los histogramas del error teórico y práctico respectivamente.

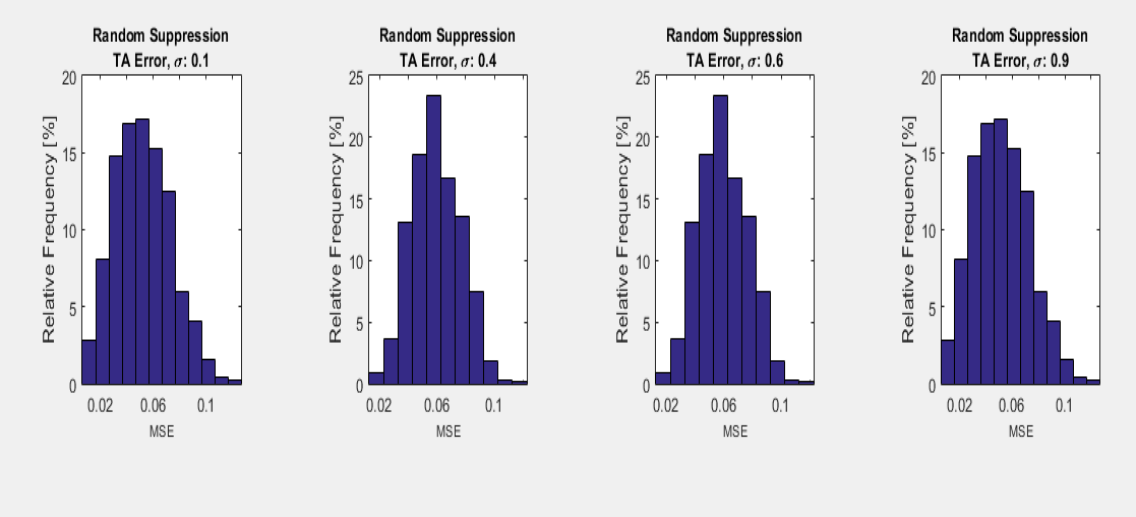


Figura 43 Error teórico para diferentes sigmas para OSTA y Userbased

Vemos que el error teórico es muy pequeño, casi el 100% del error MSE es menos a 0.1

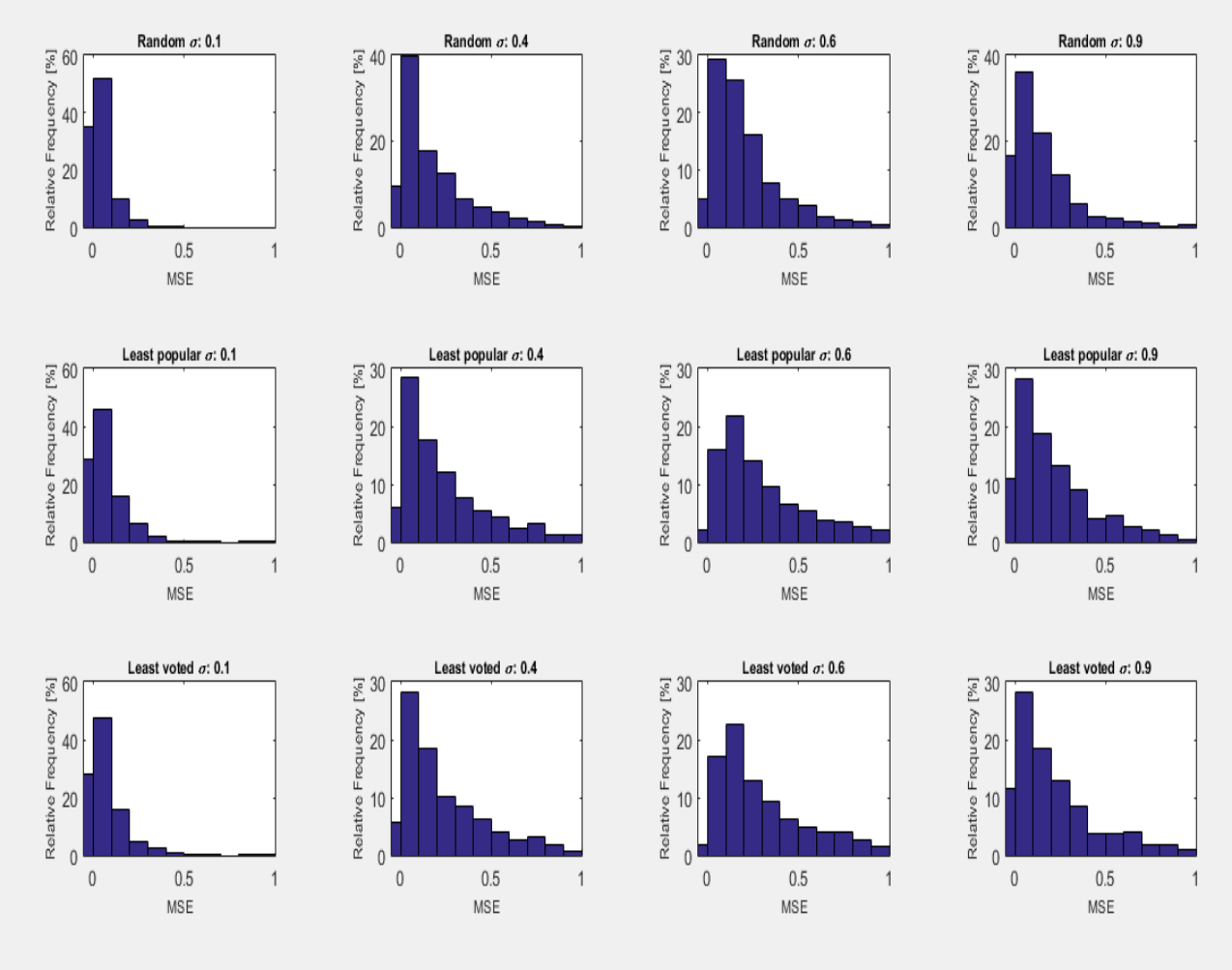


Figura 44 Error práctico para diferentes valores de sigma para OSTA, Random, Least voted, Least popular y Userbased

El error práctico también es muy pequeño como podemos comprobar y observamos cómo crece a medida que aumenta el valor de σ.

En relación al riesgo de privacidad respecto a la pérdida de utilidad, figuras 45,46 y 47, vemos que en este algoritmo teórico (OSTA) el riesgo de privacidad se reduce. Con sólo un 10% de pérdida de utilidad conseguimos un 60% de reducción del riesgo de privacidad relativa.

Las tres gráficas obtenidas son muy parecidas lo que sugiere que no hay grandes diferencias entre usar una estrategia práctica u otra.

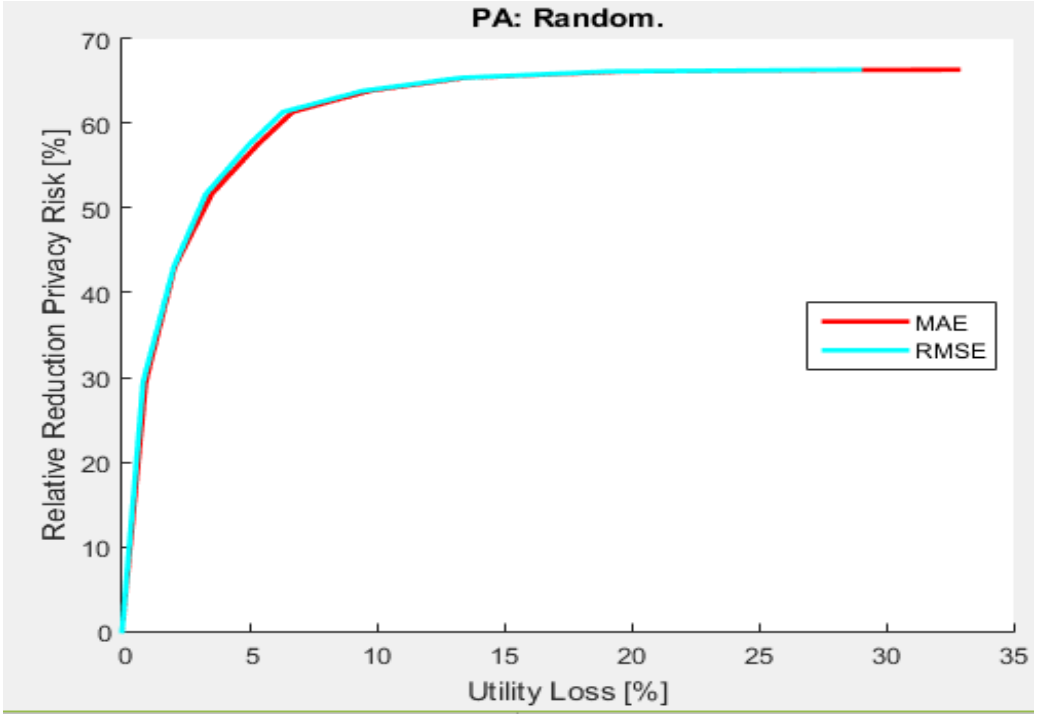


Figura 45 Reducción relativa riesgo privacidad vs pérdida de utilidad para OSTA, Random y Userbased

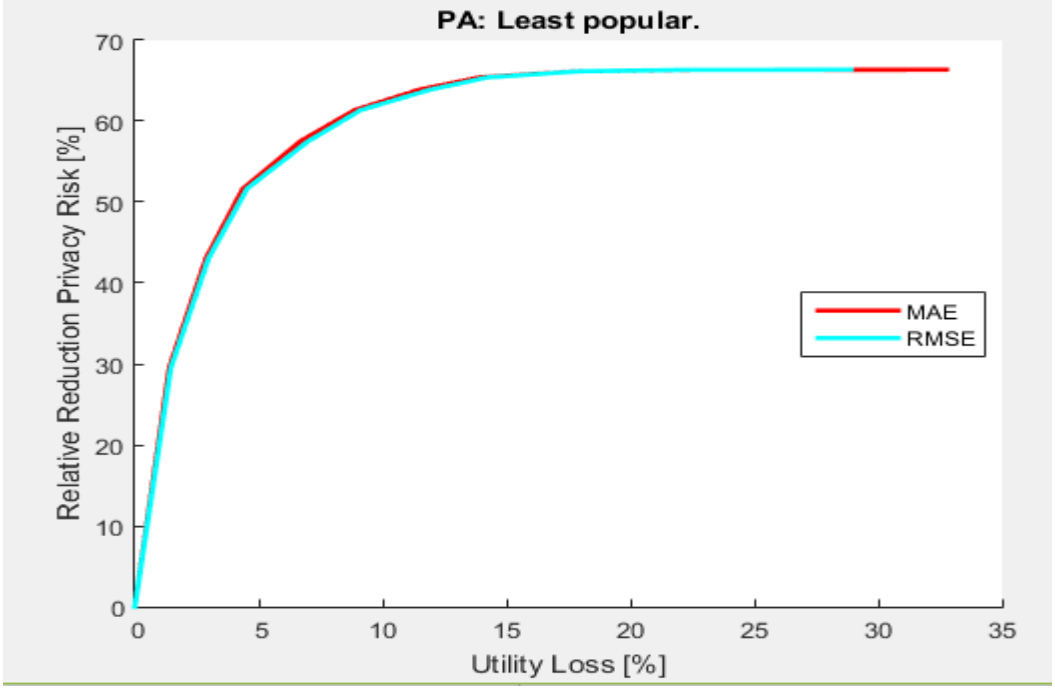


Figura 46 Reducción relativa riesgo privacidad vs pérdida de utilidad para OSTA, Least popular y Userbased

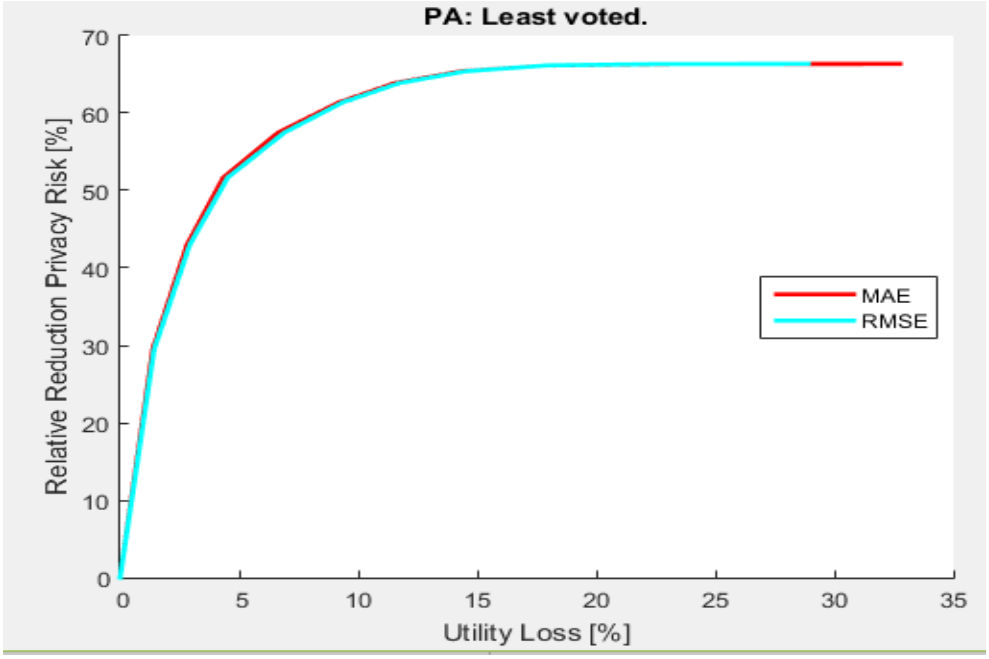


Figura 47 Reducción relativa riesgo privacidad vs pérdida de utilidad para OSTA, Least voted y Userbased

RS: Userbased y TA:RFTA (Random Forgery)

En este caso de algoritmo no optimizado vemos que el error aumenta como mucho un 4% para la estrategia práctica Random y alrededor de un 10% para Most voted y Most popular. Mucho menos que en el caso anterior se Optimized Suppression que alcanzábamos valores del 30% en pérdida de utilidad.

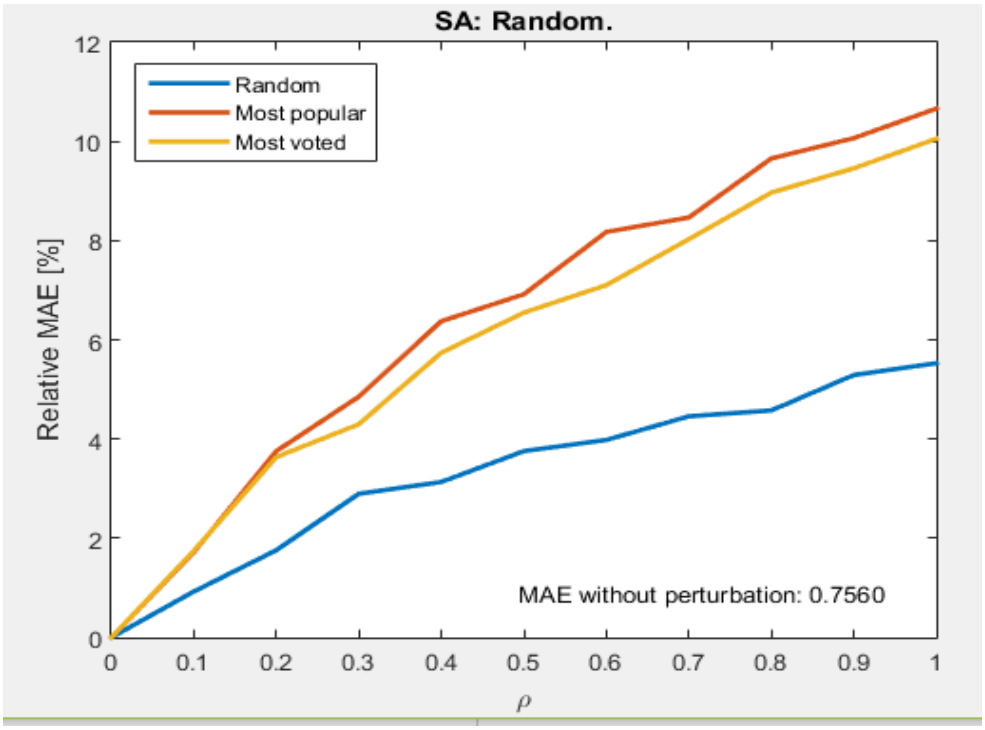


Figura 48. Evolución MAE relativa vs rho para RFTA y Userbased

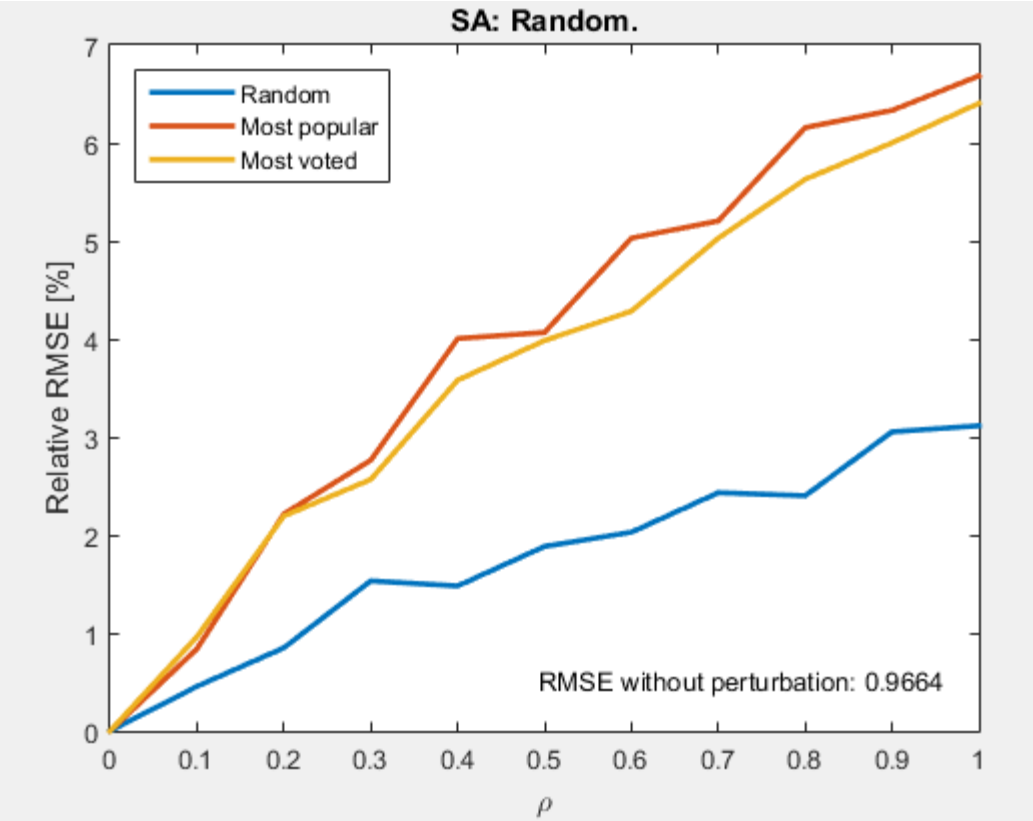


Figura 49. Evolución RMSE relativa vs rho para RFTA y Userbased

La leyenda "SA:Random" significa que al aplicar "forgery" se puntúa de forma aleatoria entre 1 y 5.

Vemos en los dos casos, tanto para MAE como para RMSE, que la estrategia práctica Random es mejor que most popular y most voted.

En la figura 50 tenemos la evolución de las dos métricas, en valor relativo, en relación a la estrategia práctica utilizada, en todos los casos RMSE se comporta mejor que MAE.

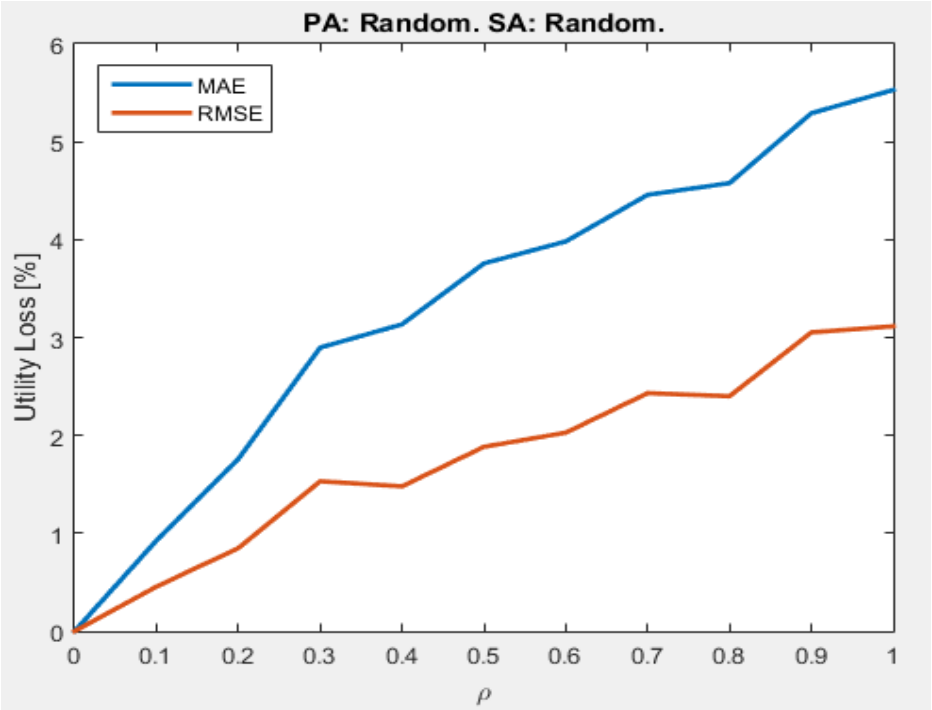


Figura 50. Evolución métricas relativa vs rho para RFTA, Random y Userbased

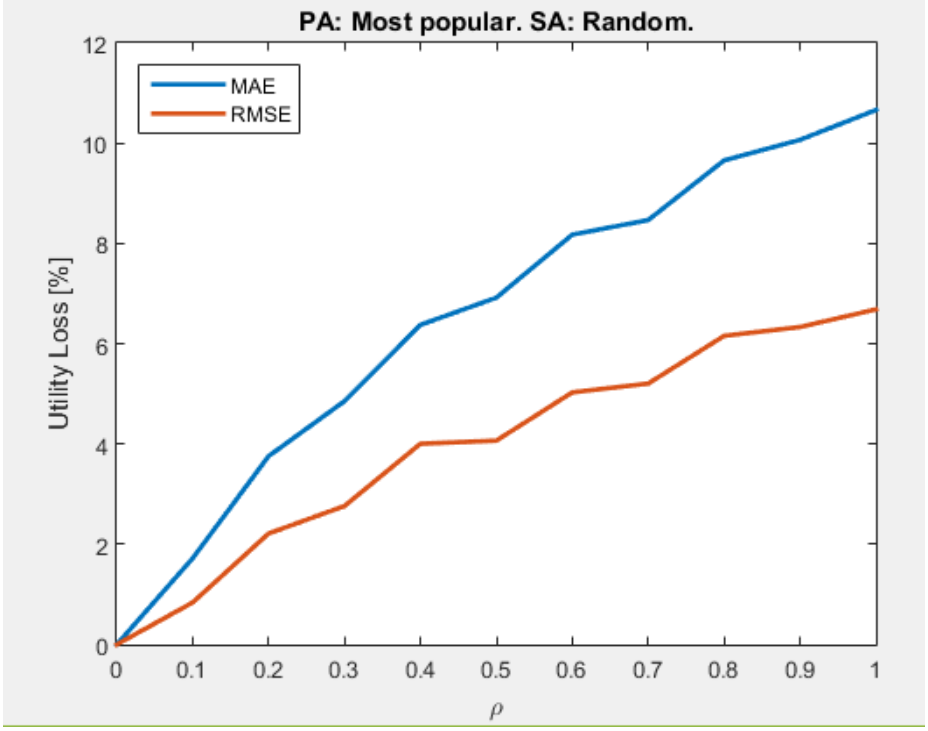


Figura 51. Evolución métricas relativa vs rho para RFTA, Most popular y Userbased

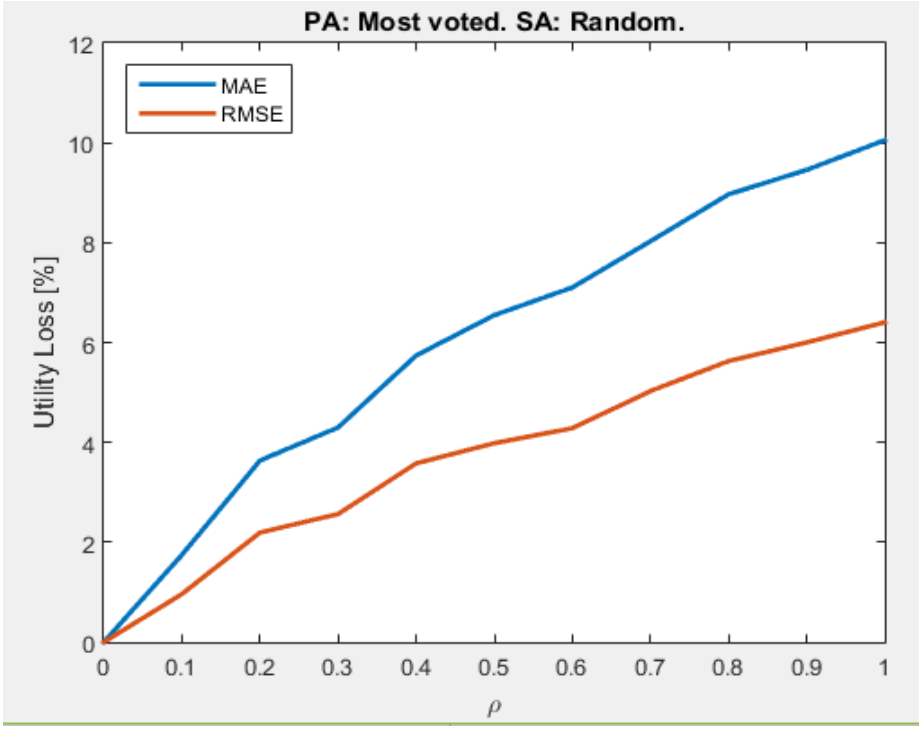


Figura 52 Evolución métricas relativa vs rho para RFTA, Most voted y Userbased

Para los algoritmos teóricos no optimizados que usan solamente "forgery" el riesgo de privacidad va aumentando a medida que crece la pérdida de utilidad como vemos en las figuras 53,54 y 55.

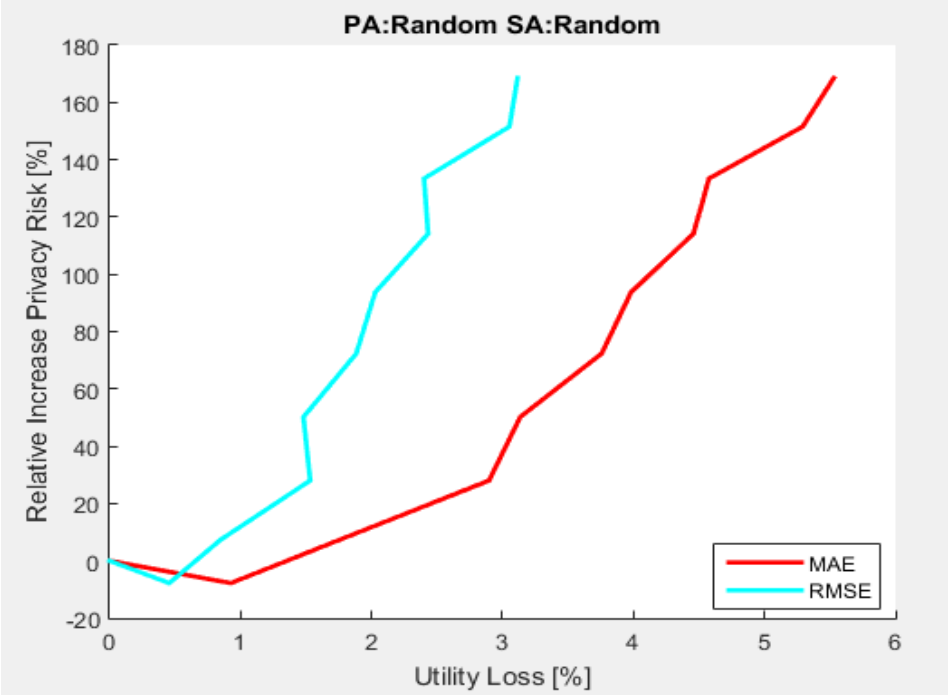


Figura 53 Incremento relativo riesgo privacidad vs pérdida de utilidad para RFTA, Random y Userbased

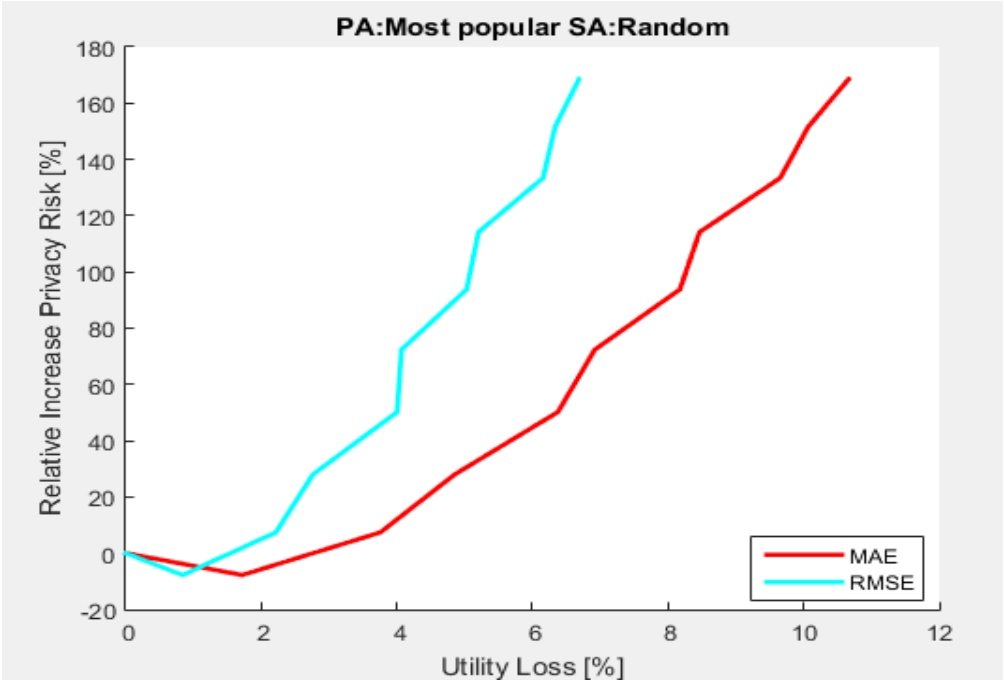


Figura 54 Incremento relativo riesgo privacidad vs pérdida de utilidad para RFTA, Most popular y Userbased

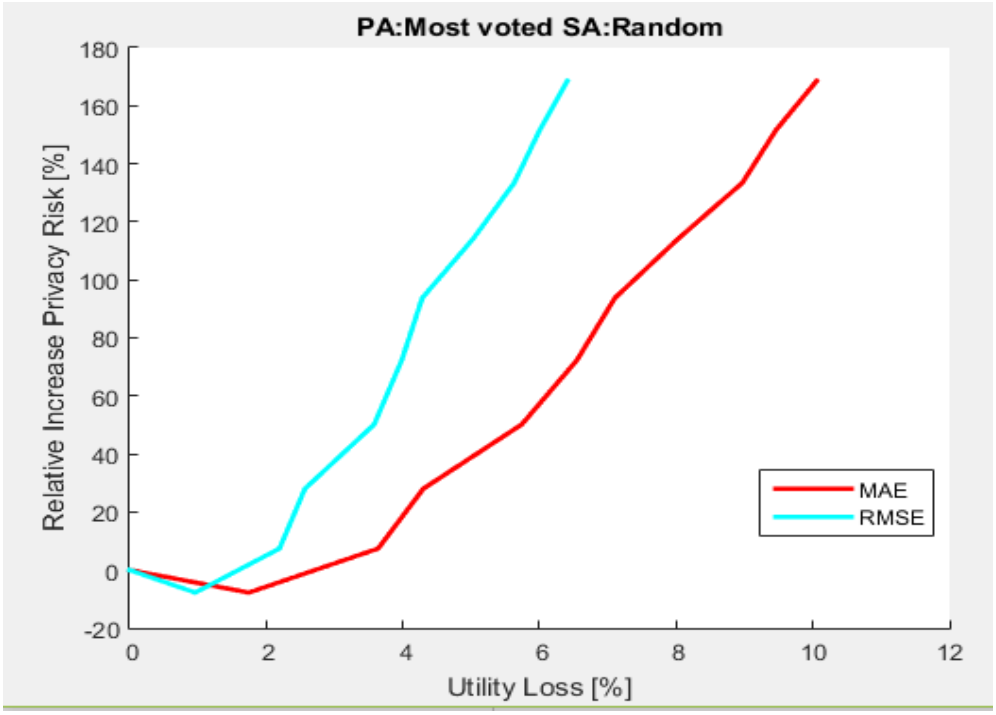


Figura 55 Incremento relativo riesgo privacidad vs pérdida de utilidad para RFTA, Most voted y Userbased

Se observa en las gráficas anteriores una tendencia clara a un incremento elevado del riesgo de privacidad con un 8% de pérdida de utilidad aumenta el riesgo de privacidad más del 100%.

RS: Userbased y TA:OFTA (Optimized Forgery)

En este caso seleccionamos el algoritmo teórico que aplica solo "forgery" de forma optimizada. En las figuras 56 y 57 vemos el comportamiento de las métricas MAE y RMSE relativas, como en los casos anteriores el error aumenta con el incremento de ρ. En este caso también se comporta mejor el algoritmo práctico Random que el resto de métodos prácticos.

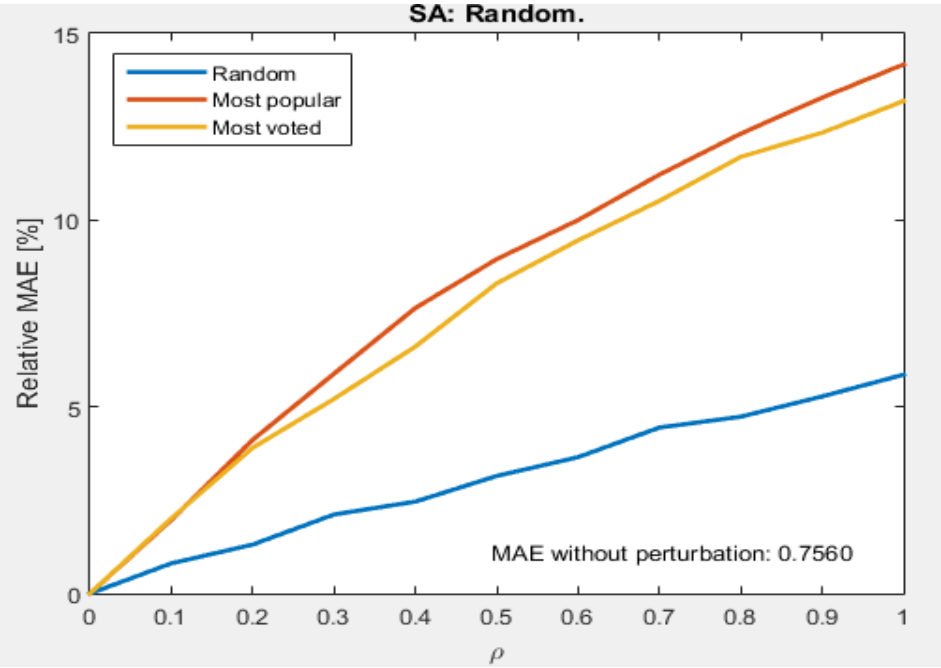


Figura 56 Evolución MAE relativa vs rho para OFTA y Userbased

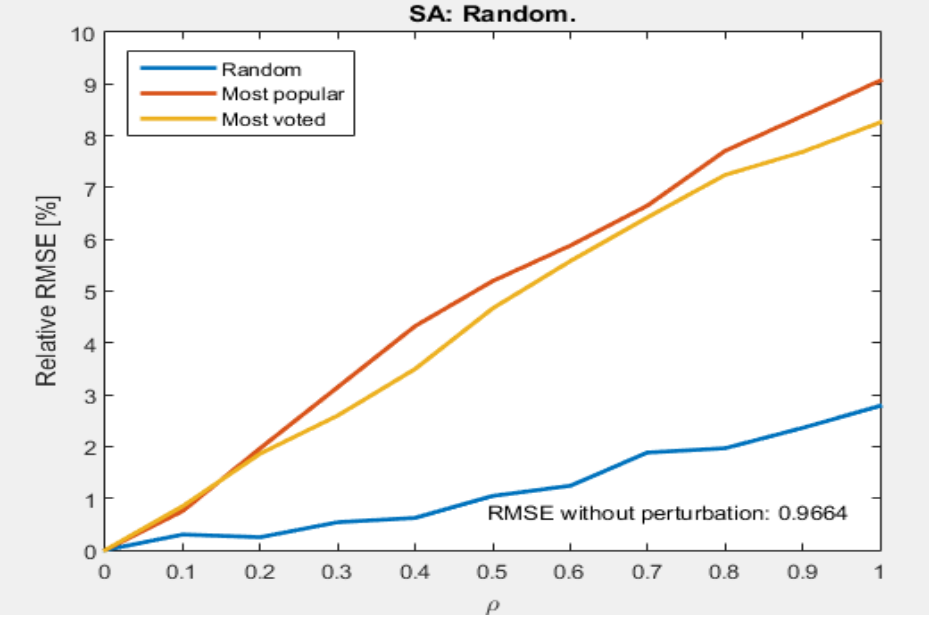


Figura 57 Evolución RMSE relativa vs rho para OFTA y Userbased

Si miramos las métricas relativas por tipo de algoritmo práctico vemos que en los tres casos el comportamiento es muy similar y que RMSE relativo tiene

mejor comportamiento que MAE relativo. En la figura 58 vemos un ejemplo de lo que decimos para PA random.

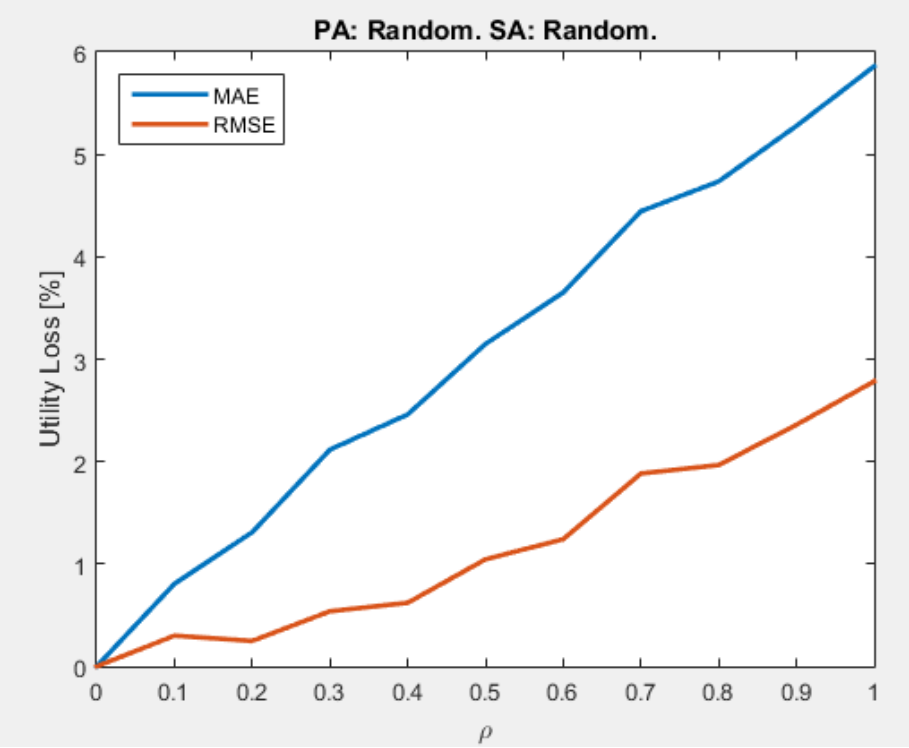


Figura 518 Evolución métricas relativa vs rho para OFTA, Random y Userbased

Con este tipo de algoritmo teórico OFTA tenemos que el riesgo de privacidad disminuye rápidamente con la pérdida de utilidad, figuras 59, 60 y 61.

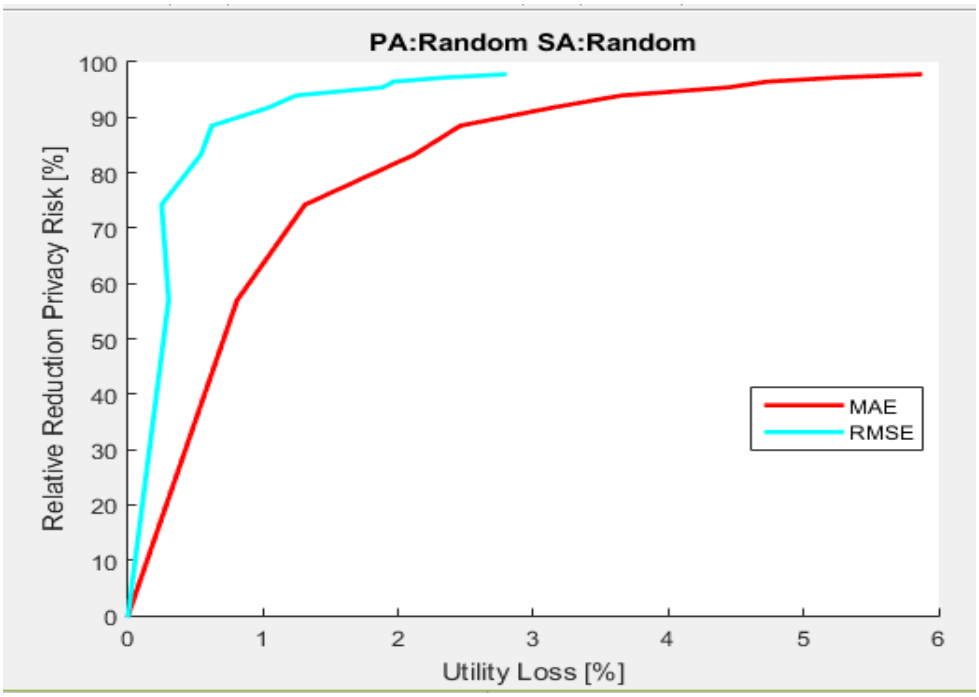


Figura 59. Reducción relativa riesgo privacidad vs pérdida de utilidad para OFTA,Random y Userbased

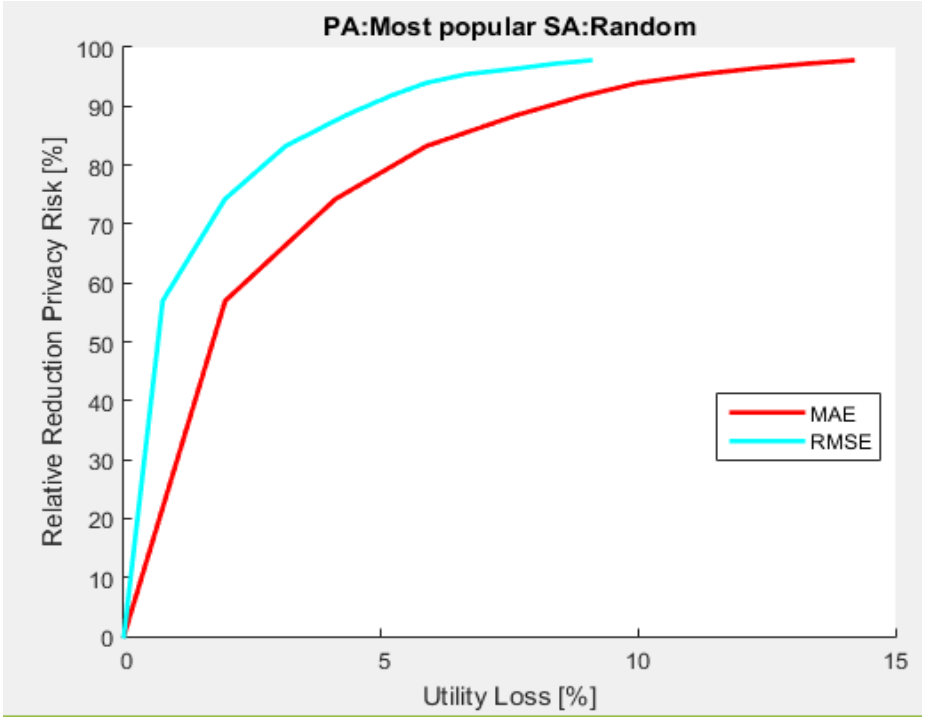


Figura 60 Reducción relativa riesgo privacidad vs pérdida de utilidad para OFTA, Most popular y Userbased

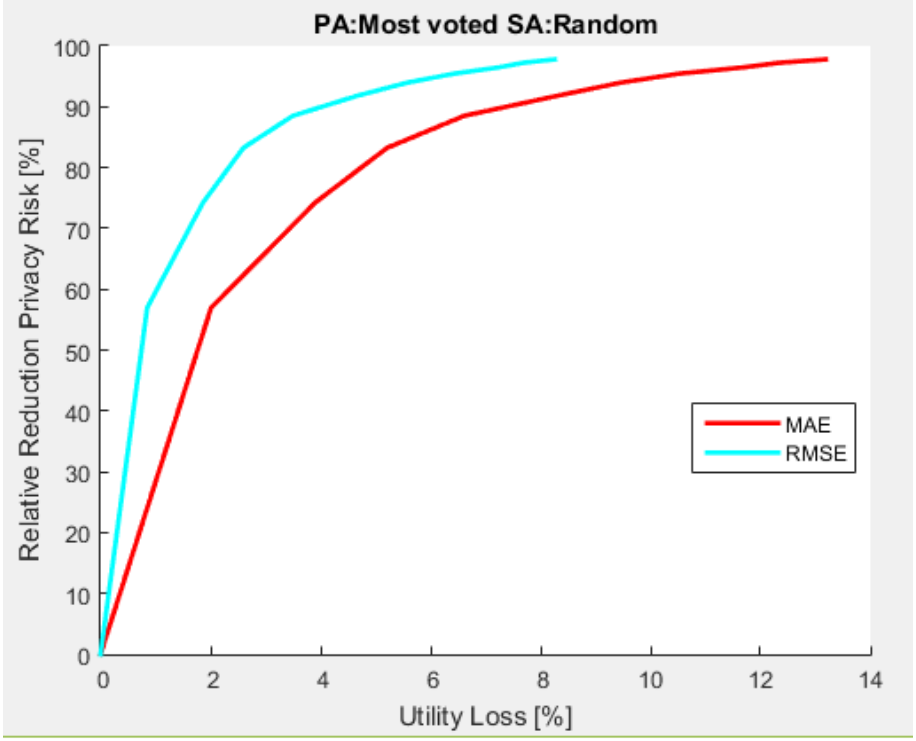


Figura 61. Reducción relativa riesgo privacidad vs pérdida de utilidad para OFTA,most voted y Userbased

Vemos en los tres casos que RMSE es más rápido que MAE y que reduce el riesgo de privacidad con mayor velocidad. De los tres algoritmos prácticos el más rápido es Random.

RS: Userbased y TA:OFSTA (Optimized Forgery Suppression)

Para este algoritmo teórico que permite trabajar tanto con valores de σ y de ρ tenemos gráficas en 3D.Los resultados son muy similares a los obtenidos cuando usamos los algoritmos teóricos optimizados (OFTA y OSTA). En la figura 62 comprobamos que al aumentar σ y ρ los valores de MAE relativa van aumentando también.

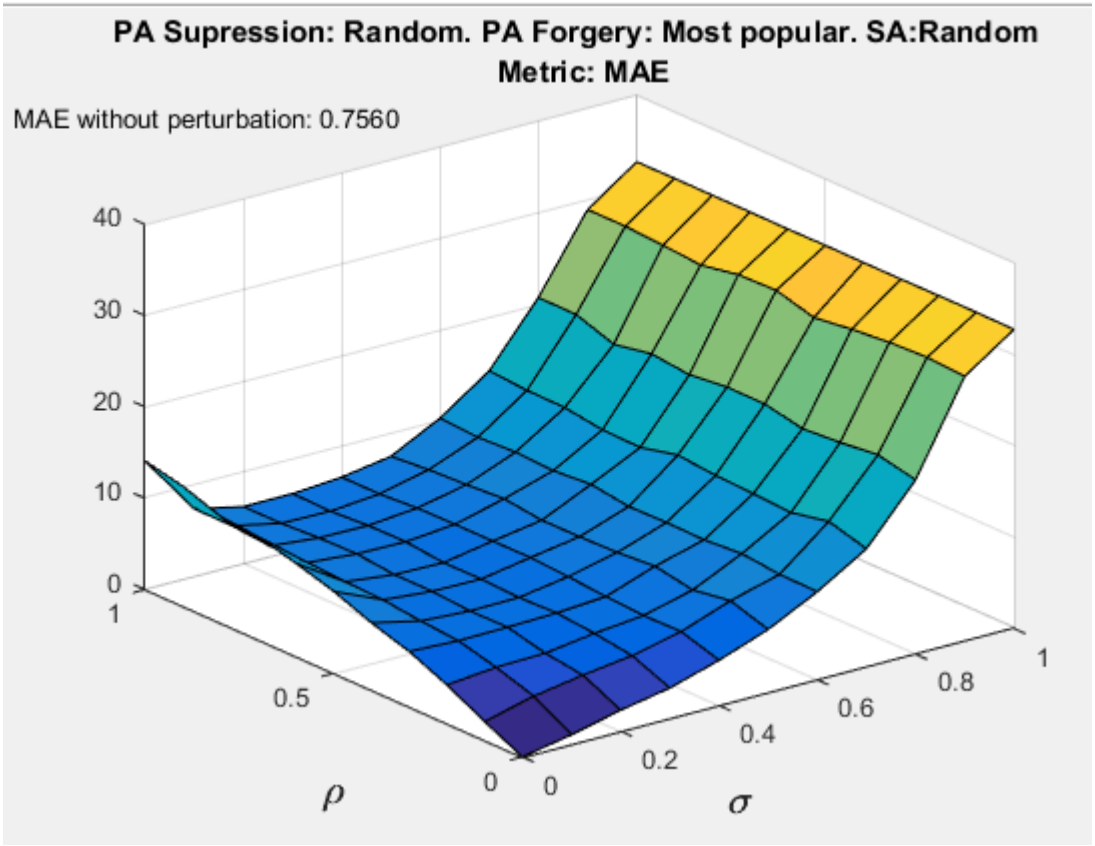


Figura 62.Evolución MAE relativa vs sigma y rho para OFSTA y Userbased

RS: Itembased y TA:OSTA (Optimized Suppression)

Ahora cambiamos de sistema de recomendación y usamos itembased y vamos a ver el caso de optimized suppression. Vemos que cuando usamos este tipo de algoritmo teórico el error máximo en las métricas MAE y RMSE puede llegar hasta el 30%.

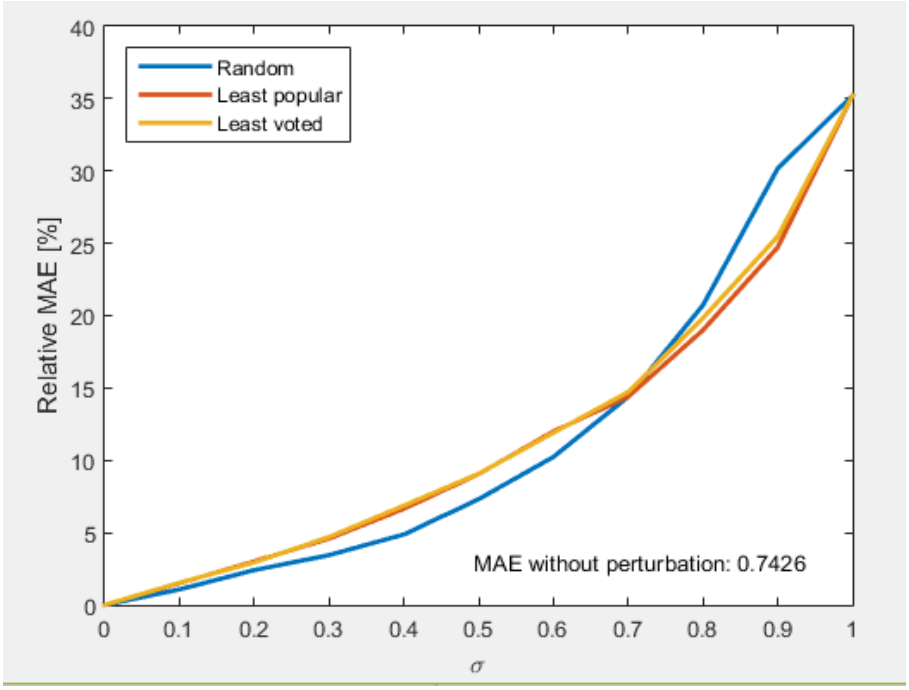


Figura 63 Evolución MAE relativa vs sigma para OSTA e Itembased

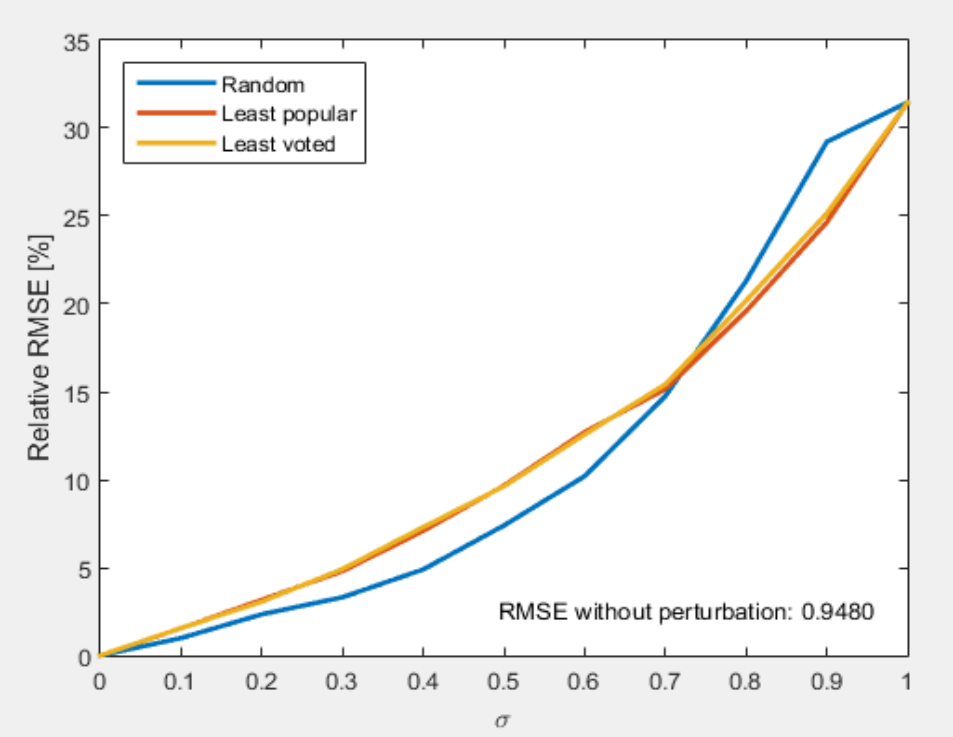


Figura 64 Evolución RMSE relativa vs sigma para OSTA e Itembased

Las métricas relativas MAE y RMSE vistas por el método práctico utilizado, figuras 65,66 y 67 tienen un comportamiento muy parecido y sus curvas casi se superponen.

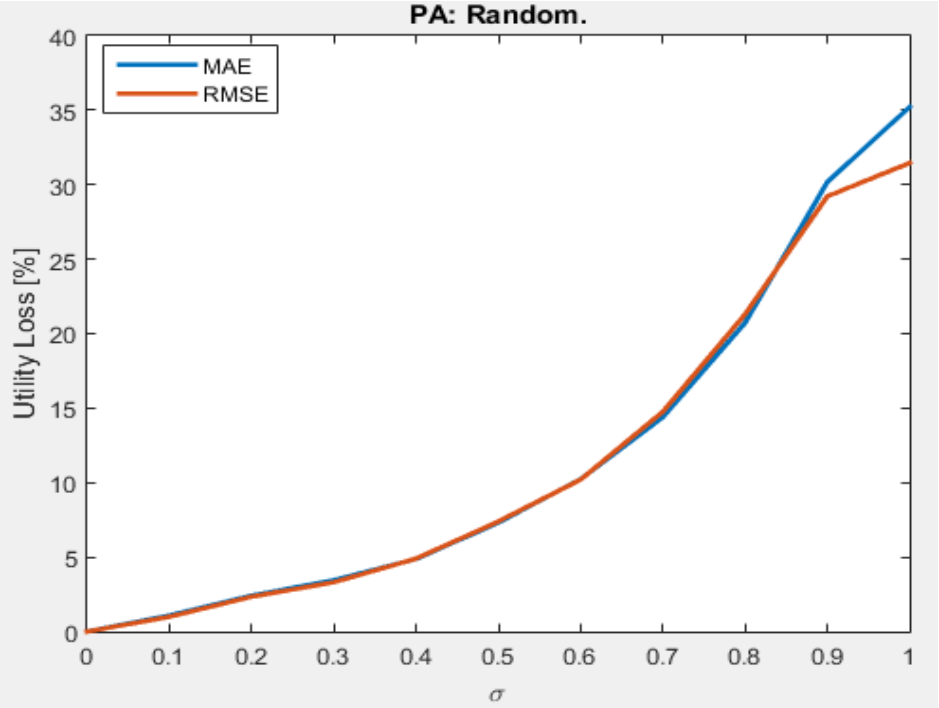


Figura 65. Evolución métricas relativas vs sigma para OSTA, Random e Itembased

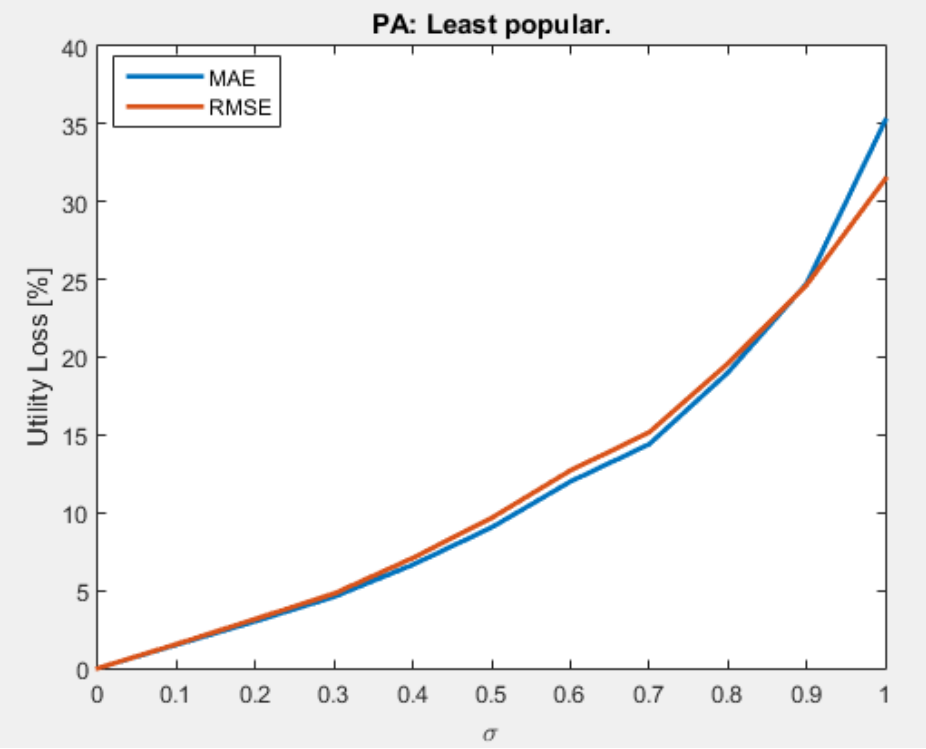


Figura 66. Evolución métricas relativas vs sigma para OSTA, Least Popular e Itembased

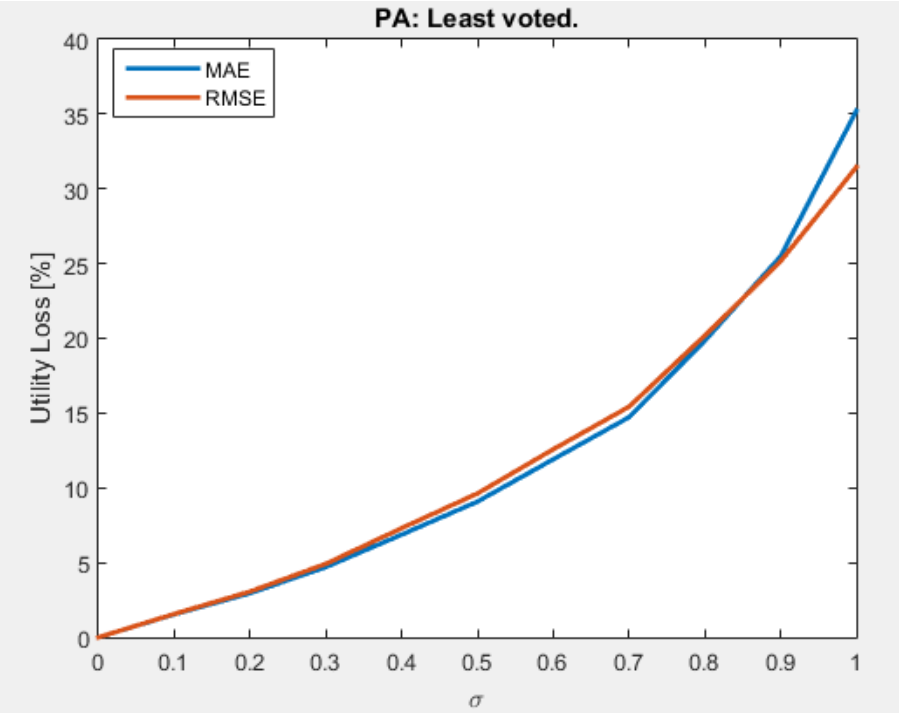


Figura 67. Evolución métricas relativas vs sigma para OSTA, Least voted e Itembased

En las siguiente figura 68 y 69 visualizamos el error teórico y práctico que se producen en los algoritmos teóricos y prácticos.

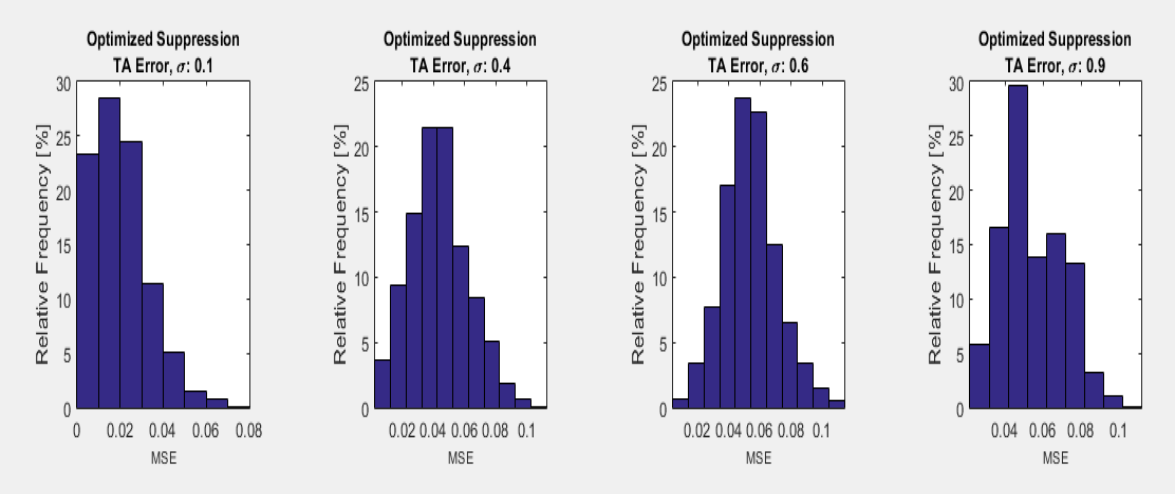


Figura 68 Error teórico para diferentes sigmas para OSTA e Itembased

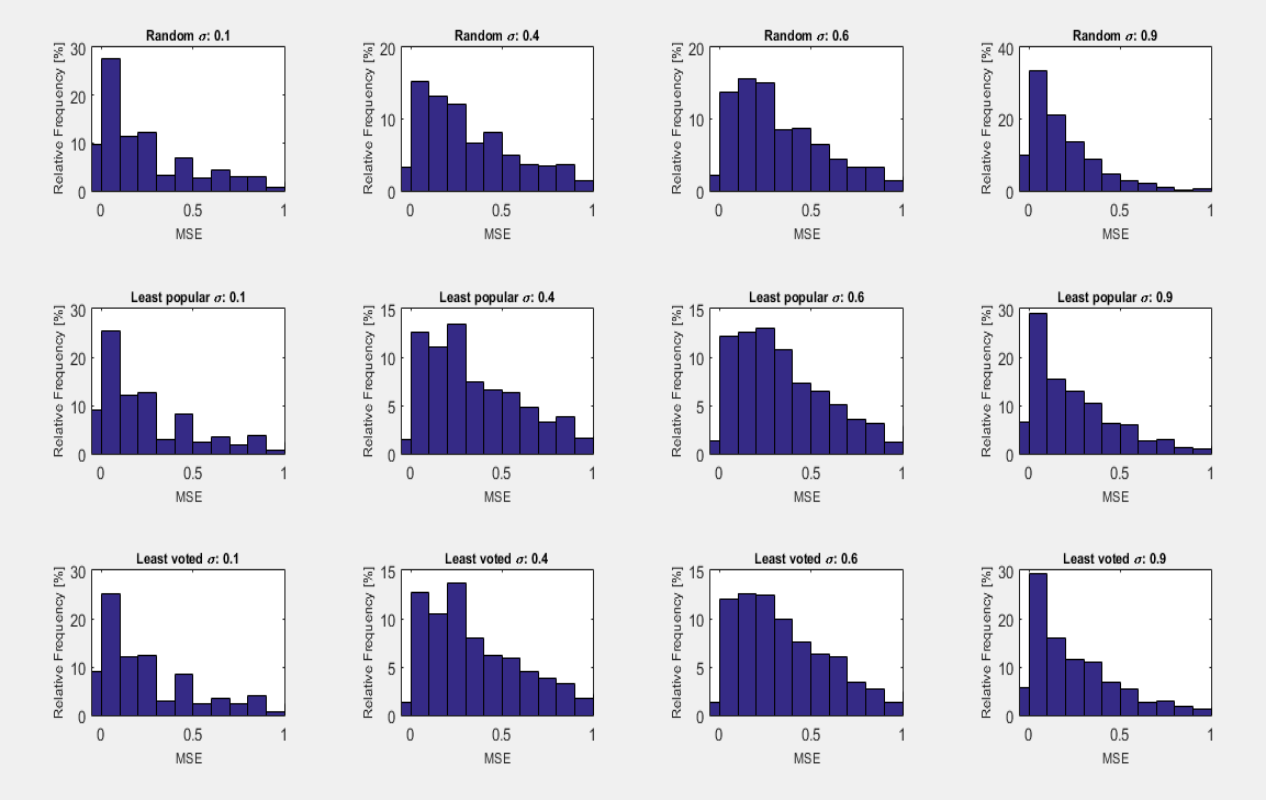


Figura 69 Error práctico para diferentes sigmas para OSTA e Itembased

Al ser un algoritmo teórico optimizado tenemos reducción relativa del riesgo de privacidad respecto a la pérdida de utilidad. Los tres métodos prácticos tienen resultados bastantes similares, en la figura 70 presentamos el caso de least popular.

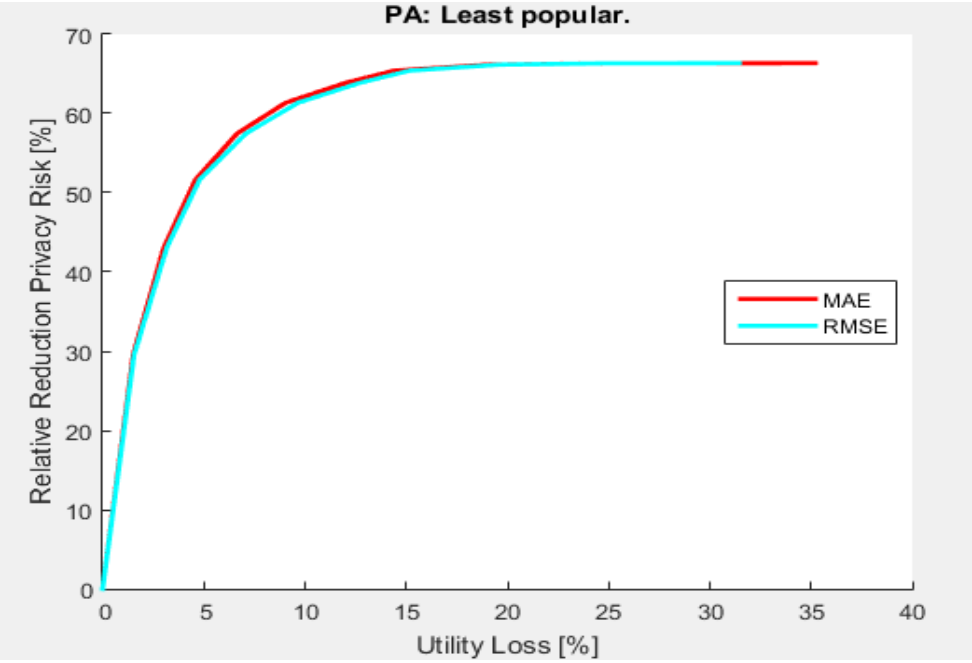


Figura 70. Reducción relativa riesgo privacidad vs pérdida de utilidad para OSTA, least popular e itembased

Vemos que con un 20% de pérdida de utilidad llegamos al 65% de reducción de privacidad y que tanto MAE como RMSE tienen un comportamiento casi idéntico.

RS: Regsvd y TA:RSTA (Random Suppression)

Ahora usamos un sistema de recomendación, Regsvd, basado en modelo, que utiliza la factorización para predecir valores. En primer lugar, como en los casos anteriores, hacemos la gráfica para las métricas MAE y RMSE relativas en relación al aumento de σ.

Vemos que cuando usamos Random las métricas no crecen tanto que en los otros dos métodos pero a partir de σ=0.9 los valores de MAE y RMSE se disparan.

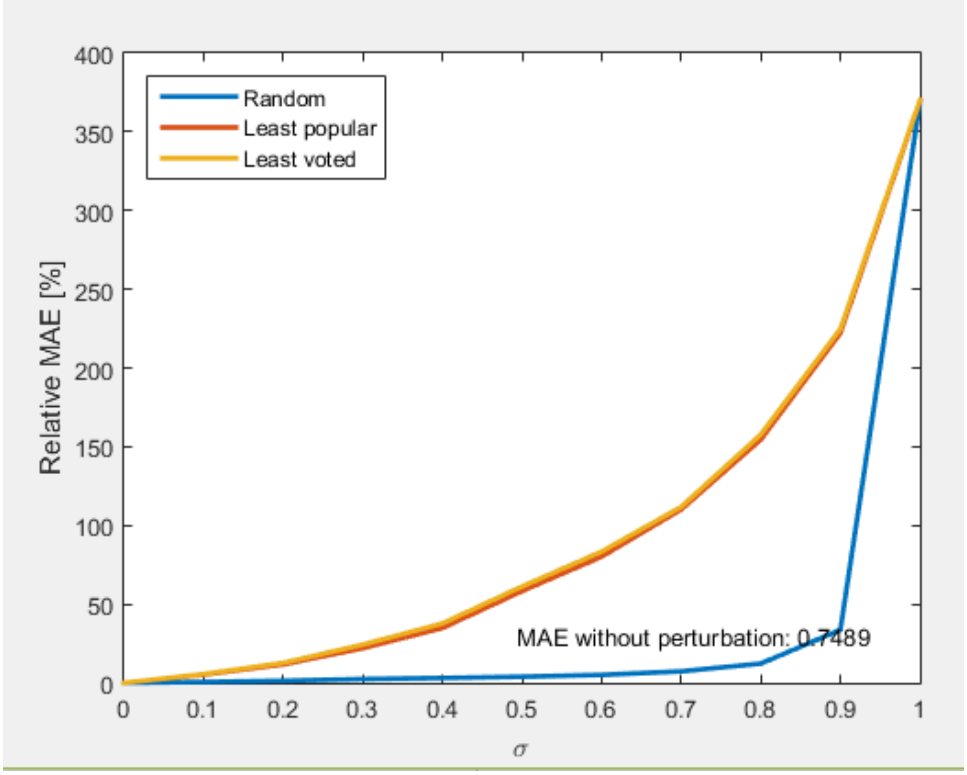


Figura 71 Evolución MAE relativa vs sigma para RSTA y regsvd

En las figuras 73, 74 y 75 observamos la evolución de MAE y RMSE relativas según el algoritmo práctico utilizado. Las dos métricas tienen un comportamiento similar pero hay que destacar que mientras en Least popular y Least voted el error aumenta de forma exponencial en Random evoluciona casi lineal hasta llegar a σ=0.9 donde se dispara.

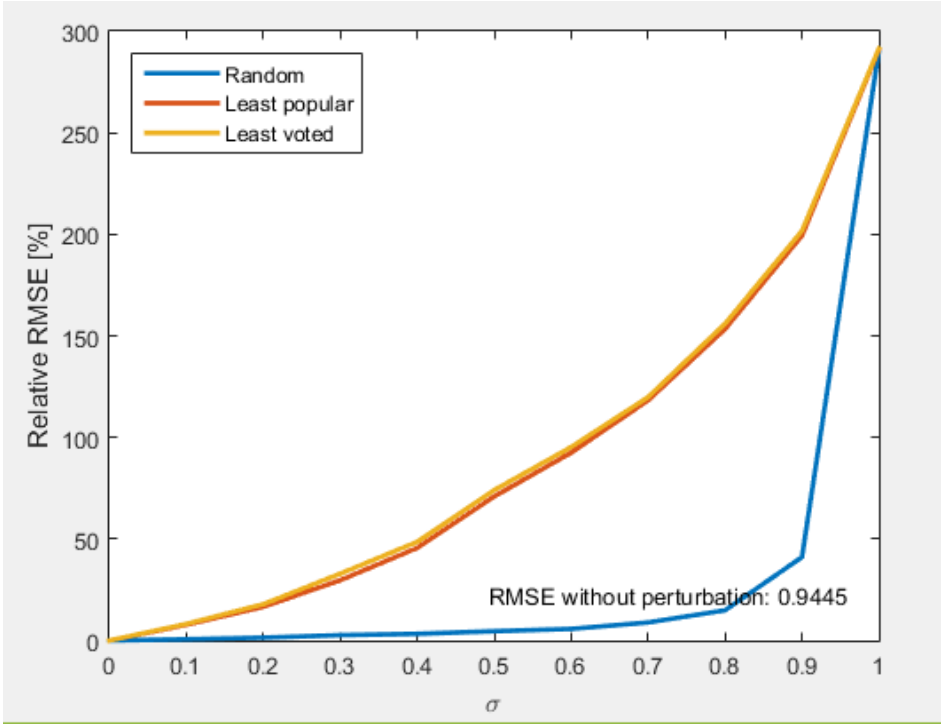


Figura 72 Evolución RMSE relativa vs sigma para RSTA y regsvd

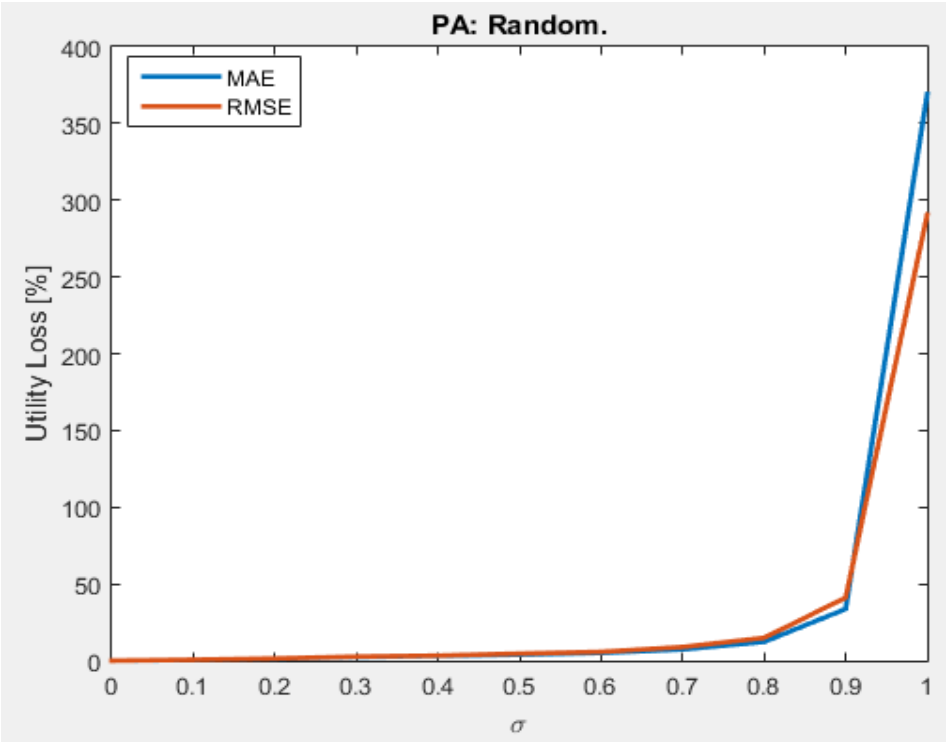


Figura 73. Evolución métricas relativas vs sigma para RSTA, Random y regsvd

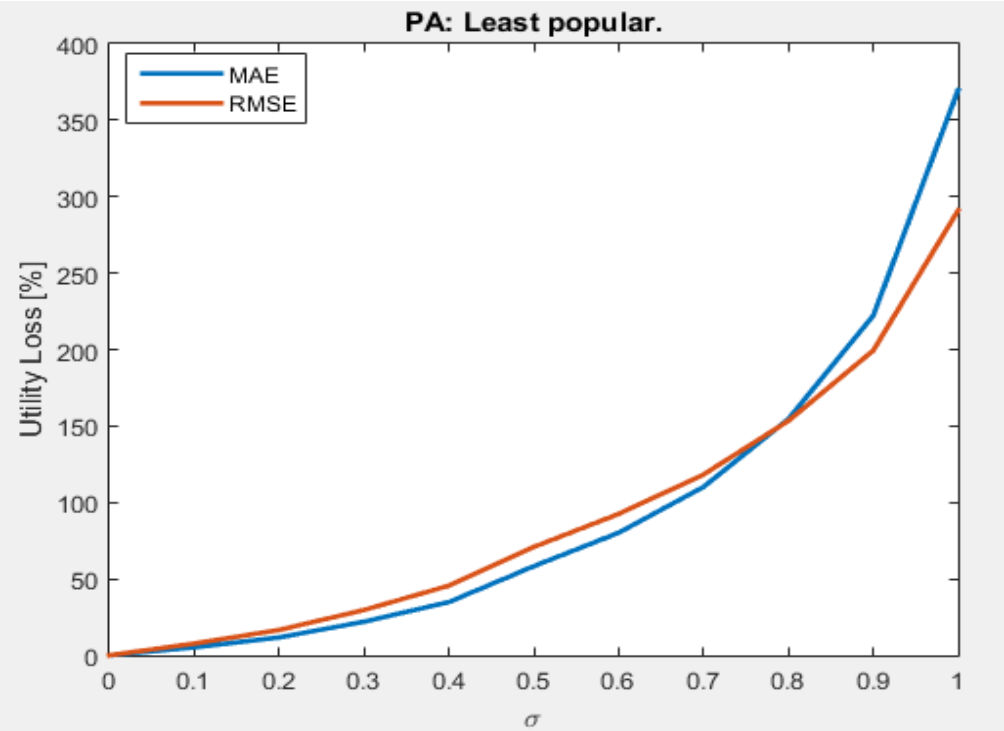


Figura 74. Evolución métricas relativas vs sigma para RSTA, Least popular y regsvd

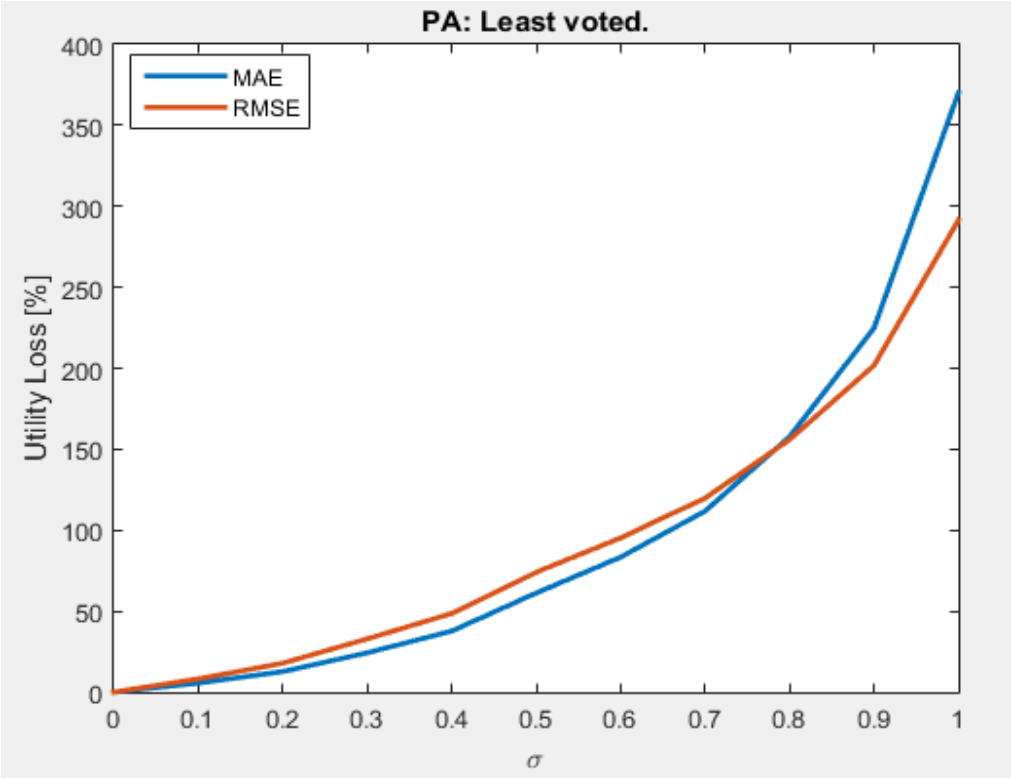


Figura 75. Evolución métricas relativas vs sigma para RSTA, Least voted y regsvd

Por último podemos ver que el error teórico y práctico tienen una evolución muy similar que en el resto de casos estudiados.

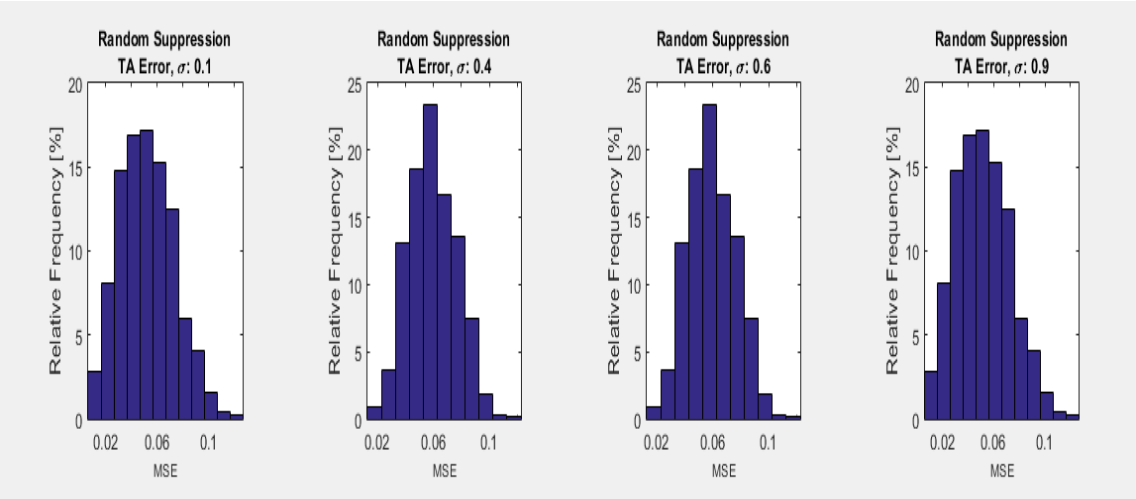


Figura 76 Error teórico para diferentes sigmas para RSTA y regsvd

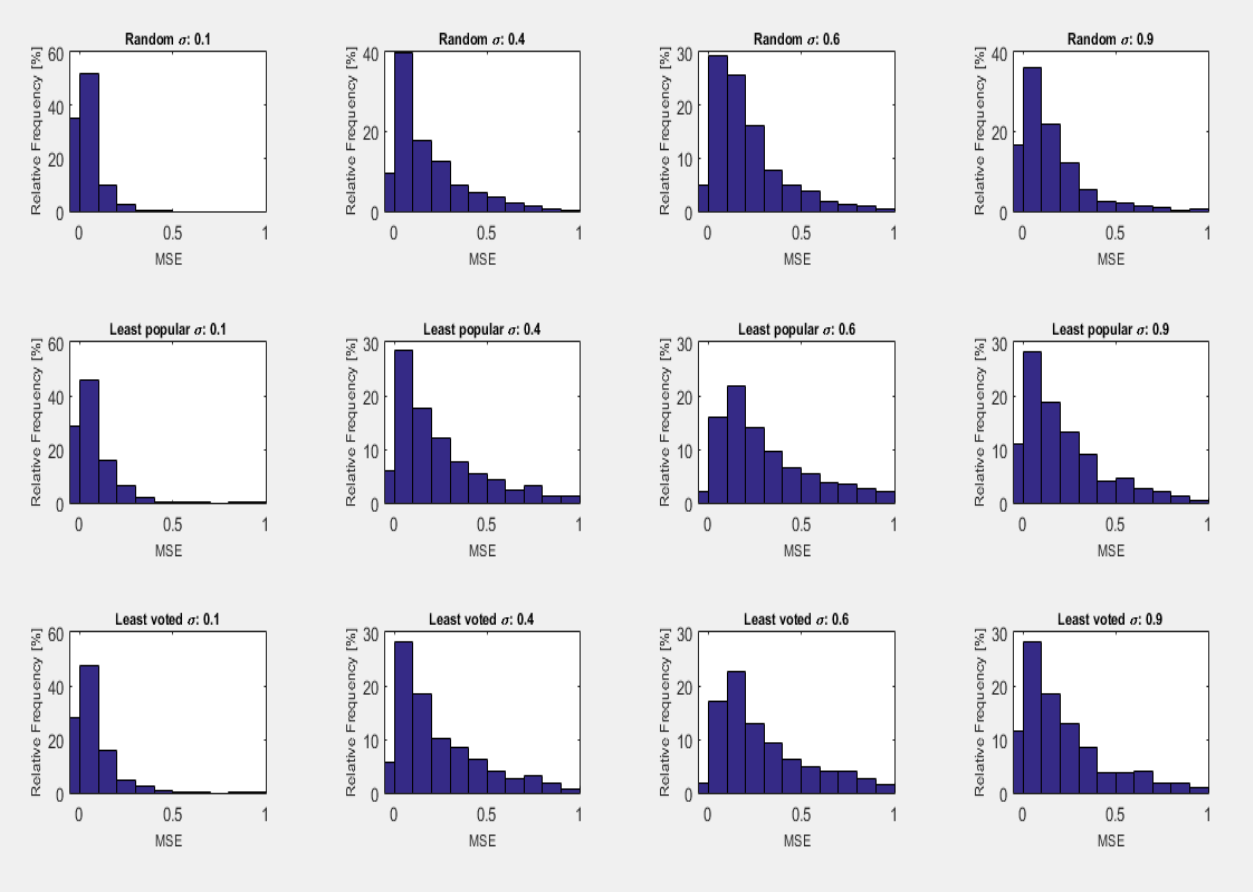


Figura 77 Error práctico para diferentes sigmas para RSTA y regsvd

RS: NMF y TA:OSTA (Optimized Suppression)

NMF es otro sistema de recomendación que utiliza factorización y vemos que tiene un comportamiento similar a Regsvd, aunque ahora las métricas MAE y RMSE relativas se comportan igual en los tres casos, a partir de σ=0.9 se dispara el error en la medida.

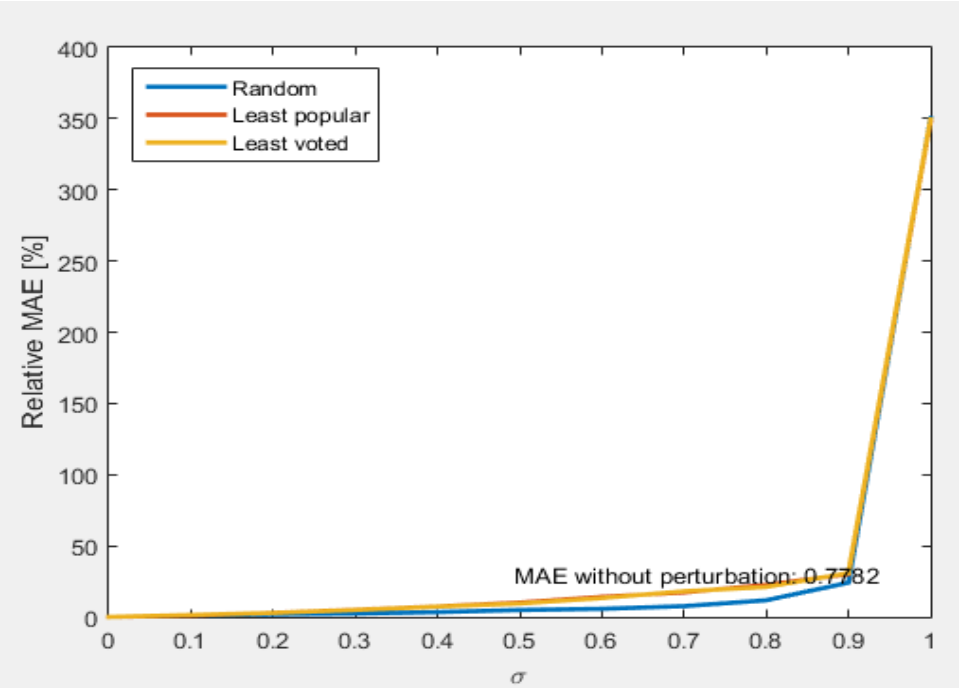


Figura 78 Evolución MAE relativa vs sigma para OSTA y nmf

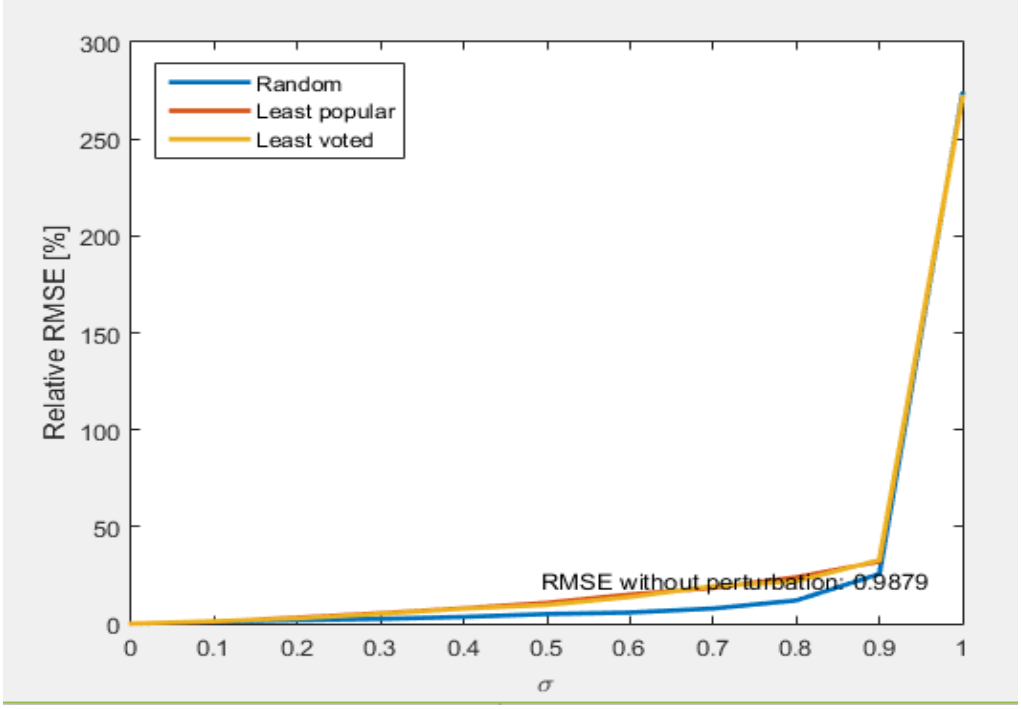


Figura 79 Evolución RMSE relativa vs sigma para OSTA y nmf

Si miramos el comportamiento del riesgo de privacidad vs la pérdida de utilidad vemos que la reducción relativa del riesgo de privacidad aumenta rápidamente para las tres estrategias prácticas. En menos del 10% de pérdida de utilidad tenemos ya un 65% de reducción relativa del riesgo de privacidad. Por último, MAE y RMSE tienen un comportamiento similar.

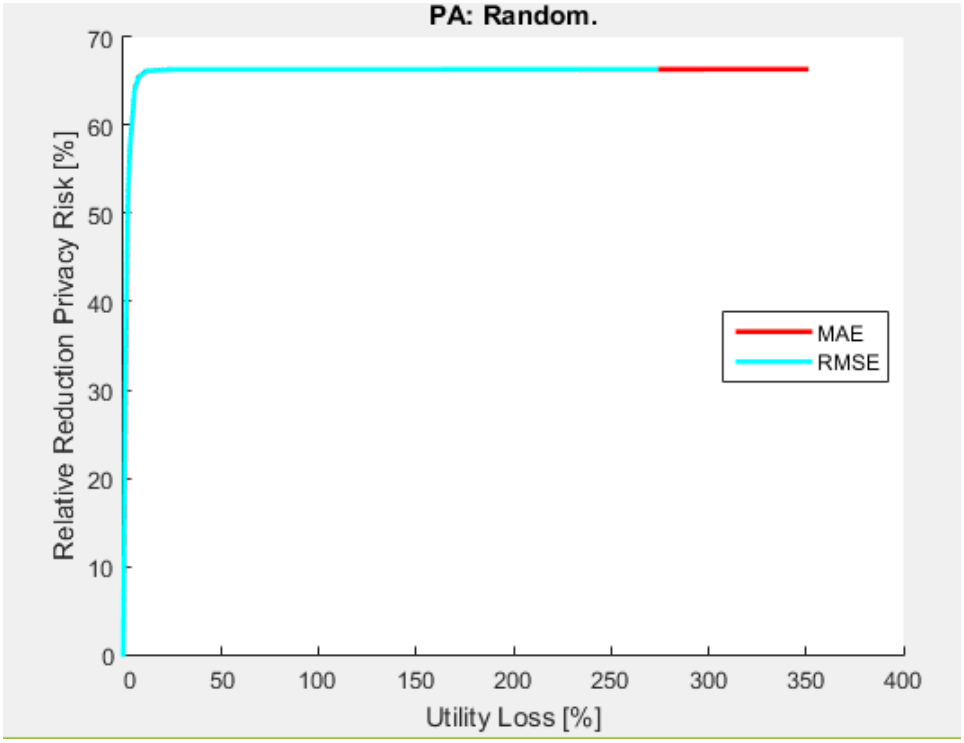


Figura 80. Reducción relativa riesgo privacidad vs pérdida de utilidad para OSTA, Random y nmf

RS: PMF y TA:OSTA (Optimized Forgery)

Aplicaremos a este sistemas de recomendación, PMF, el algoritmo teórico Optimized Forgery. Vemos en la evolución de las métricas que la variación máxima que se produce es del 3%. Como en la mayoría de los casos observados, Random tiene mejor comportamiento que Most popular o Most voted.

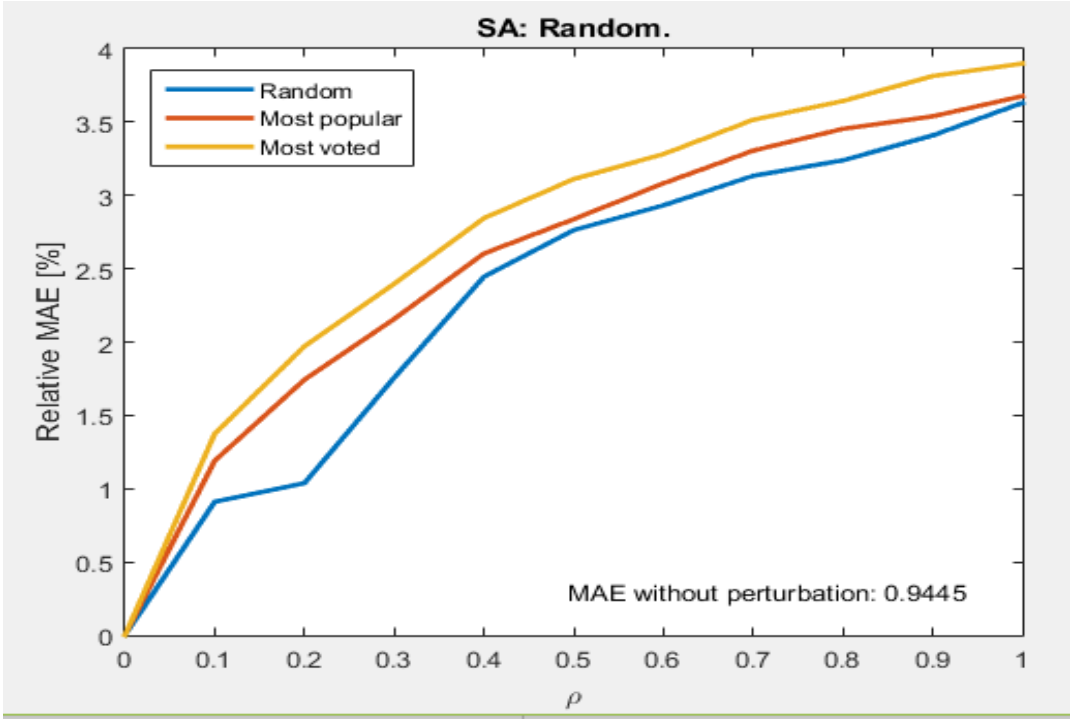


Figura 81 Evolución MAE relativa vs rho para OFTA y pmf

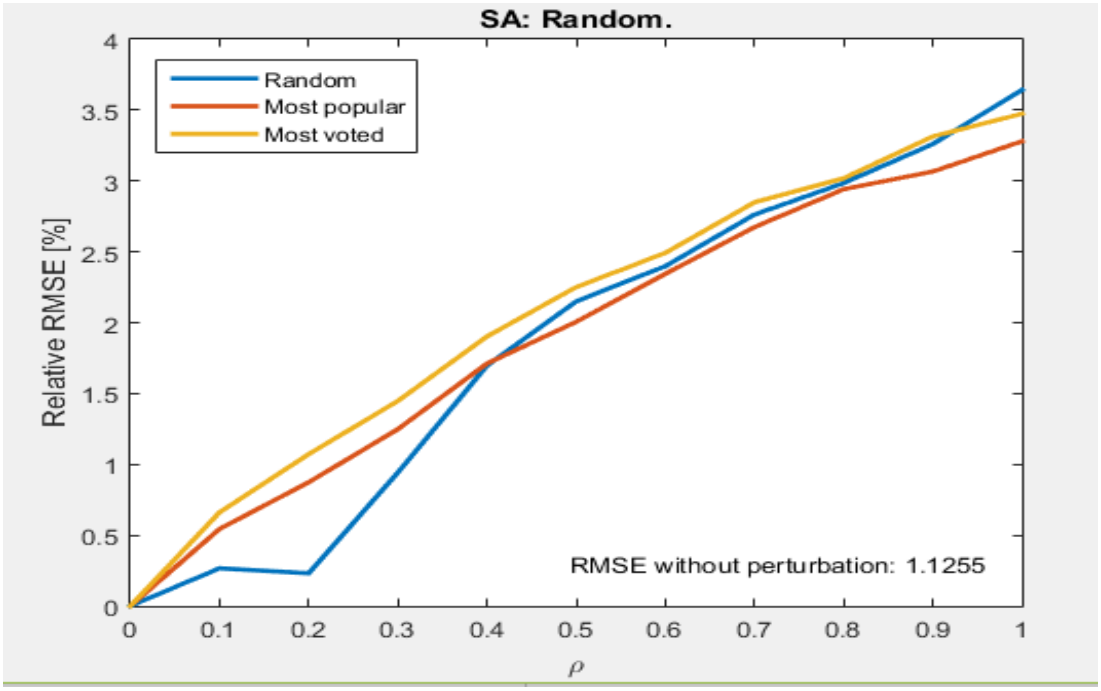


Figura 82 Evolución RMSE relativa vs rho para OFTA y pmf

El comportamiento de la reducción relativa del riesgo de privacidad es similar para los tres algoritmos prácticos. En la figura 83 visualizamos el caso de Most voted, en esta gráfica vemos que RMSE es más rápido que MAE. Para un 4% de pérdida de utilidad tenemos ya un 100% de reducción relativa de riesgo de privacidad en las dos métricas.

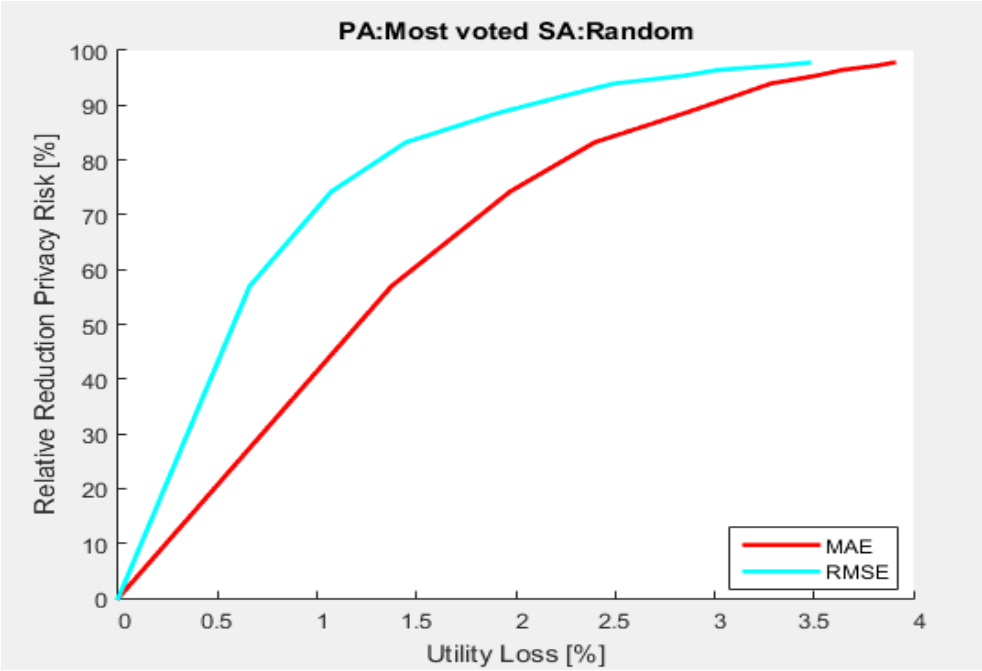


Figura 83. Reducción relativa riesgo privacidad vs pérdida de utilidad para OFTA, most voted y pmf

4.2 Dataset yahoo.

RS: Userbased y TA:RFTA (Random Forgery)

Ahora utilizamos el dataset de yahoo que trabaja con canciones. El análisis de la evolución de las métricas MAE y RMSE relativas nos indica que como máximo hay un aumento del error del 1.5% cuando usamos random y algo superior al 2% cuando usamos los otros dos métodos: most popular, most voted.

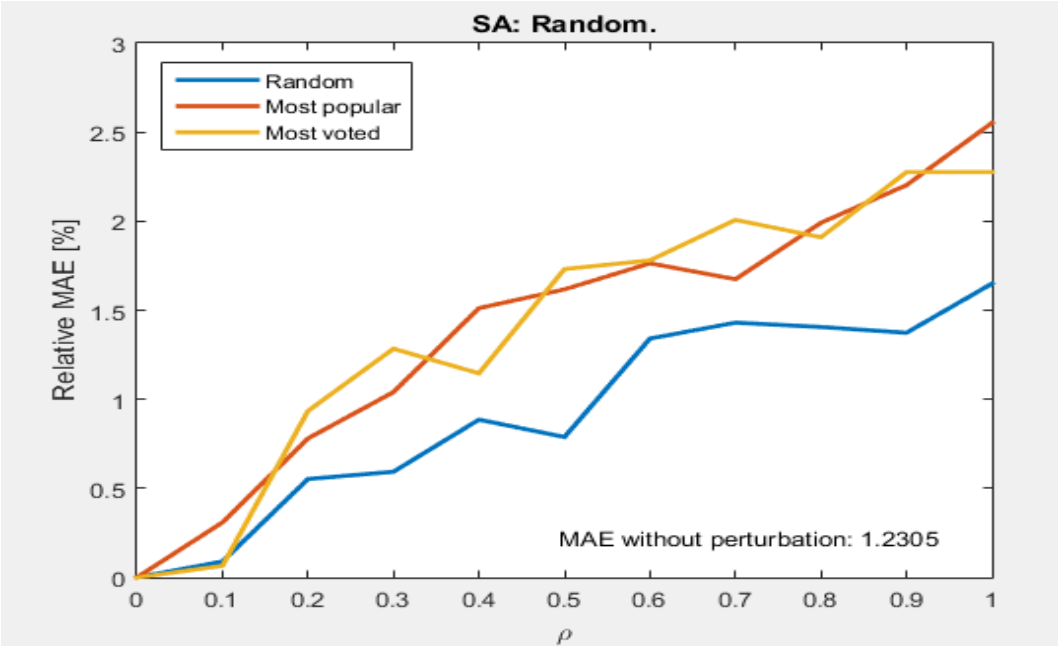


Figura 84. Evolución MAE relativa vs rho para RFTA y userbased

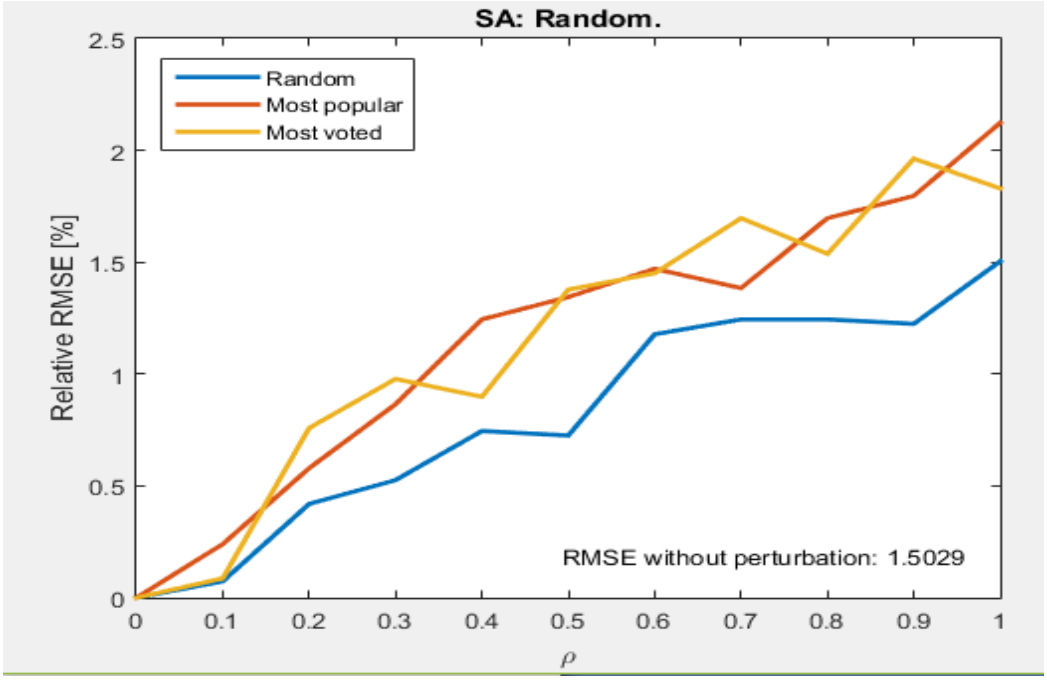


Figura 85. Evolución RMSE relativa vs rho para RFTA y userbased

También podemos visualizar las métricas respecto al algoritmo práctico que utilicemos en todos los casos RMSE, en formato relativo, se comporta mejor que MAE (figuras 86, 87 y 88)

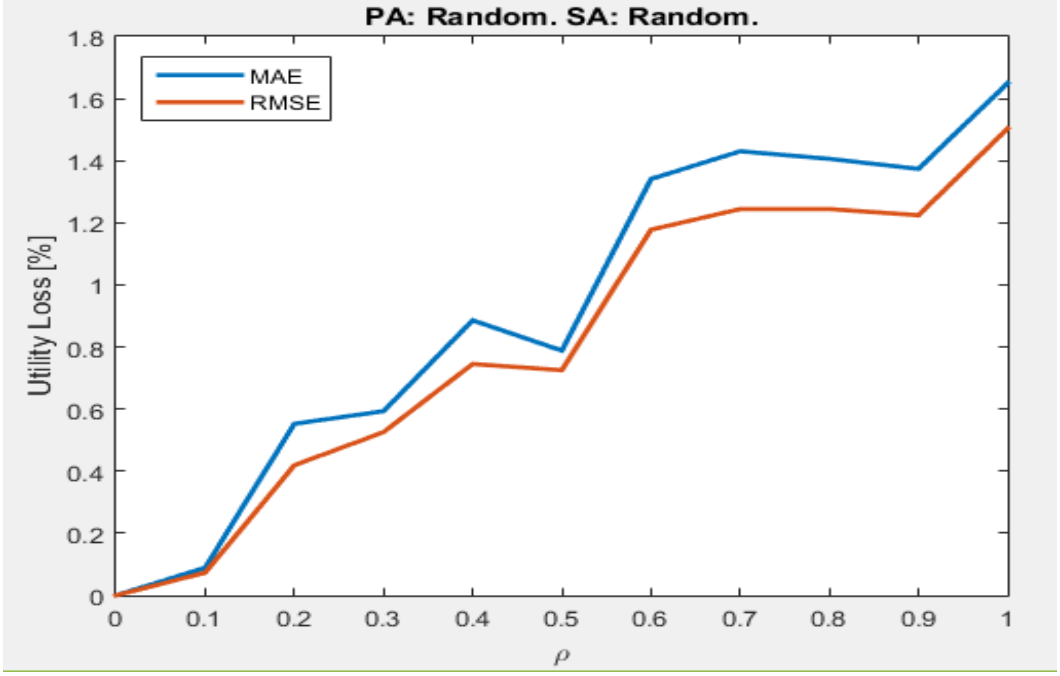


Figura 86. Evolución métricas relativas vs rho para RFTA, Random y userbased

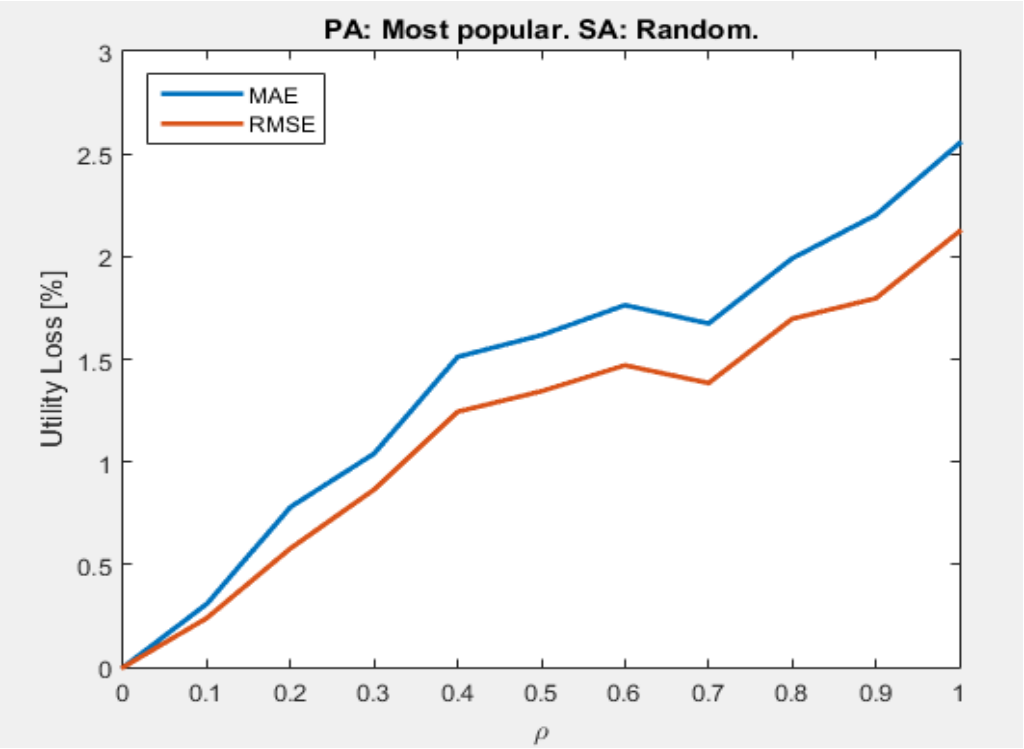


Figura 87. Evolución métricas relativas vs rho para RFTA, Most Popular y userbased

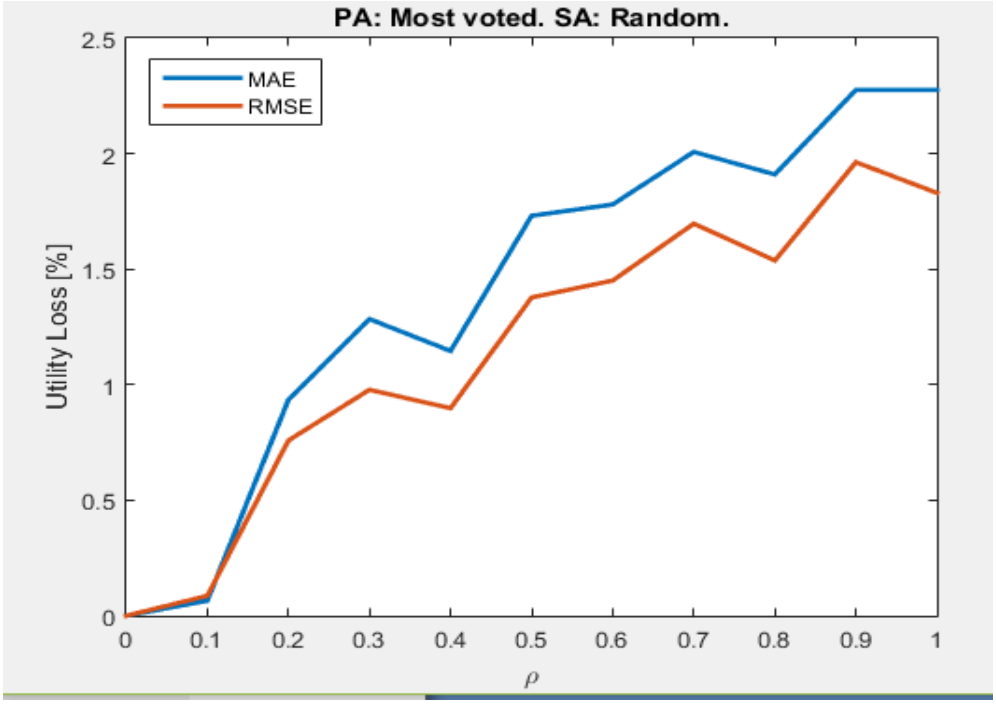


Figura 88. Evolución métricas relativas vs rho para RFTA, Most voted y userbased

El análisis del incremento relativo del riesgo de privacidad vs la pérdida de utilidad es muy parecido para los algortimos prácticos PA: Random, Most Popular y Most voted. La figura 89 representa el caso de Most popular, como vemos el incremento relativo del riesgo de privacidad se dispara con una pequeña pérdida de utilidad, para un 0.5% de pérdida tenemos casi un 1000% de incremento.

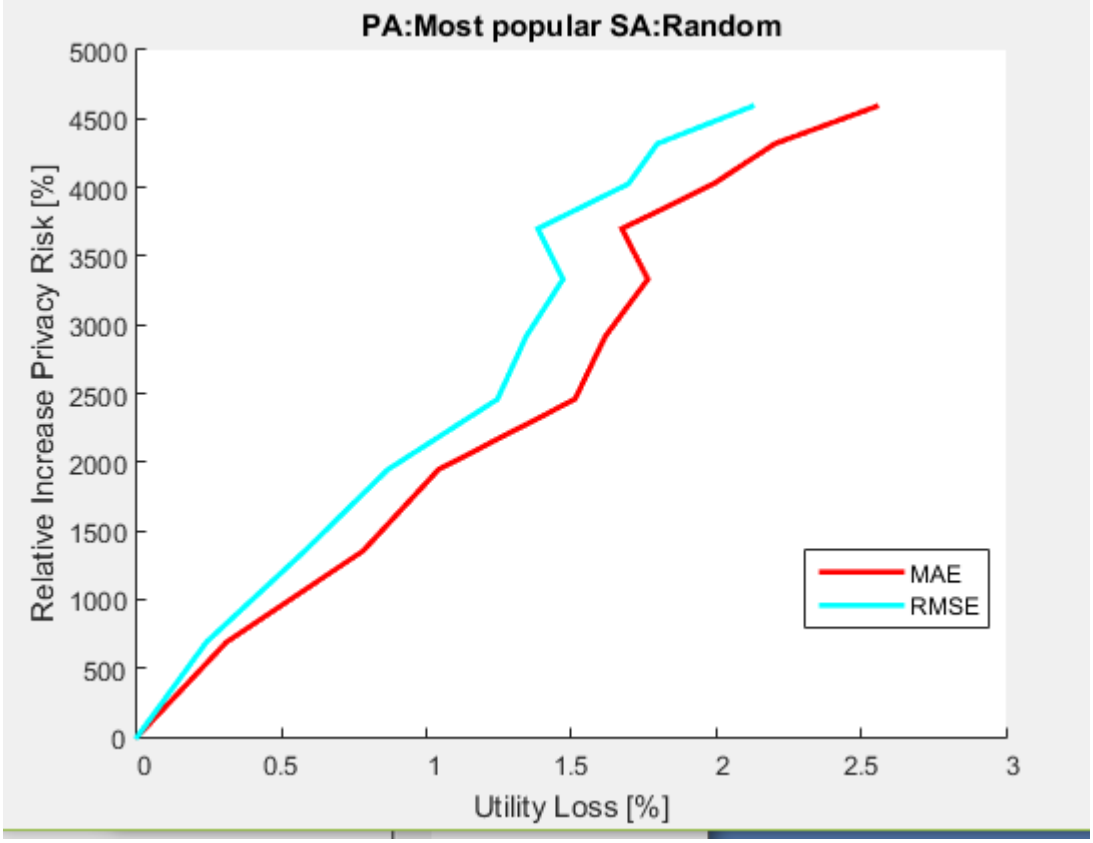


Figura 89. Incremento relativo del riesgo privacidad vs pérdida de utilidad para RFTA, most voted y userbased

Como estamos trabajando con un nuevo dataset vamos a ver cómo se comportan el error teórico y el error práctico.

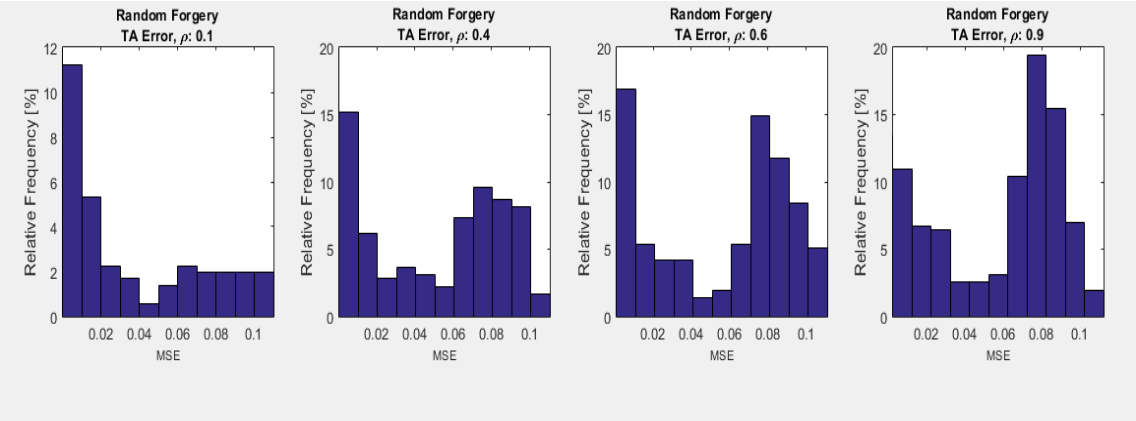


Figura 90, Error teórico para distintas rho, RFTA y userbased

Vemos que a medida que aumentmos ρ se incrementa a la vez el error teórico y práctico pero igualmente siguen siendo muy pequeños.

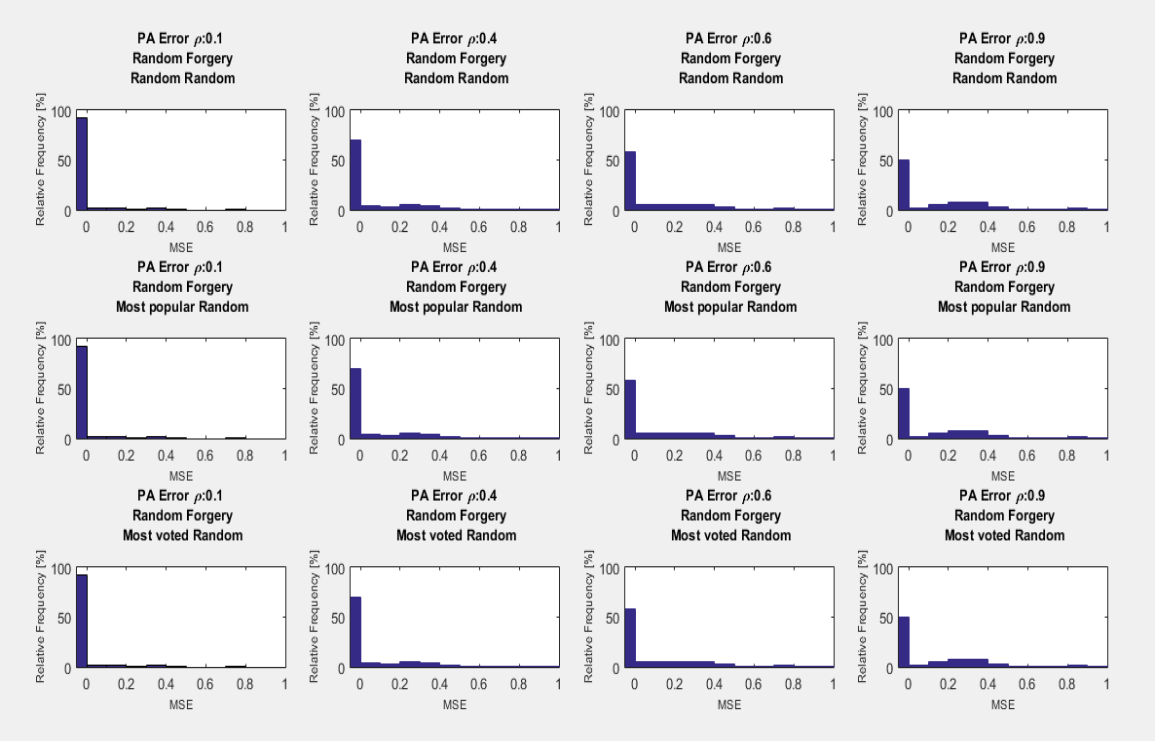


Figura 91, Error práctico para distintas rho, RFTA y userbased

RS: Itembased y TA:OFTA (Optimized Forgery)

Ahora utlizamos el sistema de recomendación itembased con el algoritmo teórico Optimizes Forgery. La evolución de las métricas MAE y RMSE relativas es parecido a otros casos vistos anteriormente, no tienen un aumento muy elevado a pesar de que aumentemos considerablemente la perturbación.

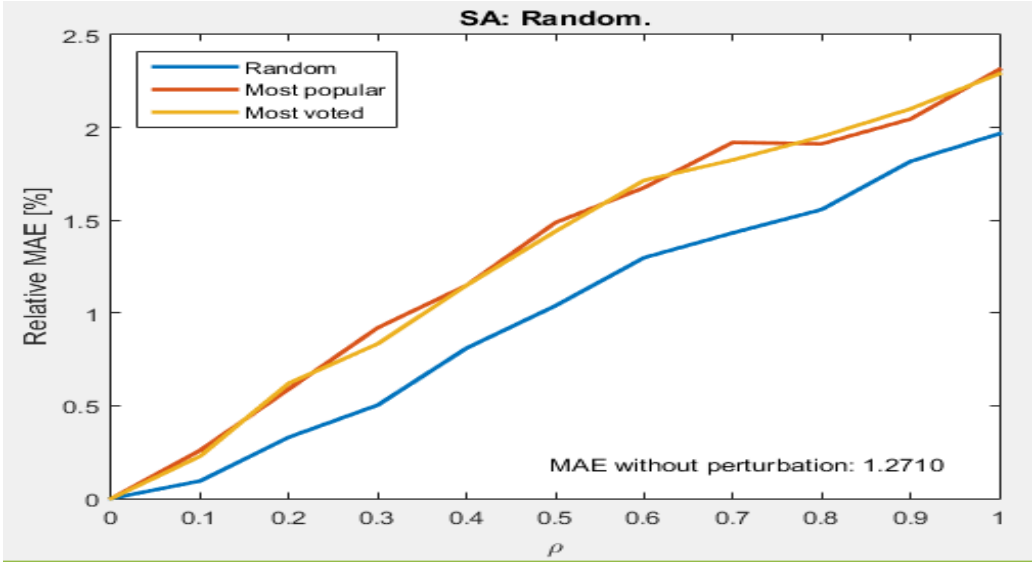


Figura 92. Evolución MAE relativa vs rho para OFTA e itembased

También, como antes, Random tiene mejor comportamiento en las dos métricas.

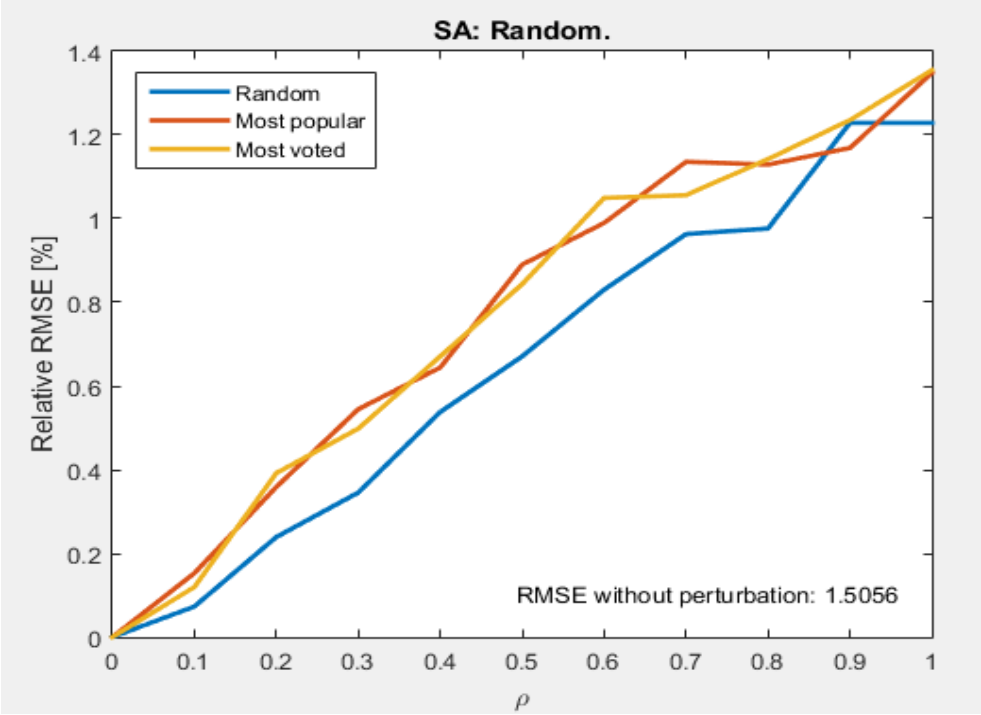


Figura 93. Evolución MAE relativa vs rho para OFTA e itembased

Como es un algoritmo optimizado tenemos reducción relativa del riesgo de privacidad vs la pérdida de utilidad. Su comportamiento es similar en los tres algoritmos prácticos, esta vez observaremos el caso de PA Random.

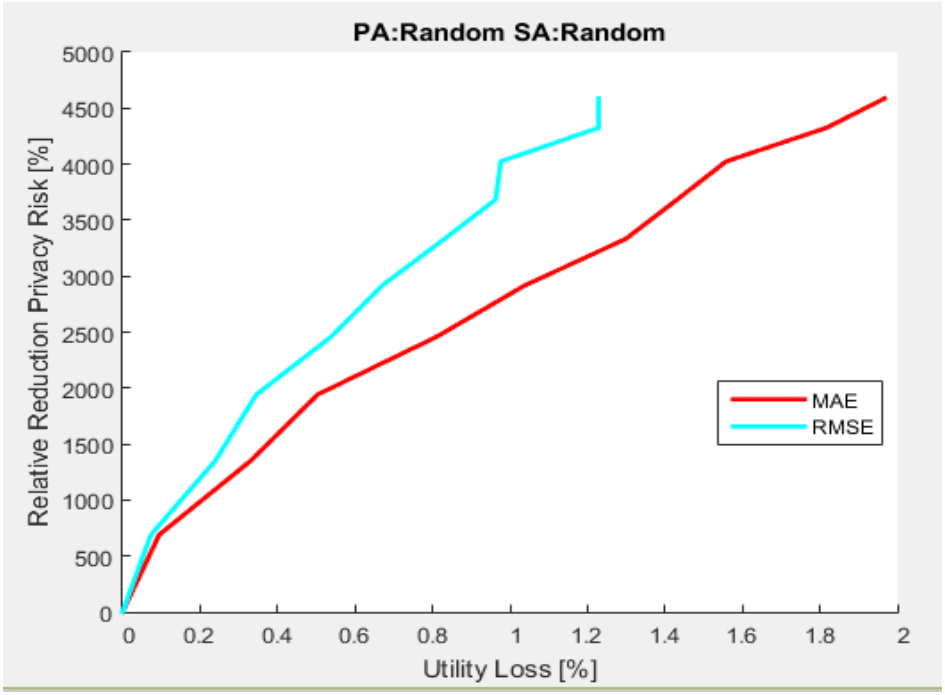


Figura 94. Reducción relativa del riesgo privacidad vs pérdida de utilidad para OFTA,random e itembased

Como en los casos anteriores que usaban el dataset yahoo, la reducción relativa del riesgo de privacidad se dispara con una pequeña modificación en la pérdida de utilidad.

RS: Regsvd y TA:OSTA (Optimized Suppression)

Por último vamos a analizar cómo se comporta el sistema de recomendación regsvd con el algoritmo teórico OSTA (Optimized Suppression) y algoritmo práctico Random.

Vemos que al aumentar σ también se incrementan las métricas MAE y RMSE relativas, llegando hasta el 55% como máximo en RMSE y hasta el 75% en MAE.

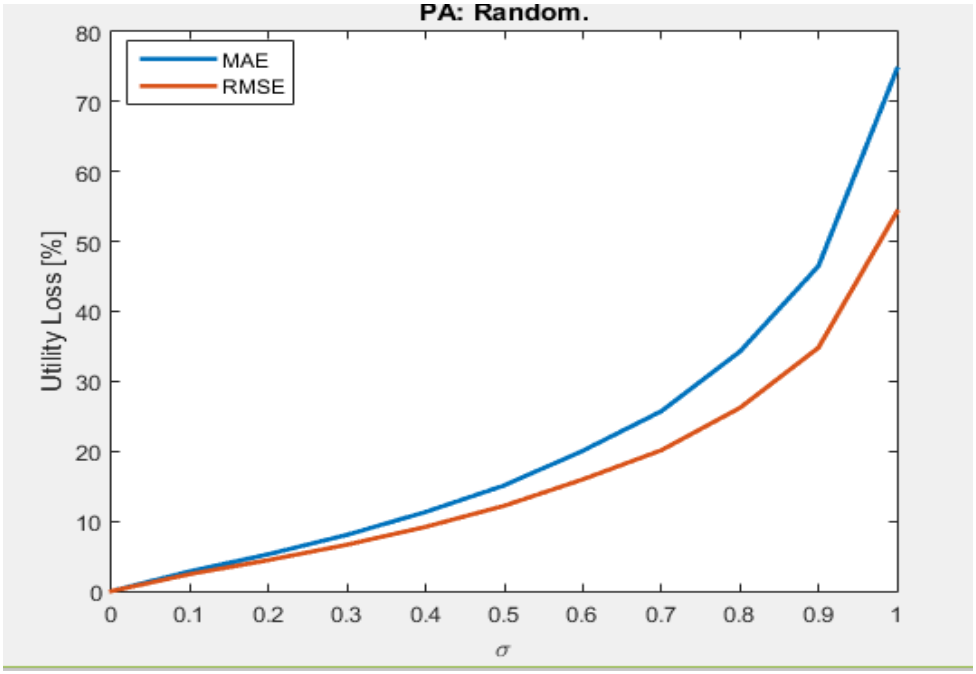


Figura 95. Evolución métricas relativas vs sigma para OSTA, Random y regsvd

Al ser un algoritmo teórico optimizado tenemos reducción relativa del riesgo de privacidad, en la figura 96 se observa como aumenta rápidamente la reducción relativa, con un 10% de pérdida de utilidad ya tenemos un 45% de reducción relativa del riesgo de privacidad.

Vemos que MAE y RMSE tienen un comportamiento casi idéntico.

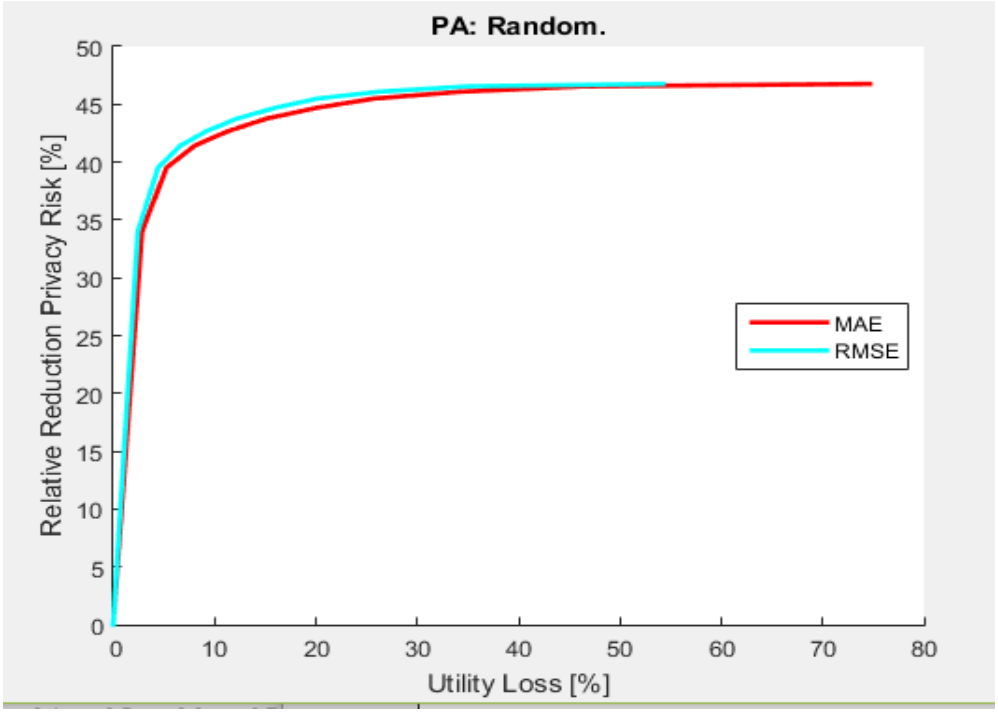


Figura 96. Reducción relativa del riesgo privacidad vs pérdida de utilidad para OSTA, random y regsvd

5. Conclusiones

El trabajo ha pasado por varias fases para su completo desarrollo. Lo primero fue conocer los sistemas de recomendación y las métricas que íbamos a utilizar y que estaban disponibles en el toolkit PREA. Escogimos aquellos sistemas más utilizados y que fueran una buena representación de los tipos basados en memoria y en modelo. Respecto a las métricas hemos escogido las más conocidas, MAE y RMSE, que nos daban una respuesta óptima aunque estudiamos y analizamos muchas más.

En la implementación de los algoritmos de perturbación desarrollamos los scripts a partir de las funciones dadas desde el comienzo del proyecto y de las características del toolkit PREA. La mayor dificultad en el trabajo radicaba en la gran cantidad de datos y cálculos que se debían realizar por lo se intentó optimizar todo lo posible el código y la forma de extraer la información que nos interesaba.

Por último, las gráficas realizadas nos permitieron analizar y comprobar cómo se ven afectadas la privacidad y la precisión de los sistemas de recomendación al utilizar mecanismos de perturbación.

Una vez finalizado el trabajo podemos destacar las siguientes conclusiones:

- Las pérdidas de utilidad no son tan grandes como a priori se podría suponer. En algunos casos, con un 100% de perturbación sólo tenemos pérdidas de utilidad del 6% (movieLens-100k,userbased, OFTA) o del

8% (movielens100k, userbased RFTA), aunque hay casos que es muy notable.

- Se comportan mejor los sistemas de recomendación basados en memoria que los basados en modelo, por ejemplo, regsvd y nmf, dos clases del segundo tipo y que se basan en la factorización podían llegar a tener pérdidas de utilidad del 300%.(movieLens-100k,regsvd,RSTA) o del 250%(movieLens-100k,nmf,OSTA).
- Las estrategias de "forgery" most popular y most voted, y las de "suppression" least popular y "least voted" tienen un comportamiento muy parecido en todas las gráficas. Esto se puede deber a que existe cierta correlación entre ellas. Parece que cuando votamos es para dar una buena puntuación, si no nos ha gustado el ítem simplemente no votamos.
- La estrategia random tanto para "suppression" como para "forgery" se comporta mejor que el resto de estrategias prácticas.
- Los errores teóricos y prácticos aumentan con el incremento de la perturbación pero siempre en valores muy pequeños por lo que no tienen casi influencia en los resultados finales.
- Respecto a las métricas, en su formato relativo, RMSE siempre tiene un mejor comportamiento que MAE.
- Los algoritmos teóricos de optimización permiten que al aumentar la pérdida de utilidad se reduzca el riesgo de privacidad mientras que los algoritmos teóricos no optimizados incrementan el riesgo de privacidad.

Durante el desarrollo del trabajo han surgido nuevas vías de estudio y posibilidades para mejorar algunos aspectos del proyecto que podrían ser interesante en futuros trabajos o estudios:

- Hemos utilizado la divergencia para calcular el riesgo de privacidad pero también se podría hacer a través de la entropía.
- Utilizar otro tipo de programas que no sea MATLAB que permita optimizar el código y reducir los tiempos de cálculo. Por ejemplo, aprovechando que PREA está realizado en JAVA, este lenguaje sería una buena opción.
- Ampliar el análisis a datasets más grandes, por ejemplo MovieLens 1M, yahoo en su totalidad, etc.

6.Glosario

ASYMM- Asymmetric measures.

ARFF- Attribute-Relation File Format.

BPMF-Bayesian Probabilistic Matrix Factorization

ETL- Extract Transform & Load.

HLU- Half-Life Utility

MAE- Mean Absolute Error

NDCG- Normalized Discounted Cumulative Gain

NLPMF- Non-linear Probabilistic Matrix Factorization

NMAE- Normalized Mean Absolute Error

NMF- Non-negative Matrix Factorization

OFTA- Optimized Forgery Theoretical Algorithm

OFSTA- Optimized Forgery Suppression Theoretical Algorithm

OSTA- Optimized Suppression Theoretical Algorithm

PA- Algoritmo práctico.

PMF- Probabilistic Matrix Factorization

PREA- Personalized REcommendation Algortihms Toolkit

RegSVD- Regularized Singular Value Descomposition

RMSE - Root of the Mean Square Error

RS- Recommender System

RSTA- Random Suppression Theoretical Algorithm

TA- Algoritmo teórico.

7.Bibliografia

"Optimal Forgery and Suppression of Ratings for Privacy Enhancement in Recommendation Systems," J. Parra-Arnau, D. Rebollo-Monedero, J. Forné,.

"Measuring the Privacy of User Profiles in Personalized Information Systems", J. Parra-Arnau, D. Rebollo-Monedero, J. Forné.

"A comparative Study of Collaborative Filtering Algortihms", Joonseok Lee, Mingxuan Sun, Guy Lebanon.

"Recommender Systems" Linyuan Lü, Matús Medo, Chi Ho Young, Yi-Cheng Zhang, Zi-Ke Zhang, Tao Zhou.

http://www.prea.gatech.edu/index.html. Fecha visita 2016-2017.

8. Anexos

Manual de uso scripts de MATLAB:

Existen tres scripts principales:

- Part1, que calcula los ítems de training y test set, crea la matriz de ratings y los perfiles de usuario y de población
- Part2, llama a los scripts de los algoritmos teóricos, los prácticos, a PREA y guarda los datos de riesgo de privacidad, medidas, errores, etc. en diferentes archivos para después poderlos representar.
- Grafica, representa las gráficas mostradas en el trabajo.

Configuración de Part1:

Solo hay que configurar ratio con el porcentaje de ítems puntuados que quieres en el training set y dataset con el nombre del dataset que te interesa.

```
ratio=0.5; % percentage of rated items to train set
%=======================================
% DATASETS
% SELECT
% ml-100k: movielens 100k ratings
% yahoo:  songs dataset 150k ratings
%=======================================

dataset='ml-100k';% 'yahoo' or 'ml-100k';
```

Configuración de Part2:

En este script se puede configurar el intervalo de trabajo de ρ, de σ, el porcentaje de perturbación de los usuarios, el dataset en uso y el sistema de recomendación que se quiere analizar.

```
sigma = 0:0.1:1;
rho = 0:0.1:1;
percentage=1;
dataset='ml_100k';
algorithm='userbased';
```

Configuración grafica:

Por último, en el script grafica podemos indicarle qué tipo de sistema de recomendación queremos ver, el algoritmo teórico, el intervalo de trabajo de σ y ρ y las métricas que queremos visualizar.

```
algorithm='userbased';
TA='Optimized Suppression';
sigma=0:0.1:1;
rho=0:0.1:1;
dataset='yahoo';
metricsWeWant={'MAE','RMSE'};
```