

## TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

**Análisis de métodos de privacidad  
 $\epsilon$ -diferencial local para la protección  
de datos sensibles.**

DIRECTOR/A: Mercedes Rodríguez García  
AUTOR/A: Antonio Requena Rodríguez

Puerto Real, septiembre de 2024

## TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

**Análisis de métodos de privacidad  
 $\epsilon$ -diferencial local para la protección  
de datos sensibles.**

DIRECTOR/A: Mercedes Rodríguez García  
AUTOR/A: Antonio Requena Rodríguez

Puerto Real, septiembre de 2024



# **Declaración personal de autoría**

---

Antonio Requena Rodríguez con DNI 49077202D, estudiante del Grado en Ingeniería Informática en la Escuela Superior de Ingeniería de la Universidad de Cádiz, como autor de este documento académico titulado Análisis de métodos de privacidad  $\varepsilon$ -diferencial local para la protección de datos sensibles y presentado como Trabajo Final de Grado en Ingeniería Informática.

## **DECLARO QUE**

Es un trabajo original, que no copio ni utilice parte de obra alguna sin mencionar de forma clara y precisa su origen tanto en el cuerpo del texto como en su bibliografía y que no empleo datos de terceros sin la debida autorización, de acuerdo con la legislación vigente. Asimismo, declaro que soy plenamente consciente de que no respetar esta obligación podrá implicar la aplicación de sanciones académicas, sin perjuicio de otras actuaciones que pudieran iniciarse. En Puerto Real, a 2 de septiembre de 2024

Fdo: Antonio Requena Rodríguez



# Agradecimientos

---

A mi familia, en especial a mis padres y a mi hermana, quienes han sido mi apoyo incondicional. Su amor y sacrificio me han guiado en cada paso de este camino. Gracias por creer en mí, por enseñarme la importancia del esfuerzo y la perseverancia, y por estar a mi lado en cada momento, celebrando mis logros y apoyándome en las dificultades. Sin vosotros, nada de esto habría sido posible.

A mis amigos, tanto a aquellos que he tenido la suerte de conocer en la universidad como a los que me han acompañado desde pequeño. A vosotros, que habéis sido mis compañeros de risas, aventuras y también de momentos difíciles, os debo mucho de lo que soy hoy. Gracias por estar siempre a mi lado, por compartir alegrías y apoyarme en cada paso del camino. Vuestra amistad ha sido un pilar fundamental en esta etapa de mi vida.

A todos los docentes de la carrera, especialmente a aquellos que forman parte de la rama de computación. Su dedicación, pasión y compromiso han sido fundamentales en mi formación académica. En particular, quiero agradecer a Mercedes, mi tutora, quien confió en mí desde el primer momento, que me introdujo en el mundo de la investigación. Su trabajo y apoyo constante durante estos años de colaboración han sido clave para la realización de este proyecto. Gracias por ser una profesora excepcional y por creer en mi potencial. Espero poder compartir futuras experiencias profesionales.

Por último, a mis compañeros de prácticas en Navantia S.A., quienes han hecho de estos últimos meses un tiempo de aprendizaje y crecimiento personal. Agradezco especialmente a mi tutor, Alejandro, por introducirme en el ámbito profesional con paciencia y dedicación. Su orientación y confianza han sido clave para que pudiera aprovechar al máximo esta oportunidad y para sentirme parte del equipo desde el primer día. Gracias por brindarme las herramientas necesarias para dar mis primeros pasos en el mundo profesional.



# Resumen

---

La situación tecnológica actual requiere la recolección continua de información acerca de gran cantidad de usuarios. Esto es fundamental para desarrollar productos y servicios personalizados, mejorar la eficiencia operativa y obtener ventajas competitivas en el mercado. Sin embargo, la recolección continua de información sobre una gran cantidad de usuarios plantea desafíos significativos en términos de privacidad y seguridad. Los usuarios son cada vez más conscientes de los riesgos asociados con la exposición de sus datos personales y exigen mayores niveles de protección. Por otro lado, la ciberdelincuencia crece constantemente, con ataques cada vez más sofisticados que comprometen la integridad y la seguridad de los datos. Por estos motivos, empresas punteras en tecnología apuestan por la investigación en técnicas dedicadas a la obtención de analíticas eficientes sobre conjuntos de datos masivos, mientras se preserva la privacidad y seguridad de los usuarios.

En este marco, el presente proyecto tiene por objetivo realizar un estudio en profundidad de las diferentes técnicas y modelos de privacidad existentes. Entre las tecnologías emergentes en este ámbito se encuentran los métodos de privacidad  $\varepsilon$ -diferencial, que permiten realizar análisis estadísticos sobre conjuntos de datos sin revelar información específica de los individuos. Y su combinación con estructuras de datos probabilísticas, en especial los data sketches. Estas estructuras tienen la capacidad de almacenar de forma compacta la información de un flujo de datos masivo. A lo largo del documento, se presentan diferentes algoritmos que combinan estas características, los cuales serán implementados y evaluados frente a una variedad de experimentos.

**Palabras clave:** Privacidad  $\varepsilon$ -diferencial, data sketch, hash, dataset, método de enmascaramiento, cuasi-identificador, anonimización, reidentificación.



# Índice general

---

Índice de figuras	ix
Índice de tablas	xii
<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	1
1.2. Alcance y objetivos . . . . .	2
1.3. Organización del documento . . . . .	3
<b>2. Privacidad de datos</b>	<b>4</b>
2.1. Introducción a la privacidad de datos . . . . .	4
2.2. Información de identificación personal . . . . .	5
2.3. Tipos de datos . . . . .	6
2.4. Modelos de privacidad . . . . .	8
2.4.1. $k$ -anonimidad . . . . .	8
2.4.2. $k$ -anonimidad probabilística . . . . .	11
2.4.3. Privacidad $\epsilon$ -diferencial . . . . .	11
2.4.3.1. Mecanismos $\epsilon$ -diferencialmente privados . . . . .	15
2.4.3.2. Privacidad $\epsilon$ -diferencial local . . . . .	16
2.5. Técnicas de enmascaramiento de datos . . . . .	17
2.5.1. Técnicas de enmascaramiento no perturbativas . . . . .	18
2.5.1.1. Generalización . . . . .	18
2.5.2. Técnicas de enmascaramiento perturbativas . . . . .	19
2.5.2.1. Microagregación . . . . .	20
2.5.2.2. Adición de ruido . . . . .	21
2.5.2.3. Rank swapping . . . . .	23
<b>3. Data sketches</b>	<b>25</b>
3.1. Introducción a las estructuras de datos probabilísticas . . . . .	25
3.2. Familia de hashes independientes entre sí . . . . .	26
3.3. Definición y fundamentos de los data sketches . . . . .	27
3.4. Propiedades de los data sketches . . . . .	28
3.5. Notación y terminología . . . . .	29
3.6. Tipos de data sketches . . . . .	30
3.6.1. Filtros de Bloom . . . . .	30
3.6.2. Estimadores de frecuencia . . . . .	33
3.6.2.1. Count-Min Sketch . . . . .	34
3.6.2.2. AMS Sketch . . . . .	36
3.6.2.3. Count Sketch . . . . .	38
3.6.3. Estimadores de cardinalidad . . . . .	39
3.6.3.1. Flajolet-Martin Sketch . . . . .	40
3.6.3.2. HyperLogLog . . . . .	42
3.6.3.3. Theta Sketch . . . . .	44

3.7. Comparativa de los data sketches expuestos . . . . .	46
<b>4. Data sketches diferencialmente privados</b>	<b>48</b>
4.1. Private Count Mean Sketch . . . . .	48
4.2. Private Hadamard Count Mean Sketch . . . . .	50
4.3. Private Sequence Fragment Puzzle . . . . .	52
4.4. Randomized Aggregatable Privacy-Preserving Ordinal Response . . . . .	55
4.5. Mecanismo dBitFlip para la estimación de histogramas . . . . .	60
<b>5. Evaluación y análisis de las técnicas implementadas</b>	<b>63</b>
5.1. Datasets empleados para el análisis . . . . .	63
5.2. Implementación de los métodos . . . . .	66
5.3. Especificaciones del entorno de ejecución . . . . .	67
5.4. Métricas de Evaluación . . . . .	68
5.4.1. Inconvenientes respecto a la medición del error . . . . .	69
5.5. Experimentos . . . . .	70
5.5.1. Experimento 1. Comparativa entre los algoritmos Private Count Mean Sketch y Private Hadamard Count Mean Sketch . . . . .	71
5.5.2. Experimento 2. Análisis de la capacidad del algoritmo Private Secuence Fragment Puzzle para la estimación de un diccionario desconocido. . . . .	77
5.5.3. Experimento 3. Evaluación del rendimiento del algoritmo RAPPOR	83
5.5.4. Experimento 4. Evaluación del rendimiento del algoritmo dBitFlip	89
<b>6. Conclusiones</b>	<b>94</b>
6.1. Objetivos alcanzados . . . . .	98
6.2. Lecciones aprendidas . . . . .	99
6.3. Trabajo futuro . . . . .	99

# Índice de figuras

---

2.1. Ejemplo de reidentificación a partir de bases de datos ajenas. . . . .	9
2.2. Conjunto de microdatos 3-anónimo . . . . .	10
2.3. Conjunto de microdatos 2-diverso . . . . .	10
2.4. Proceso de obtención de consultas $\varepsilon$ -diferencialmente privadas . . . . .	13
2.5. Dataset de características tumorales al que se le ha añadido una característica ruidosa (fractal_dimension_NOISE). Dataset original: Breast Cancer Wisconsin (Diagnostic) - Extraido de <a href="https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data">https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data</a> . . . . .	14
2.6. Dataset de características tumorales al que se le ha añadido ruido tras eliminar el registro con el registro con identificador 92529. Dataset original: Breast Cancer Wisconsin (Diagnostic) - Extraido de <a href="https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data">https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data</a> . . . . .	15
2.7. Jerarquía de generalización del atributo <i>puesto de trabajo</i> . . . . .	18
2.8. Datos sobre los salarios de una empresa previo aplicar microagregación.	20
2.9. Datos de la Figura 2.8 tras aplicar microagregación sobre el atributo <i>sueldo</i> , agrupando los registros según el puesto de trabajo. . . . .	20
2.10. Conjunto de datos sintéticos con características de individuos, previo a la adición de ruido al atributo <i>Peso</i> . . . . .	22
2.11. Conjunto de datos sintéticos con características de individuos, tras la adición de ruido al atributo <i>Peso</i> . . . . .	22
2.12. Base de datos sintética sobre los pacientes de un centro de salud. . . . .	24
2.13. Base de datos sintética sobre los pacientes de un centro de salud tras aplicar rank swapping al atributo <i>Temperatura</i> . . . . .	24
3.1. Esquema generación de un data sketch . . . . .	28
3.2. Inserción en un filtro de Bloom . . . . .	31
3.3. Búsqueda en un filtro de Bloom . . . . .	32
3.4. Actualización de un count-min sketch . . . . .	34
3.5. Estimación en un count-min sketch . . . . .	35
3.6. Fase de actualización en un AMS sketch . . . . .	37
3.7. Consulta de punto sobre un Count Sketch . . . . .	39
3.8. Flujo de actualización en un Flajolet-Martin sketch . . . . .	41
3.9. Distribución por subconjuntos en HyperLogLog . . . . .	43
3.10. Actualización en un Theta Sketch . . . . .	45
5.1. Estructura interna del dataset de palabras consideradas anglicismos. . .	64
5.2. Dataset de números enteros generados aleatoriamente a partir de una distribución exponencial. . . . .	65
5.3. Dataset de números enteros generados aleatoriamente a partir de una distribución normal ( $\mu = 12, \sigma = 2$ ). . . . .	65
5.4. Gráfica comparativa de la evolución de la media de error en ambos algoritmos en función del valor de $\varepsilon$ . . . . .	75

5.5.	Evolución del error porcentual en ambos algoritmos en función del incremento del tamaño del dataset ( $N$ ). Parámetros empleados $\varepsilon = 4$ , $k = 1024$ y $m = 256$ . . . . .	76
5.6.	Evolución del tiempo de ejecución para la generación del diccionario frente al incremento de $k'$ , . . . . .	80
5.7.	Ejemplo de diccionario con falsos positivos generado tras ejecutar el algoritmo Sequence Fragment Puzzle ( $T = 80$ ) . . . . .	82
5.8.	Evolución del número de cadenas candidatas generadas en función del parámetro $T$ . . . . .	83
5.9.	Comparativa entre los resultados obtenidos del algoritmo RAPPOR para los elementos de mayor frecuencia real. . . . .	88
5.10.	Comparativa entre los resultados obtenidos del algoritmo RAPPOR para los elementos de menor frecuencia real . . . . .	88
5.11.	Curvatura descrita por el RMSE normalizado obtenido en el algoritmo dBitFlip, en función de los parámetros $d$ y $\varepsilon$ (Valores de $\varepsilon$ reducidos). . . . .	92
5.12.	Curvatura descrita por el RMSE normalizado obtenido en el algoritmo dBitFlip, en función de los parámetros $d$ y $\varepsilon$ (Valores de $\varepsilon$ altos). . . . .	92
6.1.	Insignia de <i>Artefactos Disponibles</i> que indica que el código está disponible públicamente en un repositorio accesible. Véase la fuente en [7]. . . . .	97
6.2.	Insignia de <i>Resultados Reproducidos</i> que muestra que el código permite la reproducción de los resultados presentados en el artículo. Véase la fuente en [7]. . . . .	98

# Índice de tablas

---

3.1. Comparativa entre data sketches . . . . .	46
3.2. Métricas de error/utilidad asociadas a los diferentes data sketches . . . . .	47
5.1. Coste temporal asociado a la ejecución del algoritmo Private Count Mean Sketch, en función de los parámetros $m$ y $k$ . (dataset: exp_distrib_50k y $\varepsilon = 2$ ) . . . . .	72
5.2. Métricas de error asociadas a la ejecución del algoritmo Private Count Mean Sketch, en función de los parámetros $m$ y $k$ . (dataset: exp_distrib_50k y $\varepsilon = 2$ ) . . . . .	72
5.3. Coste temporal asociado a la ejecución del algoritmo Private Hadamard Count Mean Sketch, en función de los parámetros $m$ y $k$ . (dataset: exp_distrib_50k y $\varepsilon = 2$ ) . . . . .	73
5.4. Métricas de error asociadas a la ejecución del algoritmo Private Hadamard Count Mean Sketch, en función de los parámetros $m$ y $k$ . (dataset: exp_distrib_50k y $\varepsilon = 2$ ) . . . . .	74
5.5. Resultados de ejecución del algoritmo Private Count Mean Sketch sobre un dataset que representa una distribución normal ( $\mu = 12, \sigma = 2$ ) (Dataset: norm_distrib_200k, $k = 1024$ y $m = 256$ ) . . . . .	74
5.6. Resultados de ejecución del algoritmo Private Hadamard Count Mean Sketch sobre un dataset que representa una distribución normal ( $\mu = 12, \sigma = 2$ ) (Dataset: norm_distrib_200k, $k = 1024$ y $m = 256$ ) . . . . .	75
5.7. Coste de transmisión (Ancho de Banda) de los datos enviados por el cliente en Private Count Mean Sketch en función del parámetro $m$ . . . . .	77
5.8. Coste de transmisión (Ancho de Banda) de los datos enviados por el cliente en Private Hadamard Count Mean Sketch en función del parámetro $m$ . . . . .	77
5.9. Tiempo de ejecución obtenido tras sucesivas ejecuciones del algoritmo Private Sequence Fragment Puzzle ( $\varepsilon = 2$ y $\varepsilon' = 6$ , dataset: anglicismo_50k) . . . . .	79
5.10. Métricas de error obtenidas tras sucesivas ejecuciones del algoritmo Private Sequence Fragment Puzzle ( $\varepsilon = 2$ y $\varepsilon' = 6$ , dataset: anglicismo_50k). . . . .	81
5.11. Total de errores y número de cadenas candidatas generadas al ejecutar el algoritmo Sequence Fragment Puzzle ( $\varepsilon = 2, \varepsilon' = 6, k = 256, k' = 256, m = 64$ y $m' = 64$ ) variando el valor que toma el umbral $T$ . . . . .	82
5.12. Resultados obtenidos, en términos de coste temporal, tras la ejecución del algoritmo RAPPOR según los parámetros $m$ y $k$ . ( $f = 0,5, p = 0,5, q = 0,75$ , dataset: exp_distrib_500k) . . . . .	84
5.13. Resultados obtenidos, en términos de error, tras la ejecución del algoritmo RAPPOR según los parámetros $m$ y $k$ . ( $f = 0,5, p = 0,5, q = 0,75$ , dataset: exp_distrib_500k) . . . . .	85

5.14. Resultados obtenidos tras la ejecución del algoritmo RAPPOR según el tamaño del dataset. ( $m = 128$ , $k = 8$ , $f = 0,5$ , $p = 0,5$ , $q = 0,75$ , dataset: norm_distrib_N) . . . . .	86
5.15. Métricas de error y resultados de privacidad obtenidos tras las sucesivas ejecuciones del algoritmo RAPPOR, variando los parámetros $f$ , $p$ y $q$ . ( $m = 128$ , $k = 8$ , dataset: exp_distrib_500k) . . . . .	87
5.16. Resultados obtenidos, en términos de coste temporal, tras la ejecución del algoritmo dBitFlip según los parámetros $d$ y $\varepsilon$ . (dataset: exp_distrib_500k)	90
5.17. Resultados obtenidos, en términos de error, tras la ejecución del algoritmo dBitFlip según los parámetros $d$ y $\varepsilon$ . (dataset: exp_distrib_500k) . . .	91
5.18. Impacto del número de registros del dataset empleado en los resultados del algoritmo dBitFlip. ( $d = 4$ , $\varepsilon = 1$ , dataset: norm_distrib_N) . . . . .	93





# 1. Introducción

---

En la actualidad, la privacidad de los datos se ha convertido en un tema de preocupación primordial. La creciente cantidad de información personal y sensible que se comparte ha generado una problemática crítica en cuanto a la seguridad y la privacidad de los usuarios. En el contexto empresarial, la confianza de los consumidores en la protección de sus datos personales es un activo esencial para cualquier empresa en la economía actual. La pérdida o filtraciones de datos pueden tener consecuencias devastadoras, incluyendo sanciones legales, daños a la reputación, y pérdida de clientes. En respuesta a la necesidad de proteger la información, se ha observado un impulso importante en la investigación de privacidad de datos por parte de compañías líderes en tecnología, como Apple, Google y Microsoft. Estas entidades proponen nuevos enfoques basados en sketching y privacidad diferencial para proteger la información personal que los usuarios comparten con sus servidores. La combinación de sketching, capaz de reducir la cantidad de información crítica o sensible que se comparte, con la privacidad diferencial, capaz de proteger la información sensible mediante la adición de ruido, proporciona un método eficaz y sostenible para compartir y proteger datos personales. La aplicación de estas técnicas permite cumplir con la normativa vigente en materia de protección de datos, sin renunciar al intercambio de información personal tan necesario.

El presente proyecto busca presentar soluciones innovadoras y eficaces para la protección de datos, combinando técnicas avanzadas de sketching y privacidad diferencial. Se analizarán los conceptos y aplicaciones de la privacidad diferencial, destacando su capacidad para proteger datos mediante la adición de ruido controlado. Se investigarán diferentes métodos de sketching, que permiten reducir la cantidad de información compartida sin perder su utilidad, y su aplicabilidad en la protección de datos sensibles. Se evaluarán las propiedades de anonimato en la comunicación entre clientes y servidores. Se implementarán, en el lenguaje de programación Python, diferentes algoritmos que combinan dichas técnicas, con un enfoque centrado en algoritmos destinados a la estimación de frecuencias sobre flujos de datos masivos. Se utilizarán diferentes conjuntos de datos para probar y evaluar el rendimiento y eficacia de los algoritmos y se realizarán comparaciones entre las técnicas utilizadas para identificar sus fortalezas y debilidades. Este análisis permitirá evaluar la eficiencia y efectividad de los algoritmos implementados en comparación con otras soluciones existentes.

## 1.1. Motivación

La motivación detrás de este proyecto es doble. En primer lugar, este proyecto ofrece una oportunidad valiosa para profundizar en áreas específicas del conocimiento que son de gran interés y relevancia para el campo de la privacidad de datos. En particular, me ha permitido ampliar mis conocimientos sobre la privacidad diferencial local, una variante crucial que tiene aplicaciones significativas en la protección de datos

personales, y familiarizarme con estructuras de datos probabilísticas y técnicas de sketching. Estas técnicas de sketching, que son esenciales para manejar datos masivos de manera eficiente, representan metodologías clave en la protección de grandes volúmenes de datos. El estudio de estas técnicas enriquecerá mi comprensión teórica, pero también me proporcionará habilidades prácticas de cara a la investigación en dicho campo y el desarrollo de soluciones efectivas que hagan uso de estas técnicas.

En segundo lugar, se busca contribuir al avance del conocimiento en el campo de la protección de datos personales mediante la implementación de estos algoritmos en un lenguaje de programación popularmente extendido y su evaluación frente a diferentes escenarios de prueba. Evaluar y analizar el rendimiento y la eficacia de estas técnicas permitirá identificar qué enfoques son más efectivos en distintos contextos y para diferentes tipos de datos.

De forma adicional, este proyecto podrá ofrecerme la oportunidad de comunicar y difundir los resultados y hallazgos a la comunidad académica y profesional. Publicar los resultados en artículos especializados contribuirá a enriquecer el debate en torno a la protección de datos personales y su combinación con estructuras de datos modernas.

## 1.2. Alcance y objetivos

El presente proyecto se centra en el estudio y la aplicación de técnicas avanzadas en el ámbito de la privacidad diferencial para la protección de datos. Su alcance y objetivos abarcan una serie de áreas clave. Durante el proyecto se investigará en profundidad la literatura existente sobre privacidad diferencial, un paradigma crucial para proteger la información personal al introducir ruido en los datos. Se revisarán los fundamentos matemáticos subyacentes y las metodologías empleadas, destacando los desarrollos recientes y aplicaciones prácticas.

Se analizarán diversas técnicas de sketching, de forma aislada, las cuales podrán ser combinadas con privacidad diferencial. Detallando cómo se utilizan para estimar funciones agregadas mientras se preserva la privacidad de los datos individuales. Se explorarán sus aplicaciones específicas y su efectividad en comparación con otros métodos. En este contexto, se investigará cómo se gestionan las propiedades de anonimato en las interacciones entre clientes y servidores, evaluando la robustez de los sistemas de comunicación frente a posibles filtraciones de datos.

Uno de los objetivos prácticos del proyecto consiste en la implementación de los algoritmos de privacidad expuestos utilizando un lenguaje de programación popular. Esto permitirá evaluar la viabilidad y eficiencia de las técnicas en escenarios de procesamiento de flujos masivos de datos, cercanos a la realidad. Se utilizarán datasets sintéticos para evaluar el rendimiento de las técnicas implementadas. Se realizará un análisis comparativo para determinar la eficacia, la precisión y la eficiencia de cada técnica en términos de coste temporal, error en la estimación y privacidad asociada.

Finalmente, el proyecto permitirá identificar áreas emergentes y cuestiones abiertas en el campo de la privacidad diferencial y el sketching. Esto incluirá desafíos actuales, limitaciones de las técnicas existentes y posibles caminos futuros para la inves-

tigación. Un objetivo adicional interesante consiste en la difusión de los resultados y hallazgos del proyecto a través de seminarios, publicaciones académicas y otros medios pertinentes, con el objetivo de contribuir al avance del conocimiento en dichas áreas de estudio. Este objetivo podría expandirse en el futuro para incluir actividades adicionales, dependiendo de los recursos disponibles y la recepción de la comunidad académica. Por lo tanto, este objetivo queda definido a futuro y no se incluye en el alcance inicial del proyecto.

### 1.3. Organización del documento

Al tratarse de un proyecto centrado en la investigación, gran parte del mismo se destina al estudio en profundidad de la literatura y su comprensión, incluyendo la exposición y explicación detallada de los conceptos. En el capítulo 2 se introduce el concepto de privacidad de datos, su importancia y los marcos legales que la regulan. Se analizan los tipos de datos que se consideran información de identificación personal y se clasifica la información en diferentes categorías. Las partes más destacables de este capítulo se centran en describir los principales modelos de privacidad, sección 2.4, sus características y cómo se pueden combinar para ofrecer una protección más robusta. Adicionalmente, se exponen diferentes técnicas de enmascaramiento relacionadas y su aplicación práctica, sección 2.5. El capítulo 3 se destina al estudio de estructuras de datos probabilísticas, definiendo el propósito de estas y algunos conceptos previos necesarios. A partir de la sección 3.3, el proyecto se centra en el área de las técnicas de sketching, sus propiedades básicas, notación empleada, y se detallará el funcionamiento de diferentes técnicas específicas, discriminando según su funcionalidad.

El capítulo 4 utiliza los conceptos teóricos descritos en los capítulos anteriores para exponer, con un alto grado de detalle, diversos algoritmos que combinan técnicas y estructuras de datos propias del sketching, junto con los fundamentos teóricos de la privacidad  $\epsilon$ -diferencial local. Durante este capítulo, se tratarán los algoritmos de forma individual, apoyándose en pseudocódigo, que detalle el funcionamiento real de estos. Dichos algoritmos se implementarán en el lenguaje de programación Python. Todo el código asociado a las técnicas descritas, junto con las instrucciones de instalación y ejecución, se encontrará disponible en un repositorio de GitHub definido en la sección 5.2.

A partir de los algoritmos implementados en el capítulo anterior, el capítulo 5, recoge los resultados una serie de experimentos llevados a cabo para evaluar y analizar la eficiencia y precisión de los algoritmos descritos. Previamente se definen una serie de requisitos, como el conjunto de datasets empleados en el análisis, especificaciones acerca del hardware utilizado en el desarrollo y análisis de las técnicas, o las métricas de evaluación empleadas. Cada uno de estos experimentos recoge los resultados de diferentes ejecuciones, en las que se evalúan los resultados en función del valor de diversos parámetros. Los experimentos recogen explicaciones de los resultados fundamentadas en la base teórica subyacente, y otras conclusiones de interés. Finalmente, el capítulo 6 recoge diferentes conclusiones derivadas de la totalidad del proyecto. Además, este capítulo reserva un espacio para analizar los objetivos alcanzados, lecciones aprendidas, y asentar las bases para un trabajo futuro.

## **2. Privacidad de datos**

---

En este capítulo se introducen una serie de conceptos claves en el ámbito de la privacidad de datos. Desde la necesidad de la privacidad de los datos y su marco legal, los tipos y la forma en la que pueden agruparse los datos, los tipos de datos en base a su sensibilidad, hasta los diferentes modelos de privacidad y técnicas de enmascaramiento actuales.

### **2.1. Introducción a la privacidad de datos**

La privacidad de los datos en la era digital actual, se ha convertido en un tema de gran importancia y preocupación. La protección de nuestros datos personales es crucial en un mundo en el que la información fluye constantemente a través de las redes sociales, las aplicaciones móviles y los servicios en línea. Nuestros datos son valiosos para empresas, gobiernos y entidades malintencionadas por igual. Dichos datos incluyen los nombres de las personas, la información de contacto, la edad, la dirección y el comportamiento en línea y en la vida real. En este contexto, la protección de datos personales es esencial para prevenir el robo de identidad y mantener la confidencialidad.

La importancia de la privacidad de datos radica en varios aspectos cruciales. En primer lugar, es fundamental para proteger nuestra identidad y mantenernos a salvo del robo de identidad y otras formas de fraude cibernético. Además, preservar la privacidad de nuestros datos personales es esencial para salvaguardar nuestra autonomía y libertad. Sin ella, los usuarios corren el riesgo de ser manipulados o controlados por fuerzas externas que pueden utilizar dicha información para influir en sus decisiones y comportamientos.

El hecho de no contar con políticas de privacidad robustas, puede tener consecuencias que causen daño a la imagen de las organizaciones, empresas o gobiernos afectados. Pero también existen otras razones para que los organismos encargados de recopilar y trabajar con datos personales se preocupen por la confidencialidad, ya que si las personas no están convencidas de que su privacidad está protegida correctamente, es poco probable que acepten y proporcionen sus datos para que se trabaje sobre ellos. La falta de confianza en la protección de la privacidad puede frenar la adopción de nuevas tecnologías y limitar el potencial de la innovación digital.

En el marco legal, la protección de datos se define como un derecho que otorga a las personas el poder de controlar y decidir sobre la información personal que comparten con terceros, ya sea el Estado o particulares. Este derecho fundamental, irrenunciable y prevalece sobre otros derechos no fundamentales, permite a los individuos saber quién posee sus datos personales y para qué se utilizan, así como oponerse a su posesión o uso [18].

En el ámbito europeo, la protección de datos se establece como un derecho fundamental según el Tratado de Funcionamiento de la Unión Europea y la Carta de los

Derechos Fundamentales de la Unión Europea. La Reglamentación General de Protección de Datos (**RGPD**) [43], en vigor desde mayo de 2018, unifica y fortalece la protección de datos para los ciudadanos de la Unión Europea y regula el tratamiento de datos personales por parte de las organizaciones. Sus principales aspectos incluyen un ámbito de aplicación amplio, el requisito de consentimiento explícito para el procesamiento de datos, la garantía de varios derechos para los ciudadanos de la UE (como el acceso y la rectificación), la imposición de responsabilidades y medidas de transparencia para las organizaciones, la obligación de notificar brechas de seguridad y sanciones por incumplimiento que pueden ser significativas. En definitiva, la **RGPD** busca brindar a los ciudadanos de la UE un mayor control sobre sus datos personales y establecer estándares más rigurosos para la protección de la privacidad, mientras impone obligaciones claras y significativas a las organizaciones que manejan dichos datos.

En España, este derecho está regulado por la Ley Orgánica 15/1999, de Protección de Datos de Carácter Personal (**LOPD**) [31], y su Reglamento de desarrollo, así como por otras normativas sectoriales. La ley establece la obligación de implementar medidas técnicas y organizativas adecuadas para garantizar la seguridad de los datos personales y prevenir su alteración, pérdida, acceso o tratamiento no autorizado.

El objetivo principal de estas leyes en materia de protección, es garantizar y proteger las libertades públicas y los derechos fundamentales de las personas físicas, especialmente su intimidad personal y familiar, en relación con el tratamiento de datos personales. Esto se logra diferenciando entre datos de carácter personal y datos no personales, con el fin de salvaguardar la privacidad y la seguridad de la información de los individuos.

## 2.2. Información de identificación personal

La información de identificación personal (PII) es cualquier información relacionada con una persona específica que se puede usar para descubrir la identidad de la misma, como su número de la seguridad social, su nombre completo o su dirección de correo electrónico [29].

La cantidad de información de este tipo que se comparte con diversas organizaciones ha crecido de forma exponencial en los últimos años. Las empresas recopilan todo tipo de datos personales de los clientes para conocerlos en profundidad y poder adaptar sus productos y servicios a los deseos y necesidades de los mismos. En definitiva, para mejorar la experiencia del usuario. Sin embargo, esta información atrae la atención de los ciberdelincuentes, los cuales buscan robar esta información para su posterior venta en el mercado negro.

La información de identificación personal se clasifica según la capacidad para identificar a una persona y el nivel de confidencialidad de la misma. Según el primer nivel de clasificación, se encuentran los siguientes dos grupos:

- *Identificadores.* Aquellos atributos que identifican inequívocamente a una per-

sona o entidad (ej. pasaporte, usuario de una red social, NSS, etc). Por lo general, antes de compartir un conjunto de datos, dichos atributos se eliminan, para que el resto de atributos de una tupla no se relacionen directamente con un individuo.

- *Cuasi-identificadores*. Son atributos que no permiten una identificación directa con un individuo pero que, en conjunto con otros cuasi-identificadores, pueden llegar a señalar directamente a una persona (ej. dirección, sexo, edad). Además dichos atributos pueden enlazarse con otras fuentes de datos que no son anónimas, este proceso se llama reidentificación y nos permite obtener información personal a partir de atributos cuasi-identificadores. A diferencia que con los atributos identificadores, estos no pueden ser eliminados, ya que son una fuente clave de información, por lo que es esencial llevar a cabo una correcta protección de estos.

Respecto al nivel de confidencialidad de la información, se puede dividir los datos en dos grupos diferenciados:

- *Confidenciales*. Como su propio nombre indica, un atributo confidencial es aquel que contiene información sensible para un individuo (ej. datos bancarios, historial médico). Pueden ser considerados cuasi-identificadores y por su naturaleza, necesitan una protección mayor.
- *No confidenciales*. Un atributo no confidencial es aquel que de forma aislada, no causarían un daño significativo a la persona en caso de filtración o robo. Pueden o no ser exclusivos de una persona (ej. Lugar de nacimiento, Raza, Religión, alias de una red social). Por lo general, es información disponible a nivel público.

No obstante, considerar un dato confidencial o no confidencial puede depender en gran medida del contexto.

## 2.3. Tipos de datos

Los datos pueden ser almacenados y compartidos de distintas formas, en el campo de la privacidad se distinguen dos formas diferenciadas, aunque pueden encontrarse más: macrodatos y microdatos.

Los macrodatos representan, por lo general, valores estimados sobre ciertas características referentes a un conjunto de personas. Por ejemplo, queremos almacenar una serie de datos médicos estimados sobre una población diferenciada por sexo. En conclusión son datos que a diferencia de los microdatos, no se pueden relacionar con un individuo.

Por el contrario, los microdatos son atributos los cuales hacen referencia a una característica individual sobre algo o alguien. Una forma clásica de representar microdatos es como tablas formadas por tuplas que contienen valores para dichos atributos. A diferencia de con los macrodatos, los microdatos permiten realizar estudios más precisos e individuales. No obstante y como dichos datos pueden enlazarse a un individuo,

surge la necesidad de proteger dichos datos, de tal forma que a la hora de compartir dicha información, no se revelen datos sensibles. Para ello es necesario que los microdatos se sometan a procesos de enmascaramiento o anonimización, los cuales se detallarán más adelante. Desde este momento, se utilizará el término *datos* para hacer referencia los a microdatos.

Los atributos de un conjunto de datos pueden clasificarse según su naturaleza, es decir, el tipo de dato en sí. Existen dos tipos diferentes:

- *Numéricos*. Un atributo se considera numérico cuando, trivialmente es un número y además, se pueden realizar operaciones aritméticas sobre él. Pueden ser a su vez valores **continuos** (ej. peso de una persona) o **discretos** (ej. edad de un individuo).
- *Categóricos*. Un atributo es considerado categórico cuando no permite realizar operaciones aritméticas sobre él. No obstante, pueden ser **ordinales**, cuando sus valores representan categorías con alguna clasificación intrínseca (ej. puntuaciones sobre una encuesta de satisfacción). O **nominales**, que no permiten ninguna clasificación intrínseca (ej. ciudad de nacimiento de un individuo). La mayoría de las cadenas de texto almacenadas como datos son nominales.

Además de clasificarse por el tipo de dato, el Reglamento General de Protección de Datos (RGPD) propone la agrupación de los atributos de un conjunto de datos en distintas categorías, en función de características comunes o tratamiento similar. Aunque no existe ninguna clasificación bien definida, en la literatura se propone la siguiente [1]:

- *Identificativos* Comentados en la sección anterior, son aquellos atributos capaces de identificar a una persona de forma inequívoca. Tales como, DNI/NIF, firma digital, número de la seguridad social, etc.
- *Características personales*. Sexo, raza, estado civil, lengua materna, estatura, peso, orientación sexual, gustos o aficiones, datos relativos a la salud.
- *Circunstancias familiares y sociales*. Número y edad de hijos, permisos y licencias, inscripciones en clubes o asociaciones, ayudas y subvenciones, permisos de residencia.
- *Laborales*. Situación laboral, empresa o departamento asociado, datos no económicos de nóminas, datos corporativos.
- *Académicos y profesionales*. Formación y titulaciones, experiencia profesional, historial de estudiante.
- *Judiciales y administrativos*. Procedimientos administrativos, recursos y reclamaciones, sanciones, historial judicial.
- *Económicos y financieros*. Ingresos, rentas, inversiones, créditos, préstamos, bienes patrimoniales, planes de pensión, datos de actividad económica.
- *Información comercial*. Actividades y negocios, subscripciones a sitios web, licencias comerciales, creaciones científicas o técnicas.

- *De ámbito penal.* Certificado de antecedentes penales, certificado de delitos de naturaleza sexual, demandas y sentencias penales.

No obstante, esta es un clasificación de las muchas posibles. Al fin y al cabo, el RGPD otorga libertad a las organizaciones para establecer dicha categorización.

## 2.4. Modelos de privacidad

A diferencia de las técnicas enmascaramiento de microdatos expuestas en secciones futuras que ofrecen garantías de privacidad a posteriori. Los modelos de privacidad establecen de antemano las condiciones que deben cumplir los datos protegidos para garantizar un nivel mínimo de anonimato. Estas garantías de privacidad pueden alcanzarse para un conjunto de datos haciendo uso de uno o varios de los métodos de enmascaramiento descritos en anteriores secciones. Generalmente, los modelos de privacidad dependen de uno o varios parámetros que determinan el riesgo de reidentificación aceptable. Los modelos de privacidad existentes se han desarrollado principalmente para un único conjunto de datos estáticos. Sin embargo, en los últimos años, con el aumento del big data esta configuración se está quedando obsoleta [50].

A continuación, describimos algunos de los principales modelos de privacidad y la posibilidad de combinarse para obtener una protección más robusta. También, detallaremos qué métodos de enmascaramiento están directamente relacionados y deben utilizarse para aplicar un modelo de privacidad concreto.

### 2.4.1. *k*-anonimidad

La *k*-anonimidad es un modelo de privacidad, que tiene como objetivo, evitar la reidentificación de los individuos a los que se refieren los datos. Este modelo permite cuantificar y aplicar un determinado grado de anonimidad a dicha información confidencial[47]. La Figura 2.1 muestra un posible caso de reidentificación a partir de dos bases de datos ajena, una de ellas (2.1a) contiene información médica sensible, sobre la cual se han suprimido los atributos identificadores. La otra base de datos (2.1b), contiene información genérica sobre diferentes individuos, pero no cuenta con ningún nivel de privacidad. En caso de que un atacante tenga acceso a ambas bases de datos, este podrá realizar un cruce con los campos de ambas y descubrir, con un alto nivel de fiabilidad, que *David Calvo Pavón* padece de hipertensión.

La base de la *k*-anonimidad es hacer que la combinación de atributos cuasi-identificadores en un conjunto de datos,  $X$ , no sea única, haciéndolos indistinguibles dentro de los grupos de registros. Para cumplir esa garantía, es necesario que cada combinación de valores de los atributos cuasi-identificadores o confidenciales en el conjunto de datos, ya enmascarado  $X^*$ , sea compartida por  $k$  o más registros.

Por lo tanto, este modelo de privacidad garantiza que, para cualquier combinación de valores de los cuasi-identificadores existentes en  $X^*$ , hay al menos  $k - 1$  registros

que comparten la misma combinación. El conjunto de tuplas con igual valor en todos los atributos cuasi-identificadores recibe el nombre de *clase de equivalencia*[25].

En caso de que un atacante tenga acceso a un conjunto de datos sin anonimizar,  $Y$ , que contenga algunos de los atributos cuasi-identificadores existentes en  $X^*$ , no podrá vincular, inequívocamente, a un individuo existente en  $Y$  con una tupla de  $X^*$ . Como máximo, el atacante podrá identificar la clase de equivalencia que concuerde con el individuo que tiene como objetivo. Por lo tanto, existe una probabilidad de  $1/k$  de que se produzca una correcta reidentificación.

DNI	Nombre	Sexo	Edad	CP	Peso(kg)	Altura(cm)	Enfermedad
-	-	Hombre	68	11000	85	178	Hipertensión
-	-	Mujer	45	11026	57	171	Covid-19
-	-	Mujer	23	12100	53	162	Mononucleosis
-	-	Hombre	81	11055	72	173	Neumonía
-	-	Hombre	79	11000	121	181	Obesidad

(a) Registros de una base de datos con atributos identificadores eliminados

DNI	Nombre	Sexo	Edad	CP	Peso(kg)	Altura(cm)
... 49245955Y ...	... David Calvo Pavón ...	... Hombre ...	... 68 ...	... 11000 ...	... 85 ...	... 178 ...

(b) Registros de una base de datos sin ningún nivel de privacidad

Figura 2.1: Ejemplo de reidentificación a partir de bases de datos ajenas.

Para generar conjuntos de datos que satisfagan este modelo de privacidad, la bibliografía consultada propone dos enfoques diferentes. El primero de ellos es propuesto por P. Samarati en [45, 46] y consiste en el uso combinado de dos métodos de enmascaramiento no perturbativos: generalización, visto en la sección 2.5.1.1 y supresión, el cual comentaremos a continuación. Por un lado, mediante la generalización de los registros correspondiente a los atributos, conseguimos que el conjunto de  $k$  tuplas sea indistinguible. Sin embargo, hay ciertos atributos cuya generalización es muy complicada de realizar o directamente, no es posible. Por ello, este método se combina con la supresión, que consiste en eliminar aquellos registros atípicos para así reducir la cantidad de generalización.

Garantizar simultáneamente un correcto anonimato y la retención de una cantidad mínima de información, conlleva un alto coste computacional, de  $O(k \log k)$  en los mejores casos. Por ello, el valor de  $k$  utilizado en la práctica no suele ser superior a 5 o 6 [36].

Por ello, surge un segundo enfoque más práctico basado en la microagregación, sección 2.5.2.1. Aplicando esta técnica perturbativa a un conjunto de tuplas de tamaño  $k$ , el resultado obtenido satisface la  $k$ -anonimidad. Sin embargo, aunque de esta forma reducimos el coste computacional, la protección contra la divulgación de atributos confidenciales no está garantizada. Existen atributos, en los que los valores de los  $k$

registros de una clase de equivalencia son iguales o muy similares, por lo que un atacante que identifique a un individuo, dentro de una clase de equivalencia, puede obtener el valor real de un atributo confidencial del mismo [25].

Para evitar dicha posible divulgación de atributos confidenciales, han surgido métodos que deben de combinarse con la  $k$ -anonimidad. Uno de estos métodos se denomina  $l$ -diversidad, propuesto en [34]. Por definición, un conjunto de tuplas es  $l$ -diverso si contiene al menos  $l$  valores distintos para el atributo sensible  $S$ . En consecuencia, un conjunto de datos  $X^*$  es  $l$ -diverso, si cada grupo de  $k$  tuplas cumple con lo anterior.

Edad	CP	Enfermedad
40-70	280**	Gastroenteritis
40-70	280**	Covid-19
40-70	280**	Covid-19
>70	4100*	Neumonía
>70	4100*	Cáncer
>70	4100*	Covid-19
20-45	280**	Mononucleosis
20-45	280**	Mononucleosis
20-45	280**	Mononucleosis

Figura 2.2: Conjunto de microdatos 3-anónimo

Edad	CP	Enfermedad
$\geq 20$	280**	Gastroenteritis
$\geq 20$	280**	Mononucleosis
$\geq 20$	280**	Covid-19
>70	4100*	Neumonía
>70	4100*	Cáncer
>70	4100*	Covid-19
> 20	280**	Mononucleosis
> 20	280**	Covid-19
> 20	280**	Mononucleosis

Figura 2.3: Conjunto de microdatos 2-diverso

Como ejemplo, vemos que la tabla de la Figura 2.2 cumple con la  $k$ -anonimidad, en este caso para  $k = 3$ . Sin embargo, presenta el problema comentado anteriormente para algunas clases de equivalencia. Una posible solución es la que vemos en la Figura 2.3, aplicando  $l$ -diversidad para  $l = 2$ . Dificultando la divulgación del atributo confidencial *Enfermedad*, conociendo los valores del resto de atributos para un individuo.

Sin embargo, hay situaciones en las que un atributo confidencial tiene un dominio muy reducido, por lo que los valores para dicho atributo se repetirían numerosas veces en la tabla, dificultando así, la técnica anterior.

Existe otra técnica, denominada  $t$ -closeness [32]. Una clase de equivalencia tiene  $t$ -closeness si la distancia entre la distribución de un atributo confidencial en dicha

clase y la distribución del atributo en todo el conjunto de datos no es mayor que un valor umbral  $t$ . En consecuencia, un *dataset* tiene  $t$ -closeness si todas las clases de equivalencia cumplen con lo anterior.

En conclusión, la  $k$ -anonimidad debe combinarse con alguna de las medidas citadas para proporcionar protección contra la divulgación de atributos sensibles y poder ofrecer una mayor garantía de privacidad.

#### 2.4.2. $k$ -anonimidad probabilística

La  $k$ -anonimidad probabilística es un modelo de privacidad de odatos que ofrece las mismas garantías de protección, ante una posible reidentificación que la  $k$ -anonimidad. La probabilidad de reidentificación, al igual que en la  $k$ -anonimidad, es de  $1/k$ . No obstante, la  $k$ -anonimidad consigue estos resultados haciendo indistinguibles los valores de  $k$  registros para los atributos cuasi-identificadores. Esto, en numerosas ocasiones, puede producir un exceso de perdida de información.

La  $k$ -anonimidad probabilística, por el contrario, no exige que las combinaciones de valores de cuasi-identificadores y atributos sensibles en el conjunto de datos a liberar,  $X^*$ , sean iguales dentro de los grupos de registros de tamaño  $k$ . Lo que lo hace un modelo menos restrictivo en comparación con la  $k$ -anonimidad.

Al rebajar los requisitos de indistinguibilidad dentro de los grupos de  $k$  registros, la gama de métodos a elegir para proteger el conjunto de datos,  $X$ , es más amplia, y por lo tanto podemos esperar una reducción de la pérdida de información. Es un modelo ideal para conjuntos de datos de gran tamaño que constan con un elevado número de atributos cuasi-identificadores o sensibles.

Cualquier conjunto de datos,  $X^*$  que cumpla con la  $k$ -anonimidad, a su vez, cumple con la  $k$ -anonimidad probabilística. Sin embargo, la  $k$ -anonimidad probabilística no implica la  $k$ -anonimidad. Por lo tanto, cualquier método de enmascaramiento que sirva para hacer cumplir con la  $k$ -anonimidad, sirve para hacer cumplir con este modelo. En este sentido, puede decirse que la  $k$ -anonimidad proporciona una garantía mayor. Sin embargo, desde el punto de vista de la probabilidad de reidentificación, ambos proporcionan el mismo nivel de protección.

El rank swapping, presentado en la sección 2.5.2.3, es la técnica de enmascaramiento ideal para satisfacer la  $k$ -anonimidad probabilística. Un posible atacante que tenga acceso a un atributo  $S$ , que ha sido protegido, cuyos valores han sido intercambiados en intervalos que abarcan a  $k$  registros, sólo podría inferir los valores originales con una probabilidad máxima de  $1/k$ . Todo esto, sin reducir la riqueza y variedad del conjunto de datos original.

#### 2.4.3. Privacidad $\varepsilon$ -diferencial

Al igual que se comentó al inicio, el incremento exponencial de los datos que se almacenan ha convertido, la privacidad de estos, en una preocupación fundamental. En muchas ocasiones, las técnicas tradicionales de anonimización, como la eliminación de

identificadores directos, no siempre son suficientes para garantizar la privacidad. En este contexto, cobra gran importancia la privacidad  $\varepsilon$ -diferencial, un modelo de privacidad que surgió como un marco matemático para proteger la privacidad de los datos de forma cuantificable. El concepto matemático de privacidad  $\varepsilon$ -diferencial se remonta a la década de 1970, con el trabajo de Cynthia Dwork y Moni Naor [20], aunque no fue hasta los años 2000 cuando comenzó a ganar popularidad. Sin embargo, es una técnica muy empleada en el mundo empresarial actual, lo que ha favorecido la investigación y la evolución de la misma. Algunos ejemplos de empresas que la incorporan en su día a día son las siguientes:

- **Apple:** La utiliza para proteger los datos que recogen de sus usuarios en sus dispositivos iOS, que se utilizan para posteriores análisis.
- **Google:** La utiliza para mejorar la precisión de sus algoritmos de aprendizaje automático sin comprometer la privacidad de los usuarios.
- **Microsoft:** La utiliza para ofrecer servicios de análisis de datos en la nube que preservan la privacidad.

La aplicación de la privacidad  $\varepsilon$ -diferencial puede ayudar a las empresas a proteger su reputación al demostrar que están comprometidas con la privacidad de sus clientes y empleados. Además, permite cumplir con algunas normas de privacidad de datos, como el RGPD. En definitiva, puede ser una ventaja competitiva para las empresas que se preocupan por la privacidad de sus clientes y empleados.

A lo largo de esta sección, se profundizará en el concepto de privacidad diferencial, en concreto privacidad  $\varepsilon$ -diferencial, ambos términos se usan a menudo indistintamente. Sin embargo, es importante tener en cuenta la distinción entre ellos, privacidad  $\varepsilon$ -diferencial ( $\varepsilon$ -DP) es un término más específico que se utiliza para describir un tipo particular de privacidad diferencial que se caracteriza por un nivel específico de ruido añadido a los datos. Se expondrá el objetivo de este modelo de privacidad, su definición formal y términos claves dentro de la misma, diferentes características y un pequeño análisis en el que mostraré algunas ventajas y desventajas frente a otros modelos de privacidad existentes, algunos de ellos expuestos en secciones anteriores.

Cuando nos referimos a la privacidad  $\varepsilon$ -diferencial, no estamos hablando de un algoritmo específico, la privacidad  $\varepsilon$ -diferencial es una definición, un acuerdo sobre un objetivo de privacidad y las condiciones matemáticas que deben cumplirse para alcanzar ese objetivo [41]. La forma concreta de implementación puede variar según el caso, y un mismo caso puede resolverse utilizando diferentes algoritmos. Lo importante es que todo el sistema se mantenga dentro del marco definido por esta.

Originalmente, el modelo de privacidad  $\varepsilon$ -diferencial se propuso como una garantía de privacidad para bases de datos sobre la que se realizan consultas, generalmente de conteo. En este entorno de respuesta a consultas interactivas, la privacidad  $\varepsilon$ -diferencial establece las condiciones que las respuestas deben cumplir para que el riesgo de divulgación esté bajo control. Para cumplir con lo anterior, se necesita un software intermedio, denominado *guarda de privacidad*, encargado de lograr la privacidad  $\varepsilon$ -diferencial. Por tanto, los datos originales no se verían alterados, sino que se actúa sobre el proceso de consulta y publicación de los datos analizados. La figura 2.4 muestra el proceso de consulta descrito anteriormente. En un primer paso, diferen-

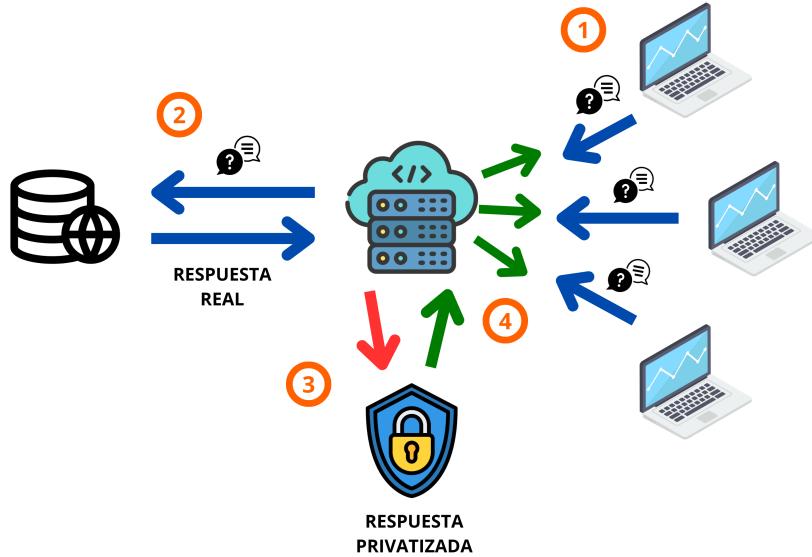


Figura 2.4: Proceso de obtención de consultas  $\varepsilon$ -diferencialmente privadas

tes clientes o analistas realizan consultas sobre un servidor. Este servidor recibe una consulta, y obtiene la respuesta real directamente desde la base de datos. Posteriormente, la respuesta real pasa por un proceso de enmascaramiento, que altera el valor real mientras que se mantiene la utilidad de la información. Finalmente, la respuesta privatizada se comparte con el usuario que originó la consulta.

La privacidad  $\varepsilon$ -diferencial funciona añadiendo ruido estadístico a los datos existentes, bien a los datos iniciales o a los datos de salida de una consulta, esta fue la idea original. La utilidad estadística de los datos modificados se sustenta gracias a la Ley de los Grandes Números [40], un principio estadístico que indica que a medida que aumenta el tamaño de la muestra, los valores promedio se acercan al valor real de la información. Así, al agregar ruido aleatorio a grandes volúmenes de datos, se compensan estos efectos y se genera un valor que es esencialmente equivalente al original.

Un ejemplo para mostrar lo anterior puede verse en la figura 2.5, a partir de unos datos clínicos sobre características tumorales. He añadido una nueva columna con los mismos datos recogidos que en la columna *fractal\_dimension*, los cuales han sido modificados añadiéndole una secuencia de ruido gaussiano. Como se puede apreciar en el recuadro de color naranja, los valores estadísticos de la media y la desviación típica obtenidos son muy similares pese a que los datos son completamente diferentes.

En definitiva, la privacidad  $\varepsilon$ -diferencial se apoya en el empleo de funciones matemáticas, conocidas como **mecanismos** que añaden ruido aleatorio a los datos de salida del sistema. El parámetro  $\varepsilon$ , **presupuesto de privacidad**, establece el equilibrio entre la precisión en el resultado obtenido y la privacidad de la información consultada [25]. Esta técnica nos permite cuantificar, a diferencia de otros modelos de privacidad, la perdida de privacidad producida y la precisión del análisis.

El principio subyacente de la privacidad  $\varepsilon$ -diferencial es que la presencia o ausencia de cualquier individuo único en la base de datos no puede ser detectable al analizar los resultados de las consultas. Supongamos que contamos con una base de datos  $X$  y una segunda,  $X'$ , similar a la primera pero con la exclusión de un registro, lo que se

id	radius_mean	perimeter_mean	area_mean	concavity_mean	fractal_dimension	fractal_dimension_NOISE
543	921386	14,47	95,81	65,64	0,1009	0,06341
544	921644	14,74	9,47	66,86	0,4105	0,00568
545	922296	13,21	84,88	53,84	0,2987	0,05781
546	922297	13,87	89,77	58,48	0,3688	0,055451429
547	922576	13,62	87,19	57,32	0,2974	0,05801
548	922577	10,32	65,31	32,49	0,1012	0,06201
549	922840	10,26	65,85	32,08	0,4358	0,06714
550	923169	96,83	61,05	28,57	0,2337	0,06235
551	923465	10,82	68,89	36,16	0,1548	0,06328
552	923748	10,86	68,51	36,05	0	0,05948
553	923780	11,13	71,49	37,84	0,4824	0,06552
554	924084	12,77	81,35	50,79	0,1997	0,05637
555	924342	93,33	59,01	2,64	0,3996	0,06576
556	924632	12,88	8,25	51,43	0,6195	0,05708
557	924934	10,29	65,67	32,14	0,5999	0,06127
558	924964	10,16	64,73	31,17	0,5025	0,06331
559	925236	94,23	59,26	27,13	0	0,06059
560	925277	14,59	96,39	65,71	0,1029	0,06147
561	925291	11,51	74,52	40,35	0,1112	0,00657
562	925292	14,05	91,38	60,04	0,4462	0,06171
563	925311	1,12	70,67	3,86	0	0,05502
564	925622	15,22	10,34	71,69	0,0255	0,07152
565	926125	20,92	1,43	13,47	0,3174	0,06879
566	926424	21,56	1,42	14,79	0,2439	0,05623
567	926682	20,13	13,12	12,61	0,0144	0,05533
568	926954	1,66	10,83	85,81	0,9251	0,05648
569	927241	2,06	14,01	12,65	0,3514	0,07016
570	92751	7,76	47,92	1,81	0	0,05884
571						
572					MEDIA	0,056700228
573					DESV. ESTÁNDAR	0,018816661
						0,05717814
						0,019930959

Figura 2.5: Dataset de características tumorales al que se le ha añadido una característica ruidosa (fractal\_dimension\_NOISE). Dataset original: Breast Cancer Wisconsin (Diagnostic) - Extraido de <https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data/data>

conoce como conjuntos de datos **adyacentes**. En este punto, se detalla la definición de privacidad  $\varepsilon$ -diferencial [20].

**Definición 1.** Sea  $\varepsilon$  un número real positivo y  $\eta$  una función aleatoria que recibe como entrada el conjunto de datos  $X$ . La salida,  $\eta(X)$ , de una consulta cualquiera, proporciona privacidad  $\varepsilon$ -diferencial si,  $\forall X, X'$ , conjuntos de datos adyacentes y cualquier subconjunto  $S$  del dominio de  $\eta$ , se cumple que

$$P(\eta(X) \in S) \leq e^\varepsilon \times P(\eta(X') \in S) \quad (2.1)$$

Se puede decir que la privacidad  $\varepsilon$ -diferencial garantiza que la información que se puede extraer de la respuesta a una consulta cualquiera está limitada por un factor de  $e^\varepsilon$ . Un ejemplo de esto se puede ver en la figura 2.6, es el mismo conjunto de datos que en la figura 2.5 a excepción de un registro, el registro cuyo identificador es 925291 (subrayado en rosa). Como podemos ver, las características estadísticas del nuevo conjunto de datos apenas varía respecto al caso anterior. Desde el punto de vista del atacante sería difícil extraer información relevante a partir de la misma consulta realizada sobre cada conjunto de datos.

Un concepto importante en la privacidad  $\varepsilon$ -diferencial es la **sensibilidad global** [3], la cual expresa el peso concreto de un registro sobre el resultado de una consulta,  $f$ . En el caso de valores numéricos, se calcula como la diferencia máxima entre dos conjuntos de datos adyacentes. Formalmente se puede expresar como

$$\Delta f = \max \|f(X) - f(X')\| \quad (2.2)$$

	<b>id</b>	<b>radius_mean</b>	<b>perimeter_mean</b>	<b>area_mean</b>	<b>concavity_mean</b>	<b>fractal_dimension</b>	<b>fractal_dimension_NOISE</b>
543	921386	14,47	95,81	65,64	0,1009	0,06341	0,071981429
544	921644	14,74	9,47	66,86	0,4105	0,00568	0,011394286
545	922296	13,21	84,88	53,84	0,2987	0,05781	0,069238571
546	922297	13,87	89,77	58,48	0,3688	0,06688	0,061165714
547	922576	13,62	87,19	57,32	0,2974	0,05801	0,046581429
548	922577	10,32	65,31	32,49	0,1012	0,06201	0,049152857
549	922840	10,26	65,85	32,08	0,4358	0,06714	0,064282857
550	923169	96,83	61,05	28,57	0,2337	0,06235	0,048064286
551	923465	10,82	68,89	36,16	0,1548	0,06328	0,054708571
552	923748	10,86	68,51	36,05	0	0,05948	0,06948
553	923780	11,13	71,49	37,84	0,4824	0,06552	0,071234286
554	924084	12,77	81,35	50,79	0,1997	0,05637	0,043512857
555	924342	93,33	59,01	2,64	0,3996	0,06576	0,067188571
556	924632	12,88	8,25	51,43	0,6195	0,05708	0,051365714
557	924934	10,29	65,67	32,14	0,5999	0,06127	0,048412857
558	924964	10,16	64,73	31,17	0,5025	0,06331	0,06331
559	925236	94,23	59,26	27,13	0	0,06059	0,067732857
560	925277	14,59	96,39	65,71	0,1029	0,06147	0,060041429
561	925292	14,05	91,38	60,04	0,4462	0,06171	0,073138571
562	925311	1,12	70,67	3,86	0	0,05502	0,040734286
563	925622	15,22	10,34	71,69	0,0255	0,07152	0,065805714
564	926125	20,92	1,43	13,47	0,3174	0,06879	0,063075714
565	926424	21,56	1,42	14,79	0,2439	0,05623	0,06623
566	926682	20,13	13,12	12,61	0,0144	0,05533	0,049615714
567	926954	1,66	10,83	85,81	0,9251	0,05648	0,045051429
568	927241	2,06	14,01	12,65	0,3514	0,07016	0,083017143
569	92751	7,76	47,92	1,81	0	0,05884	0,073125714
570							
571					<b>MEDIA</b>	<b>0,056788486</b>	<b>0,057125508</b>
572					<b>DESV. ESTÁNDAR</b>	<b>0,018714999</b>	<b>0,020643307</b>

Figura 2.6: Dataset de características tumorales al que se le ha añadido ruido tras eliminar el registro con el registro con identificador 92529. Dataset original: Breast Cancer Wisconsin (Diagnostic) - Extraido de <https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data/data>

para todo  $X, X'$  que difieren en un registro.

La cantidad de ruido que añadirá a la consulta el mecanismo  $\eta$  viene condicionado por el valor la sensibilidad global. Por lo general, la privacidad  $\varepsilon$ -diferencial obtiene mejores resultados en tipos de análisis con baja sensibilidad global. Esto se debe a que al agregar ruido, se puede preservar la privacidad de los datos sin distorsionar el valor real del resultado de la consulta por encima del umbral de utilidad establecido.

#### 2.4.3.1. Mecanismos $\varepsilon$ -diferencialmente privados

Generalmente en privacidad  $\varepsilon$ -diferencial, se utilizan mecanismos de adición de ruido. Una vez obtenida la respuesta a la consulta  $f$ , se añade ruido aleatorio a  $f(X)$  para privatizar la respuesta. Posteriormente, se devuelve al usuario que envió la consulta  $f$  la respuesta con ruido  $\eta(X) = f(X) + \gamma$ , siendo  $\gamma$  el ruido.

Se pueden emplear diferentes distribuciones para generar el ruido, por lo general, la distribución escogida viene determinada según el tipo de datos o el valor de la sensibilidad asociado a la consulta.

Entre alguna de las distribuciones más utilizadas se encuentra la distribución de Laplace y su versión discreta [25].

**Definición 2.** Dada cualquier consulta  $f : Z^n \rightarrow R^k$  dependiente de los datos, definimos el mecanismo Laplaciano como:

$$\eta_{Lap}(X, f, \varepsilon) = f(X) + (Y_1, \dots, Y_k) \quad (2.3)$$

siendo  $Y_i$  variables independientes e idénticamente distribuidas que distribuyen  $Lap(0, \Delta f/\varepsilon)$ .

El valor de la sensibilidad determina cuánto ruido Laplaciano se agrega para valores de  $\varepsilon$  prefijados. Se requiere más ruido cuando el resultado de la función  $f$  varía en alto grado para dos conjuntos de datos adyacentes. Para valores de  $\varepsilon$  muy pequeños,  $\eta(X)$  debe producir valores muy similares para todos los pares de conjuntos de datos adyacentes, lo cual se logra agregando mucho ruido.

Existen otros mecanismos utilizados en privacidad  $\varepsilon$ -diferencial que se alejan de la adición de ruido estadístico. Uno de ellos se conoce como respuesta aleatoria (randomized response). Este mecanismo cobra sentido para preguntas de sí o no, por ejemplo, para una encuesta de contenido político. Se fija una probabilidad,  $p$ , de la que dependerá que la respuesta de un usuario a una pregunta concreta, sea falsa. Realmente, es otro método que añade ruido a las respuestas reales del usuario. Por lo general,  $p$ , se fija a partir de la siguiente ecuación:

$$p = \frac{e^\varepsilon}{1 + e^\varepsilon} \quad (2.4)$$

No obstante, este mecanismo puede introducir sesgo en los datos. Si la probabilidad de dar una respuesta veraz es demasiado baja, por ejemplo, los datos podrían no representar adecuadamente a la población.

#### 2.4.3.2. Privacidad $\varepsilon$ -diferencial local

Originalmente la privacidad  $\varepsilon$ -diferencial se diseño para bases de datos sobre las que una entidad realiza consultas. Este el caso mostrado a lo largo de la sección, el sistema adopta la privacidad  $\varepsilon$ -diferencial desde una perspectiva **global** y siempre existe un repositorio donde se encuentran los datos reales, sin ruido. Es decir, este mecanismo depende de un recolector externo confiable.

De esta forma, existe un riesgo de acceso o uso malintencionado a los datos limpios, por lo que se necesita hacer un mayor esfuerzo en su seguridad y restringir la forma en que el sistema pueda compartirse con terceros.

Sin embargo, en los últimos años debido al aumento en la compartición de datos y la ciberdelincuencia y con el objetivo de de eliminar la confianza en el recolector, se está abordando la privacidad  $\varepsilon$ -diferencial desde un enfoque más **local**. Dejando de existir, por tanto, una versión de los datos reales. Lo que se recopila, centraliza y analiza es siempre la versión ruidosa de los datos, pasada por el algoritmo correspondiente.

Por lo tanto, los datos originales solo son accesibles para los usuarios y el recolector solo recibe los datos perturbados. Un mecanismo  $M$  que cumple con privacidad  $\varepsilon$ -diferencial local se puede definir de la siguiente manera [33].

**Definición 3.** Sea  $\varepsilon > 0$  y  $M$ , una función aleatoria.  $M$  proporciona  $\varepsilon$ -privacidad  $\varepsilon$ -diferencial local, si y solo si, para dos conjuntos de datos adyacentes cualesquiera,  $X$  y  $X'$  y la salida  $y$ , se cumple que

$$\frac{P(M(X) = y)}{P(M(X') = y)} \leq e^\varepsilon \quad (2.5)$$

Existen otras estrategias de privacidad  $\varepsilon$ -diferencial local que proponen la creación de un conjunto de datos  $\varepsilon$ -diferencialmente privado a partir de histogramas ruidosos. Se construye un histograma de un atributo dividiendo su dominio en subconjuntos mutuamente disjuntos y obteniendo las frecuencias en cada subconjunto. Posteriormente, se añade ruido discreto para evitar revelar información real sobre los datos.

Otro popular enfoque consiste en agregar ruido Laplaciano a los valores de atributo reales. Para disminuir el ruido, al conjunto de datos se le aplica, previamente, una microagregación. De esta forma, los valores originales se reemplazan por promedios, que tienen una sensibilidad más baja.

La privacidad  $\varepsilon$ -diferencial es un modelo de privacidad que cuenta con numerosas ventajas frente a otros modelos existentes, algunos de ellos comentados en secciones anteriores. A diferencia de otros modelos, permite cuantificar la pérdida de privacidad, lo que establece una compensación entre la pérdida de privacidad y la precisión de la información. Además, proporciona una protección individual más fuerte al garantizar que el riesgo de identificación de cualquier individuo en el conjunto de datos sea limitado, independientemente de cuántos registros adicionales se agreguen o se quiten del conjunto de datos.

Sin embargo, la pérdida de privacidad producida al realizar una consulta sobre un conjunto de datos es **acumulativa**. Es decir, si un atacante puede realizar la misma consulta varias veces sobre la misma base de datos, la incertidumbre que proporciona el mecanismo aleatorio se reduce. Cuantas más veces pueda repetir la consulta, mayor certeza tendrá sobre el valor original y mayor será la probabilidad de acierto. En ese caso, surge la necesidad de imponer restricciones o límites en el numero de consultas que se pueden hacer. Empresas como, por ejemplo Apple [15], limitan el número de contribuciones diarias que un usuario puede realizar para añadir un nivel extra de protección.

Otro problema de la privacidad  $\varepsilon$ -diferencial viene derivado de la adición de ruido, puesto que puede disminuir la efectividad de los datos al hacerlos menos precisos o útiles para ciertos tipos de análisis. Manejar esto puede resultar complicado y requiere equilibrar con cuidado la privacidad y la utilidad. También es un desafío la falta de estandarización y acuerdo sobre las mejores prácticas. No existe actualmente un enfoque estandarizado para implementar la privacidad  $\varepsilon$ -diferencial, lo que puede dificultar su implantación en muchas empresas.

## 2.5. Técnicas de enmascaramiento de datos

A partir de lo extraído en las secciones anteriores, se puede apreciar que existen ciertos atributos que deben ser protegidos previamente a ser compartidos, da tal forma que no se puedan relacionar directamente con una entidad y en caso de contener información sensible, esta no pueda ser revelada. No obstante, el conjunto de datos, tras pasar un proceso de anonimización, debe seguir siendo útil para análisis. La dificultad de proteger un conjunto de datos es obtener un equilibrio entre anonimizar los datos para reducir una posible reidentificación y que estos datos sigan teniendo utilidad.

Aunque profundizaremos en los de enmascaramiento, los métodos de privacidad de datos pueden clasificarse en dos tipos:

- *Métodos de enmascaramiento.* Aquellos en los que los datos originales son transformados en nuevos datos que, a la vez que siguen siendo válidos para estudios, mantienen la confidencialidad necesaria. En función de cómo se modifica el conjunto de datos, pueden denominarse métodos **no perturbativos** o **perturbativos**.
- *Métodos sintéticos.* Al contrario que los anteriores, los métodos sintéticos generan, con la ayuda de modelos estadísticos, un nuevo conjunto de datos  $\mathbf{V}'$  a partir del conjunto original  $\mathbf{V}$ , el cual conserva las propiedades estadísticas de  $\mathbf{V}$ . Al no provenir directamente del conjunto original, se reduce el problema de la reidentificación.

### 2.5.1. Técnicas de enmascaramiento no perturbativas

Los métodos de enmascaramiento no perturbativos son aquellos que no se basan en la distorsión o modificación de los datos originales, sino en supresiones parciales de datos o eliminación de detalles, preservando siempre la veracidad de estos. Una de las principales técnicas de enmascaramiento de este tipo es la Generalización.

#### 2.5.1.1. Generalización

La generalización consiste en la representación de los valores para un atributo como valores más generales. En esta técnica el concepto de *jerarquía de generalización* es clave, dado un dominio de un atributo, se crea una jerarquía cuya raíz es el valor más general posible y en las hojas de la jerarquía encontramos valores específicos. A la hora de generalizar un conjunto de datos, hay que ir sustituyendo los valores que tengamos en dicho momento por valores que estén un nivel por encima en la jerarquía. Este proceso puede repetirse en función del número de niveles de la jerarquía, consiguiendo así, varios niveles de generalización.

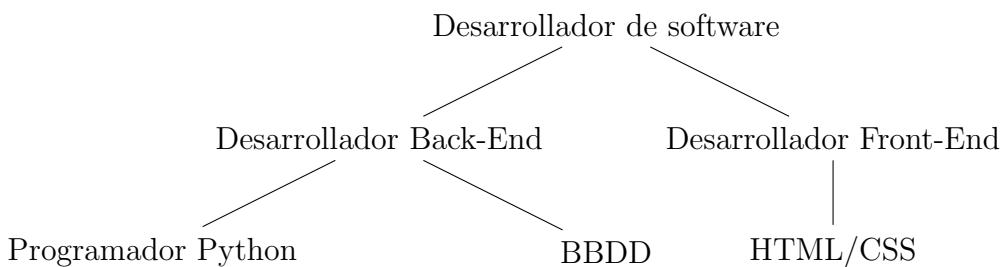


Figura 2.7: Jerarquía de generalización del atributo *puesto de trabajo*.

Supongamos que tenemos una tabla que almacena información acerca de los trabajadores de una empresa, la cual tiene personal cualificado en el ámbito de la informática. Dicho esquema, tiene un atributo *puesto de trabajo*, el cual actualmente

toma los valores de las hojas de la jerarquía referente a la Figura 2.7 Al aplicar un primer proceso de generalización, los valores se sustituirán por su correspondiente valor entre *Desarrollador Back-End* o *Desarrollador Front-End*. Si queremos, un enmascaramiento mayor, podemos volver a aplicar el proceso de generalización, sustituyendo cada valor actual en dicho campo por el valor más general, en este caso, *Desarrollador de software*.

Esta técnica es compatible para datos tanto categóricos como numéricos, aunque quizás sea más correcta de utilizar con los primeros. Por ello, existe otra técnica, que pueden considerarse un caso especial de generalización para valores numéricos, aunque en algunas bibliografías la clasifican como un método de enmascaramiento diferente. Dicha técnica llamada *recodificación global* consiste en, dado un dominio numérico para un atributo (ej. edades de individuos), este se divide en diversos intervalos, de tamaños similares, donde cada uno puede asociarse con un identificador. A la hora de generalizar el conjunto de datos, se sustituirán los valores actuales por el intervalo o identificador que le corresponda. Por ejemplo, en el caso de las edades, podríamos establecer los siguientes intervalos: [0-18 años] *Menor de edad*, [18-35 años] *Joven*, [35-65 años] *Adulto* y [65-99 años] *Jubilado*.

A partir de la definición anterior, surgen dos subtipos, para atributos continuos o categóricos que puedan ser ordenados linealmente. Se presentan a continuación:

- *Codificación superior o Top-Coding*. Se basa en la definición de un límite superior para el dominio de un atributo. De tal forma que a la hora de proteger los datos, cualquier valor superior a dicho límite será sustituido por él. Por ejemplo, si para un atributo establecemos que el límite superior es "10" en el esquema hay un valor para dicho atributo igual a "15", se sustituiría dicho valor por ">10".
- *Codificación inferior o Bottom-Coding*. Funciona igual que la codificación superior, pero usando un límite inferior en vez de uno superior, de tal forma que cualquier valor inferior a dicho límite, será sustituido por él.

Existen casos en los que podemos generalizar el mismo atributo a diferentes niveles, es importante destacar que a mayor nivel de generalización, la información pertinente pierde utilidad y se reduce, cada vez más, la posibilidad de reidentificación.

### 2.5.2. Técnicas de enmascaramiento perturbativas

En este tipo de métodos de enmascaramiento, los datos son modificados previamente a su publicación. A diferencia de los métodos no perturbativos, el conjunto de datos compartido puede contener registros con datos no veraces, fruto de introducir nuevas combinaciones. A continuación, se introducen algunos de los métodos perturbativos más utilizados.

### 2.5.2.1. Microagregación

La microagregación [14], originalmente diseñada para proteger valores continuos, consiste, primero, en agrupar las tuplas individuales en grupos de dimensión fija o variable. Generalmente las tuplas se organizan en grupos de un tamaño  $k$ , donde  $k \in \mathbb{N}$  es el llamado nivel de microagregación. Una vez los grupos están formados, para un atributo en concreto, se sustituye el valor actual de cada tupla en dicho atributo por un valor representativo del grupo. Cuando hablamos de un valor representativo, normalmente nos referimos a la media, no obstante, para la microagregación para valores categóricos se suelen utilizar otros métodos como la mediana.

Possiblemente el principal problema de este método sea formar los grupos de la forma más correcta posible, ya que estos tienen que ser lo más homogéneos posibles, en cuanto al valor del atributo que queremos proteger. Si agrupáramos una serie de tuplas con valores muy diferentes en el atributo que queremos enmascarar, a la hora de calcular el valor medio, este no sería un valor representativo para el grupo, por ello, los grupos se forman utilizando un criterio de máxima similitud. Aunque pueden definirse diversas funciones para medir la similitud, puede ser difícil encontrar una solución óptima. En las Figuras 2.8 y 2.9 vemos un ejemplo (las tuplas se agrupan por coincidencia en el atributo *Puesto de trabajo*).

Puesto de trabajo	Edad	Sueldo
Diseñador web	31	2700€
Administrador de BBDD	46	2950€
Diseñador web	51	3500€
Administrador de BBDD	30	2760€

Figura 2.8: Datos sobre los salarios de una empresa previo aplicar microagregación.

Puesto de trabajo	Edad	Sueldo
Diseñador web	31	3100€
Administrador de BBDD	46	2855€
Diseñador web	51	3100€
Administrador de BBDD	30	2855€

Figura 2.9: Datos de la Figura 2.8 tras aplicar microagregación sobre el atributo *sueldo*, agrupando los registros según el puesto de trabajo.

Dentro de la propia microagregación existen algunas variaciones, por ejemplo, según si el tamaño de los grupos es fijo o variable, podemos formar los grupos en función a otro atributo existente en la tabla (ej. Tenemos una tabla de una empresa y queremos enmascarar un atributo *sueldo*, podríamos agrupar las tuplas que concuerden en el atributo *puesto de trabajo*).

También existe una diferenciación según si la microagregación se realiza a uno o a varios atributos a la vez (univariante o multivariante). La primera realiza el procedimiento comentado anteriormente con cada atributo individualmente, mientras que la

segunda clasifica el conjunto de datos mediante ordenación multidimensional utilizando el primer componente principal del conjunto de datos, aquel atributo que está más correlacionado con la mayoría de los atributos originales del conjunto de datos y una vez hecho esto, se aplica la microagregación univariante a cada atributo.

Existen más variaciones, por ejemplo, el promedio puede sustituir el valor original sólo para una tupla del grupo o para todas ellas. Al inicio comentamos que esta técnica está pensada para valores continuos, no obstante, hay técnicas para valores categóricos, por ejemplo, a la hora de calcular el valor representativo, puede escogerse el valor que más aparece u otras técnicas estadísticas.

### 2.5.2.2. Adición de ruido

La adición de ruido engloba a un conjunto de métodos que funcionan de la siguiente forma, dado un conjunto de datos de entrada, por ejemplo, los valores de una tabla correspondientes a un atributo  $x$ , se le añade una secuencia de ruido aleatoria que por lo general, viene determinada por alguna distribución probabilística como Gauss o Laplace [25].

Las dos variaciones más populares de la adición de ruido son, la adición de ruido correlacionado y no correlacionado, profundizaremos en ellas más adelante. Es importante comentar que la secuencia de ruido, la cual viene representada por un vector de ruido, puede sumarse o multiplicarse a los valores que queremos enmascarar, es más común que el ruido se sume [6].

A continuación, definimos la adición de ruido no correlacionado y correlacionado y las principales diferencias entre ellas:

- *Adición de ruido no correlacionado.* Esta basada en añadir secuencias de ruido aleatorio, obtenidas a partir de la distribución normal, a los valores, de entrada, correspondientes a un atributo. Sea  $X_j$  la columna correspondiente al atributo  $j$ , la cual queremos proteger. Cada valor de dicha columna se sustituiría de la siguiente forma:

$$x_{ij} = x_{ij} + \epsilon_{ij} \quad (2.6)$$

Donde el vector de ruido  $\epsilon_j \approx N(0, \sigma_j^2)$  y  $\sigma_j^2 = \alpha \cdot \sigma_{X_j}^2$ , donde  $\sigma_{X_j}^2$  es la varianza de  $X_j$  y  $\alpha$  es un parámetro que determina la cantidad de ruido a añadir y suele tomar valores entre 0.1 y 0.5 [6, 25]. Como resultado obtenemos una secuencia de valores correspondientes a un atributo distorsionado, que conserva la media de los valores de entrada y mantiene una varianza proporcional a la original, mientras que los coeficientes de correlación no se conservan. Este método es mejor para enmascarar atributos individuales.

En la Figura 2.10 tenemos un conjunto de datos cuyo atributo *Peso* va a ser enmascarado mediante adición de ruido. Fijando los parámetros  $\sigma_{X_j}^2 = 221,81$  y  $\alpha = 0,1$ , obtenemos el siguiente vector de ruido aleatorio: [24.42, 42.45, 11.41, -5.50, 28.63, -12.40]. En la Figura 2.11 se muestran los datos ya enmascarados tras la suma del vector anterior.

Sexo	Edad	Peso(kg)
Hombre	65	76.5
Mujer	35	54.6
Mujer	62	59.8
Hombre	42	81.2
Hombre	51	91.5
Mujer	22	50.9

Figura 2.10: Conjunto de datos sintéticos con características de individuos, previo a la adición de ruido al atributo *Peso*.

Sexo	Edad	Peso(kg)
Hombre	65	100.9
Mujer	35	97.0
Mujer	62	71.2
Hombre	42	75.6
Hombre	51	120.1
Mujer	22	38.4

Figura 2.11: Conjunto de datos sintéticos con características de individuos, tras la adición de ruido al atributo *Peso*.

- *Adición de ruido correlacionado* [25]. Este método, como su nombre indica, permite añadir ruido aleatorio correlacionado a varios atributos en un conjunto de datos  $X$  con  $m$  atributos, de tal manera que:

$$X^* = X + \epsilon \quad (2.7)$$

De tal forma que,  $X, X^*$  y  $\epsilon$  son matrices de tamaño  $n \times m$ . La diferencia con el método anterior es que la matriz de covarianza de los datos distorsionados es proporcional a la matriz de covarianza de los datos originales en un factor  $1 + \alpha$ . También, dicho método preserva la media de cada atributo y mantiene el coeficiente de correlación de Pearson entre los atributos.

El enmascaramiento por ruido correlacionado produce datos enmascarados con mayor validez para el análisis que el enmascaramiento por ruido no correlacionado. Sin embargo, el ruido aditivo se utiliza raramente aislado debido al bajo nivel de protección que proporciona. Lo común en la práctica es combinar dichos métodos con transformaciones lineales o no lineales, según el tipo de los datos. Para que la adición de ruido se utilice eficazmente, hay que determinar en qué medida preserva la confidencialidad. Sin embargo, antes de poder hacerlo, hay que definir la confidencialidad y la integridad de los datos. El grado de protección mediante adición de ruido no correlacionado puede no ser tan alto como se podría pensar, sobre todo si la dimensión de los datos es alta. El método correlacionado, proporciona mejor protección en un sentido global. En muchos casos, el método de adición de ruido puede utilizarse eficazmente para preservar la confidencialidad. Sin embargo, cada caso debe considerarse por separado para determinar

la cantidad de ruido que debe añadirse y cómo influye este ruido en la integridad de los datos.

Se puede concluir con que, esta técnica perturbativa es adecuada para proteger, principalmente, datos continuos, ya que no se pueden hacer suposiciones correctas sobre los posibles valores, que pueden tomar, los atributos sensibles y porque no se puede hallar una correspondencia exacta con otras fuentes de información externas, que permitan una reidentificación.

### 2.5.2.3. Rank swapping

El rank swapping [38] es una variación de otro popular método de enmascaramiento perturbativo denominado *data swapping*, que consiste en modificar una serie de tuplas de un conjunto de datos, intercambiando los valores de una serie de atributos entre pares de tuplas seleccionados, según un criterio bien definido [6]. El *data swapping* es un método simple, capaz de dificultar en gran medida una posible reidentificación, aunque para colecciones enormes de datos, el coste computacional puede llegar a ser demasiado alto.

El rank swapping, es una variación de lo anterior, que puede aplicarse tanto a atributos continuos como a atributos categóricos que tengan una relación de orden. La forma de proceder es la siguiente, los valores de un atributo a enmascarar se ordenan en orden ascendente, y cada valor se intercambia con otro valor, garantizando siempre que las tuplas intercambiadas se encuentran a una distancia de rango determinada,  $p$ . Esa distancia determinada puede determinarse de diversas formas, por ejemplo, un valor de entrada que represente la máxima distancia de rango entre dos tuplas a intercambiar. A valores mayores de dicha distancia de rango, obtenemos mayores posibles permutaciones en los datos mientras que cuanto más pequeño sea dicho valor, aumenta el riesgo de reidentificación [25]. El procedimiento de intercambio puede realizarse sucesivas veces para diversos atributos sobre un mismo conjunto de datos, que ya podía haber sido modificado previamente, esto pese a aumentar la anonimización, puede eliminar la utilidad de los datos. El rank swapping se definió, en un inicio, para datos categóricos ordinales y posteriormente se empezó a emplear con datos continuos.

En el siguiente ejemplo, la Figura 2.12 representa una base de datos sintética con información sobre unos pacientes ingresados en un centro de salud. Queremos aplicar rank swapping para el atributo *Temperatura* con un  $p = 20\%$ , el rango de valores que puede tomar este atributo es [36,5-39,2], por tanto, para poder intercambiar dicho valor entre dos tuplas, la diferencia debe ser menor o igual a  $((39,2 - 36,5) \cdot 20)/100 = 0,54$ . Por esa regla, podríamos intercambiar los valores de las tuplas 1/6, 2/5 y 3/4, como podemos ver en la Figura, ya modificada, 2.13 .

Nacionalidad	Edad	Enfermedad	Temperatura
Portugal	65	Neumonía	36.9
España	55	Covid-19	37.9
Marruecos	72	Miocarditis	38.8
España	82	Peritonitis	39.2
Francia	61	Covid-19	38.2
España	89	Pulmonía	37.2

Figura 2.12: Base de datos sintética sobre los pacientes de un centro de salud.

Nacionalidad	Edad	Enfermedad	Temperatura
Portugal	65	Neumonía	37.2
España	55	Covid-19	38.2
Marruecos	72	Miocarditis	39.2
España	82	Peritonitis	38.8
Francia	61	Covid-19	37.9
España	89	Pulmonía	36.9

Figura 2.13: Base de datos sintética sobre los pacientes de un centro de salud tras aplicar rank swapping al atributo *Temperatura*.

### **3. Data sketches**

---

En la actualidad, con la explosión del Big Data, resulta crucial para las organizaciones de todos los sectores poder almacenar, procesar y analizar grandes cantidades de información. Los dispositivos conectados en aumento, la digitalización de procesos y la mayor interconexión de sistemas causan una gran cantidad de datos que supera las capacidades de las infraestructuras tradicionales.

La gestión de grandes volúmenes de datos implica no solo la necesidad de manejarlos, sino también la extracción de valor significativo en un tiempo razonable como la posibilidad de realizar consultas eficientes sobre dichos datos. No obstante, en esta situación las estructuras de datos convencionales pueden tener limitaciones significativas. Cuando se enfrentan a conjuntos de datos a gran escala, las bases de datos relacionales pueden experimentar *cuellos de botella* en el rendimiento. Las estructuras de datos estándar utilizadas en algoritmos y aplicaciones tradicionales pueden volverse ineficientes o incluso inviables al enfrentarse a la gran cantidad de datos generada.

Imagina que cuentas con una red social muy destacada con cientos de millones de usuarios activos. Este sitio guarda grandes cantidades de información, como perfiles de usuarios, publicaciones, interacciones y comentarios. Ahora, imaginemos que deseas añadir una característica que permita a los usuarios buscar publicaciones relevantes basadas en ciertos criterios como palabras clave, ubicación geográfica y fecha. Para lograr esto utilizando métodos de almacenamiento convencionales, podrías pensar en utilizar una base de datos relacional. No obstante, cuando la plataforma crece considerablemente, recuperar publicaciones relevantes mediante consultas a esta base de datos relacional se vuelve muy costoso en términos de recursos y extremadamente lento. Indexar cada publicación para permitir búsquedas eficientes puede resultar poco práctico debido al gran volumen y a la variabilidad de los datos. Las consultas complejas que involucran múltiples tablas y filtros, además de resultar en tiempos de respuesta inaceptables, pueden causar una experiencia deficiente para el usuario y pérdida de satisfacción del cliente. En definitiva, las estructuras estándar utilizadas en algoritmos y aplicaciones tradicionales pueden volverse ineficientes.

#### **3.1. Introducción a las estructuras de datos probabilísticas**

Debido a lo comentado anteriormente, surge la necesidad de explorar y adoptar nuevas técnicas y estructuras de datos que puedan manejar eficazmente el tamaño y la complejidad de los volúmenes de datos que se manejan en la actualidad. En este punto, debemos preguntarnos si realmente necesitamos un alto nivel de precisión en nuestras consultas, esto dependerá del caso concreto que se esté abordando.

Si una respuesta aproximada es aceptable, entonces es posible que existan algoritmos que permitan responder estas consultas en órdenes de magnitud más rápidos.

Asumiendo que la capacidad de respuesta y la velocidad de las consultas son primordiales. Las estructuras de datos probabilísticas son herramientas innovadoras que ofrecen soluciones para problemas de almacenamiento, consulta y análisis en entornos de Big Data.

Las estructuras de datos probabilísticas se caracterizan por su habilidad para proporcionar respuestas aproximadas con un alto nivel de precisión, lo que las hace idóneas especialmente para aplicaciones donde la velocidad y escalabilidad son fundamentales. Aprovechan principios probabilísticos para proporcionar estimaciones y aproximaciones de consultas en tiempos razonables, independientemente del tamaño del conjunto de datos. Asimismo, su diseño eficiente permite el ahorro de recursos computacionales y de almacenamiento, convirtiéndolas en una opción atractiva en entornos donde la eficiencia es fundamental.

En la mayoría de los casos, estas estructuras de datos utilizan funciones hash para aleatorizar los elementos. Como ignoran las posibles colisiones, mantienen el tamaño constante, pero esta es también la razón por la que no pueden dar valores exactos. Entre las ventajas que aportan destacan el hecho de que utilizan poca memoria (pudiéndose controlar la misma), su facilidad para paralelizarse y la capacidad de realizar consultas sobre ellas en tiempo constante.

Existen diversas estructuras o técnicas para realizar consultas aproximadas. A lo largo del capítulo se profundizará en los Data Sketches. Se expondrá este tipo de estructura a nivel conceptual, profundizando en implementaciones concretas destinadas a la estimación de frecuencias o cardinalidad sobre conjuntos de datos masivos.

### 3.2. Familia de hashes independientes entre sí

Las tablas y funciones hash es un campo muy amplio dentro del mundo de la informática y las matemáticas. Han demostrado ser herramientas insustituibles en los sistemas modernos, siendo más difícil encontrar un sistema que no las utilice que uno que sí lo haga. Recientemente ha habido mucho trabajo de innovación que aborda problemas algorítmicos que surgen a medida que las tablas hash crecen para adaptarse a datos masivos, como el redimensionamiento eficiente, la representación compacta y trucos para ahorrar espacio. En una línea similar, el hashing se ha adaptado para soluciones distribuidas, donde la tabla hash se divide entre servidores, con los desafíos que esto ocasiona [35]. A lo largo del capítulo, se presentan una serie de estructuras de datos probabilísticas donde las funciones hash tienen un papel fundamental.

Normalmente se define una función hash como una función matemática que toma una entrada de cualquier tamaño y produce una salida de longitud fija, generalmente una cadena de bits de tamaño predefinido. Esta salida única, conocida como hash, se calcula de manera determinista, lo que significa que una misma entrada siempre producirá la misma salida. Las funciones hash deben ser rápidas de calcular y diseñadas de manera que pequeños cambios en la entrada produzcan grandes cambios en el hash resultante. Otra característica de una función hash de calidad es el hecho de ser resistente a las colisiones, es decir, que sea improbable que dos entradas diferentes produzcan el mismo hash.

A partir del concepto de función hash, surge la estructuras de datos denominadas tabla hash, también conocida como mapa hash o diccionario. Esta estructura utiliza funciones hash para mapear claves a valores. Puede verse como un vector de contenedores y una función hash que asigna cada clave a uno de estos contenedores. Esto permite un acceso rápido a los valores asociados con cada clave, con un tiempo de búsqueda promedio constante,  $O(1)$ , en lugar de tiempos del orden lineal respecto al tamaño del conjunto de datos, como en las estructuras de datos lineales. En el peor de los casos, pueden tener tiempos de búsqueda de orden lineal,  $O(n)$ , pero con un buen diseño de la tabla hash, casi siempre se puede evitar tales situaciones.

Un concepto importante, que se utilizará a lo largo del capítulo, es el de funciones hash independientes entre pares y entre cuatro elementos. Una familia de funciones hash independientes entre pares se define mediante las funciones de la forma  $h(x) = ax + b \bmod p$ , los valores de  $a$  y  $b$  son elegidos uniformemente entre 0 y  $p - 1$ , donde  $p$  es un número primo [12]. Sobre la elección aleatoria de  $a$  y  $b$ , la probabilidad de que dos elementos colisionen bajo la función hash es  $\frac{1}{p}$ .

De manera similar, una familia de funciones hash independientes entre cuatro elementos significa que si aplicamos una función hash sobre cuatro valores diferentes, los valores resultantes serán lo más independientes posible entre sí. Esto se consigue con funciones hash de la forma  $g_j(x) = 2((ax^3 + bx^2 + cx + d \bmod p) \bmod 2) - 1$ , con los parámetros  $a, b, c$  y  $d$  seleccionados uniformemente a partir del valor primo,  $p$  [8]. De igual manera, este concepto puede extenderse a familias de funciones hash  $k$ -independientes, incrementando el número de parámetros. Todas estas funciones hash pueden calcularse muy rápidamente, incluso más rápido que funciones hash más populares, usadas en criptografía, como MD5 o SHA-1.

### 3.3. Definición y fundamentos de los data sketches

Para mejorar la experiencia del usuario y optimizar las operaciones comerciales, las empresas necesitan realizar análisis en tiempo real de los datos que recopilan continuamente. Sin embargo, debido a la enorme escala de los mismos y la necesidad de respuestas rápidas, es inviable almacenar y procesar toda la información de manera tradicional. Aquí es donde entra en juego el uso de un data sketch o sketch. En lugar de almacenar la totalidad de la información de forma detallada, la empresa puede utilizar una estructura de datos eficiente que resuma la información clave de manera compacta. Por ejemplo, podría utilizar un data sketch para calcular la cantidad total de productos vendidos por categoría, la frecuencia de búsqueda de ciertos términos o la distribución de precios de los productos más populares.

Un sketch es simplemente una estructura de datos para representar un resumen de un flujo de entrada de datos. Estas estructuras son representaciones de los datos originales que se obtienen al aplicar una operación matemática lineal a los datos de entrada. Supongamos que tenemos una serie de datos, como por ejemplo, un conjunto de números que representan la frecuencia de ocurrencia de ciertos elementos, como en un histograma. Estos datos pueden ser organizados en forma de vector o matriz, donde cada elemento del vector representa una característica o una categoría específica de los

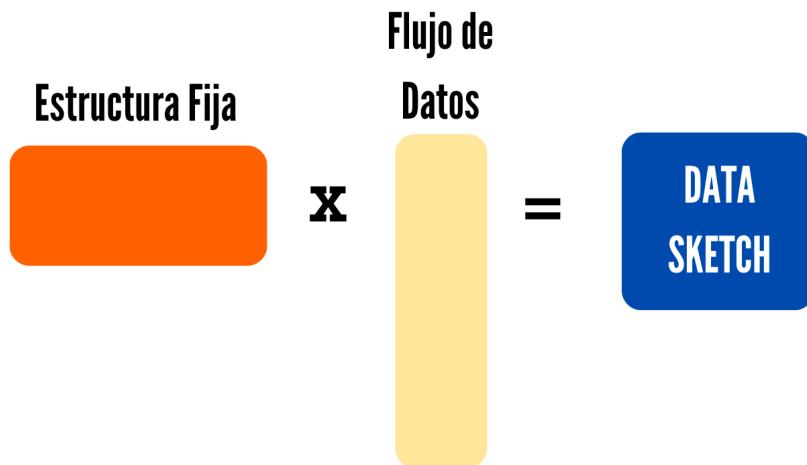


Figura 3.1: Esquema generación de un data sketch

datos.

Para obtener el data sketch de estos datos, se utiliza una estructura fija (es decir, un vector o matriz predefinida que no cambia), y se realiza una multiplicación entre esta matriz y los datos originales. La multiplicación de la matriz fija con los datos originales produce un nuevo vector que representa el data sketch de los datos. Este nuevo vector resumirá de alguna manera la información contenida en los datos originales, pero de una manera más compacta y simplificada, ver Figura 3.1. La estructura fija puede ser un conjunto de funciones hash predefinidas que se definen antes de comenzar el proceso, y su estructura y comportamiento no cambian durante la ejecución del algoritmo.

Una propiedad interesante de los data sketch es que cualquier actualización tiene el mismo impacto independientemente de las actualizaciones anteriores. Por otro lado, suelen ser estructuras de un tamaño asintóticamente más pequeño que la entrada, como máximo polilogarítmico en función de la entrada. Debido a esta compresión de la información, casi ninguna función o consulta sobre los datos puede realizarse con precisión. Responder consultas de un tipo dado debe ser rápido y tener garantías de precisión asociadas. Típicamente, las garantías de precisión se exponen en términos de los parámetros especificados por el usuario,  $\varepsilon$  y  $\delta$ , lo que significa que el error al responder a una consulta está dentro de un margen de  $\varepsilon$  con una probabilidad  $\delta$ .

En secciones siguientes se expondrán diversas técnicas de sketching propuestas en la literatura, diferenciando en dos bloques según el tipo de estimación, de frecuencia o cardinalidad, que se tenga por objetivo realizar sobre los datos existentes.

### 3.4. Propiedades de los data sketches

Tal y como se ha comentado anteriormente, los diferentes tipos de data sketch comparten una serie de características. A continuación, se exponen las principales propiedades de estas estructuras de datos [12].

- **Consultas admitidas.** Cada data sketch está definido para admitir un cierto

conjunto de consultas. Algunos están orientados a realizar estimaciones de frecuencia sobre un elemento concreto, estimaciones de cardinalidad sobre un flujo de datos u otro tipo de cálculos estadísticos. El procedimiento a seguir en cada estimación difiere de un tipo a otro.

- **Espacio requerido.** Los data sketch tienen una serie de parámetros que determinan el tamaño del mismo. Un caso común es cuando los parámetros  $\varepsilon$  y  $\delta$  son elegidos por el usuario para determinar la precisión (error de aproximación) y la probabilidad de exceder los límites de precisión, respectivamente. En los casos concretos, se detallará como la anchura y altura de la matriz de sketch se obtienen a partir de dichos parámetros.
- **Tiempo de actualización.** Puesto que este tipo de estructuras trabajan sobre flujos de datos en tiempo real, deben proporcionar tiempos relativamente cortos en sus actualizaciones. Por lo general, dependerá del data sketch y del número de funciones hash utilizadas. Dado que las funciones hash son económicas de producir y calcular, el acceso a la estructura de datos, en modo escritura, se puede considerar de tiempo constante.
- **Tiempo de consulta.** El tiempo necesario para responder a una consulta utilizando un data sketch puede variar según el algoritmo concreto y la naturaleza de la consulta. En algunos casos, el tiempo de consulta puede ser lineal respecto al tamaño de la estructura, mientras que en otros casos puede ser mucho menor, llegando a  $O(1)$ .
- **Estado inicial.** La inicialización de un data sketch es típicamente sencilla, ya que generalmente se establece en un estado inicial nulo. No obstante, cuando se utilizan funciones hash, puede ser necesario inicializar estas funciones de manera adecuada antes de su uso.

### 3.5. Notación y terminología

A lo largo de toda la sección se presentan una serie de estructuras de datos probabilísticas que comparten entre sí una terminología y notación específica. En primer lugar todas estas estructuras se plantean para trabajar sobre flujos de actualizaciones de datos. Supongamos que contamos con un vector,  $a$ . Este vector podría representar nuestro conjunto de datos de interés y está formado por elementos pertenecientes al dominio,  $U$ . Adicionalmente, se define  $M = |U|$ . El vector,  $a$ , tiene dimensión,  $n \leq M$ , y el valor de los elementos del vector,  $a(t) = [a_1(t), a_2(t), a_3(t), \dots, a_n(t)]$ , dependerá del instante de tiempo  $t$ . Inicialmente, todos los elementos de  $a$  toman el valor 0. El valor de  $t$  se suele omitir, puesto que, se trabaja sobre el vector más actualizado.

Sobre este vector,  $a$ , se realizan actualizaciones, proceso de agregar nuevos elementos o actualizaciones a un data sketch existente, de la forma  $\langle i, c \rangle$ , donde  $i$  representa el elemento del vector cuyo valor se desea modificar y  $c$ , el incremento o decremento a realizar sobre el valor actual,  $a_i(t)$ . También puede darse actualizaciones de la forma  $\langle i \rangle$ , especialmente en aquellos casos sobre los que se busca realizar estimaciones de cardinalidad, más adelante se presentan.

En cualquier momento,  $t$ , se puede requerir realizar una consulta sobre dicho vector. Existen diferentes tipos de consultas pero centraremos nuestro estudio en las denominadas *consultas de punto*. Una consulta de punto, denotada como  $Q(i)$ , consiste en devolver una aproximación del valor  $a_i$ . También se abordarán consultas sobre la cardinalidad o alguna métrica de carácter estadístico sobre el vector  $a$ .

Otro término importante es el de *error relativo*. Este se refiere a la discrepancia entre la estimación proporcionada por el data sketch y el valor real del conjunto de datos. Dado que los data sketch son estructuras de datos probabilísticas diseñadas para ofrecer estimaciones rápidas y aproximadas en conjuntos de datos grandes, es importante comprender y controlar el error asociado con estas estimaciones. Este error dependerá en gran medida del tamaño de la estructura, prefijado por el usuario, y las familias de funciones hash escogidas.

## 3.6. Tipos de data sketches

Una clasificación comúnmente utilizada para diferenciar los data sketches se basa en su funcionalidad específica. Algunos data sketches, como los filtros de Bloom, se emplean para estimar la existencia de un elemento en un conjunto, permitiendo determinar si un elemento probablemente pertenece al conjunto o no pertenece a él. Por otro lado, existen data sketches diseñados para la estimación de frecuencias, que permiten aproximar cuántas veces aparece un elemento dentro de un flujo de datos. Por último dentro de la clasificación propuesta en el documento, existen otros data sketches que se enfocan en la estimación de cardinalidad, proporcionando una aproximación del número total de elementos distintos en un flujo de datos. A continuación, se profundizará en los diferentes tipos concretos de data sketches.

### 3.6.1. Filtros de Bloom

Los filtros de Bloom se han convertido en una técnica muy popular en sistemas que procesan grandes volúmenes de datos. Fueron propuestos en 1970 [5], aunque, en los últimos años, ha aumentado su uso notablemente, debido a una creciente necesidad de manejar y comprimir conjuntos de datos grandes. Además, se han convertido en un área clave de la investigación informática, desarrollándose muchas variantes sobre esta estructura de datos para mejorar algunas deficiencias de la misma [35].

Los filtros de Bloom guardan una estrecha relación con las tablas hash. Ya que admiten la inserción y búsqueda de la misma manera que lo hacen las tablas hash, pero utilizando muy poco espacio. Esto se debe a que los filtros de Bloom no almacenan los elementos en sí mismos, sino que determinan la existencia o no de un elemento en un conjunto. Formalmente, se define un filtro de Bloom como una cadena binaria,  $B$ , de longitud  $m \leq M$ , siendo,  $M$ , el cardinal de nuestro dominio de estudio. Inicialmente,  $B$ , se encuentra con todos los bits a 0. Adicionalmente, se requiere una familia de,  $k$ , funciones hash  $k$ -independientes,  $h_1, h_2, \dots, h_k$ , que mapean los elementos, de manera uniforme, al intervalo  $[0, m - 1]$ .

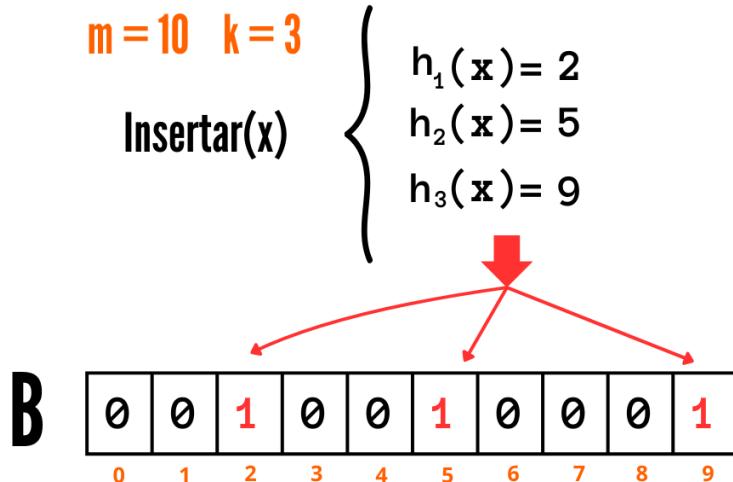


Figura 3.2: Inserción en un filtro de Bloom

La inserción en un filtro de Bloom es muy simple. Para insertar un elemento,  $x$ , basta con calcular las,  $k$ , funciones hash y para cada una de las salidas obtenidas, establecer el bit en dicha posición a 1. Es decir,  $B[h_j(x)] = 1, \forall j, [1, k]$ . La operación de inserción tiene un coste temporal del orden,  $O(k)$ , en función del número de funciones hash utilizadas. Se puede considerar de tiempo constante para aquellos casos en los que,  $k \ll M$ . La Figura 3.2, muestra de forma gráfica la inserción de un elemento en un filtro de Bloom.

Para comprobar la existencia de un elemento,  $y$ , en el filtro de Bloom bastaría con calcular las,  $k$ , funciones hash para dicho elemento, al igual que en la inserción. Posteriormente, se comprueba para cada una de las salidas obtenidas el estado del bit en la posición  $h_j(x)$ ,  $\forall j, [1, k]$ . Si alguno de los bits se encuentra establecido a 0, se determina que el elemento,  $y$ , no se encuentra en el conjunto. Por el contrario, si todos los bits toman el valor 1, se determina la existencia de dicho elemento. La Figura 3.3, muestra el proceso de búsqueda para un caso positivo y para uno negativo.

Una característica clave de esta estructura es que pese a que pueden producirse falsos positivos, jamás se producirá un falso negativo. En situaciones donde se espera que la respuesta de la consulta sea que el elemento no está presente la mayor parte del tiempo, los filtros de Bloom ofrecen una gran precisión además de beneficios de ahorro de espacio. La búsqueda también pertenece al orden,  $O(k)$ , en caso de que la operación tenga que revisar todos los bits, pero la mayoría de las búsquedas no exitosas finalizarán antes.

La idea original de los filtros de Bloom no cumple los requisitos para ser considerado un data sketch, la estructura de datos no es una transformación lineal de la entrada ya que establecer un bit a 1 no es una operación lineal [12]. Los filtros de Bloom se pueden modificar para convertirse en un data sketch, con la desventaja de aumentar el espacio requerido. En lugar de una cadena de bits, el filtro de Bloom se representa por un conjunto de contadores. Al agregar un elemento, aumentamos los contadores correspondientes en 1, es decir,  $B[h_j(x)] = B[h_j(x)] + 1$ . Esto nos permite realizar otro tipo de consultas, como la frecuencia de un elemento en un flujo de datos,

$$m = 10 \quad k = 3$$

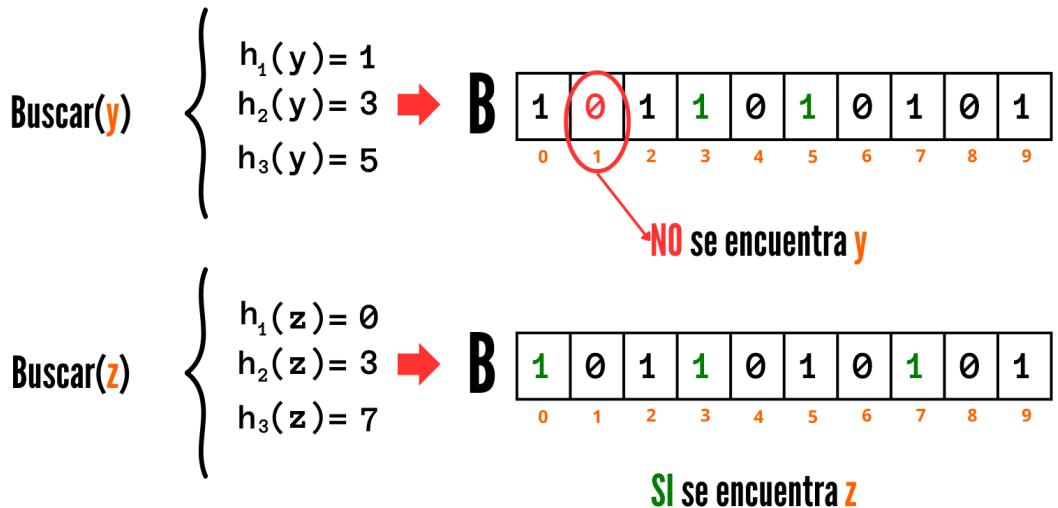


Figura 3.3: Búsqueda en un filtro de Bloom

obteniéndose como el valor mínimo de los contadores determinados por las salidas de las funciones hash. El número de entradas necesarias sigue siendo el mismo, pero ahora las entradas son contadores, números enteros, en lugar de bits, por lo que se requiere mayor espacio.

Independientemente de que se trate el filtro de Bloom como un data sketch o no, existen una serie de parámetros claves en la construcción y análisis de la estructura, alguno de ellos pueden definirse en base a otros. En primer lugar,  $n$ , se define como el número de elementos posibles a insertar,  $n \leq M$ , dependiendo del caso concreto.  $f$ , se define como la tasa de falsos positivos,  $m$ , es el número de bits o contadores necesarios y, por último,  $k$ , es el número de funciones hash. La tasa de falsos positivos se puede obtener a partir de la siguiente formula:

$$f \approx (1 - e^{-\frac{nk}{m}})^k = e^{k \ln(1 - e^{-\frac{kn}{m}})} \quad (3.1)$$

A partir de unos límites en  $n$  y  $m$ , se pueden establecer valores óptimos de  $k$ . Se asume que las funciones hash son totalmente aleatorias. Es decir, la ubicación a la que un elemento es mapeado por cualquier función hash se considera uniformemente aleatoria dentro del rango de posibilidades, y completamente independiente de las otras funciones hash [12]. Un aspecto clave de cara a la eficiencia del filtro de Bloom, es la relación bits o contadores por elemento,  $m/n$ . A medida que la proporción aumenta, la tasa de falsos positivos disminuye, para el mismo número de funciones hash. Si embargo, un valor alto de  $m/n$ , puede volverse impracticable para aquellos casos donde,  $n$ , sea muy grande.

En [12, 35], se muestra el calculo seguido para obtener un valor de,  $k$ , óptimo a partir de un ratio, bits por elemento establecido,  $k_{opt} = \frac{m}{n} \ln 2$ . Seleccionar el valor de,  $k$ , a partir de dicha formula garantiza la tasa de falsos positivos mínima posible, para

los valores establecidos de  $m$  y  $n$ , de la forma:

$$f_{opt} = \left(\frac{1}{2}\right)^{k_{opt}} \quad (3.2)$$

Generalmente, los filtros de Bloom se construyen a partir de un valor de  $n$ , dependiente del problema en cuestión, y de una tasa de falsos positivos,  $f$ , deseada. Lo ideal es contar con una tasa de falsos positivos de entorno al 2 % – 5 %.

### 3.6.2. Estimadores de frecuencia

En el contexto actual, es innegable el valor de los datos. Las empresas recolectan una gran cantidad de información de diferentes fuentes, que incluyen desde transacciones comerciales hasta interacciones o emoticonos en redes sociales. No obstante, tener simplemente estos datos no asegura el éxito. El objetivo está en poder extraer conocimientos significativos y aplicables a partir de ellos. En este sentido, el análisis de popularidad o frecuencia se convierte en una herramienta necesaria.

El objetivo de un análisis de popularidad es comprender cómo se distribuyen y ocurren los eventos dentro de un conjunto de datos. Este enfoque brinda una visión detallada de la actividad empresarial, desde el recuento de ventas de productos hasta la monitorización de patrones de tráfico web. Las organizaciones pueden entender la demanda del mercado, identificar tendencias u optimizar la oferta de productos, lo que les permite tomar decisiones estratégicas y mejorar la experiencia del cliente.

Un ejemplo concreto de este tipo de análisis es implementado por Apple [15]. El navegador Safari en macOS High Sierra proporciona a los usuarios la capacidad de navegar por la web con menos distracciones, normalmente en forma de alivio de los sitios web que reproducen automáticamente sonidos. Safari determina que un usuario puede querer la reproducción automática con sonido de un sitio web, si el usuario reproduce un elemento multimedia que originalmente se bloqueó, o cuando permite que un elemento multimedia habilitado continúe reproduciéndose. Gracias a la participación de los usuarios y la recolección de sus datos, Apple es capaz de identificar con éxito un conjunto actualizado de sitios web para los cuales la mayoría de los usuarios prefieren la reproducción automática con sonido en todo el mundo.

Todas las situaciones comentadas anteriormente se traducen en medir la frecuencia, para cada elemento único (en el ejemplo anterior, un sitio web), se debe mantener una estructura de datos que recoja el número de apariciones del mismo. Los enfoques tradicionales, como un diccionario, ocupan un espacio proporcional al número de elementos diferentes que nos interesa medir su frecuencia,  $O(N)$ . Estas soluciones son válidas cuando el conjunto de elementos es relativamente pequeño. Los casos que se presentan en el mundo empresarial actual llevan asociados conjuntos de datos masivos. Imagina que la empresa Amazon, desea conocer sus productos más vendidos. Con la gran cantidad de productos diferentes existentes (la gran mayoría cuentan con una frecuencia de compra baja), las estructuras de datos tradicionales se vuelven inservibles. Por lo tanto, necesitamos recurrir a soluciones aproximadas.

### 3.6.2.1. Count-Min Sketch

Un count-min sketch es una estructura de datos probabilística compacta capaz de representar un vector de alta dimensión y responder consultas con alto grado de precisión sobre este vector, en particular, consultas de puntos, definidas previamente. Esta estructura se representa mediante una matriz de números enteros, de anchura  $w$  y profundidad  $d$ , la cual funciona a modo de contador.

$$CMS[1, 1], \dots, CMS[d, w] \quad (3.3)$$

Para definir esta estructura se requieren dos parámetros, introducidos en la sección 3.3, el margen de sobreestimación,  $\varepsilon$  y la probabilidad de fallo,  $\delta$ . A partir de estos parámetros, fijados por el usuario, se establecen los siguientes,  $d = \lceil \ln \frac{1}{\delta} \rceil$  y  $w = \lceil \frac{\varepsilon}{\delta} \rceil$ . Además, esta estructura de datos requiere establecer  $d$  funciones hash de una misma familia,  $h_1, \dots, h_d : \{1, \dots, n\} \rightarrow \{1, \dots, w\}$ , seleccionadas de manera aleatoria.

#### Fase de actualización

Cuando llega una actualización de la forma  $\langle x, c \rangle$ , debemos actualizar el contador para el elemento,  $x$ , en la cantidad,  $c$ . Como nuestra matriz no almacena información real sobre el número de apariciones del elemento,  $x$ , se realiza el siguiente proceso. Para cada uno de los  $d$  hashes, se calcula el valor,  $h_d(x)$ . Una vez calculado, se actualiza el valor de la celda,  $CMS[d, h_d(x)]$ , de la forma:

$$CMS[d, h_d(x)] = CMS[d, h_d(x)] + c \quad (3.4)$$

La Figura 3.4, muestra un ejemplo visual de la actualización de un count-min sketch. Se desea incrementar el contador para el elemento,  $x$ , en una unidad. En dicho ejemplo, el rango de las distintas funciones hash, se representa con letras griegas en lugar de números enteros.

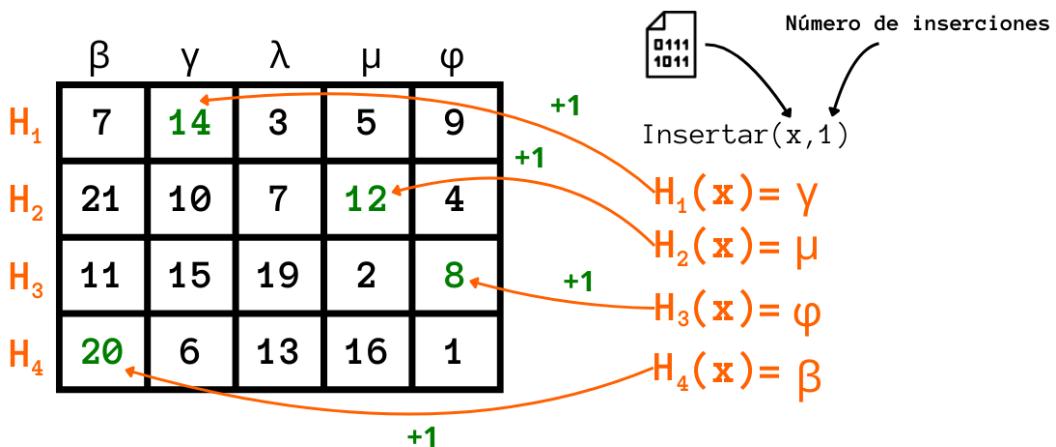


Figura 3.4: Actualización de un count-min sketch

Se puede intuir que una correcta elección de los parámetros, aumentará la precisión de las consultas. Aumentar el rango de las funciones hash,  $w$ , aumenta la precisión

de las consultas, y aumentar el número de funciones hash,  $d$ , disminuye la probabilidad de una estimación errónea, esto se detallará a continuación.

### Fase de estimación

Las consultas de punto, definidas anteriormente, no son las únicas consultas que se pueden realizar sobre este tipo de data sketches. Existen otras como la consultas de rango, para determinar la frecuencia de un rango de elementos, o las consultas de producto interno. Enfocamos nuestro análisis en las consultas de punto debido a su sencillez y puesto que son la base de muchas de las aplicaciones comentadas en anteriores secciones.

Supongamos que queremos conocer el contador o la frecuencia asociada al elemento,  $x$ . La respuesta a la consulta  $Q(x)$ , viene determinada por el valor aproximado,  $\tilde{a}$ .

$$\tilde{a} = \min_j CMS[j, h_j(x)] \quad (3.5)$$

Es decir, se calculan los valores de las  $d$  funciones hash para la entrada,  $x$ , y se devuelve el mínimo entre  $d$  celdas de la matriz de sketch en  $d$  filas diferentes, donde la celda seleccionada en la  $j$ -ésima fila viene especificada por la función hash  $h_j(x)$ , con  $1 \leq j \leq d$ . La Figura 3.5 ejemplifica la fase de estimación de la frecuencia para un elemento,  $x$ .

	$\beta$	$\gamma$	$\lambda$	$\mu$	$\varphi$
$H_1$	7	14	3	5	9
$H_2$	21	10	7	12	4
$H_3$	11	15	19	2	8
$H_4$	20	6	13	16	1

$\text{Estimar}(x) = \text{MÍNIMO}(14, 12, 8, 20) = 8$

Figura 3.5: Estimación en un count-min sketch

En lugar de seleccionar el valor mínimo, se puede optar por realizar otra operación sobre los valores obtenidos por cada hash, como por ejemplo, la operación de la media (**count-mean sketch**) o la mediana (**count-median sketch**). El funcionamiento de estas estructuras a la hora de actualizar o las garantías de precisión que proporcionan son similares.

Inicialmente, se definieron los parámetros  $\varepsilon$  y  $\delta$ , los cuales actúan como medida de error para esta estructura de datos. Según el Teorema número 1 en [9], se puede afirmar que el valor estimado,  $\tilde{a}$ , cuenta con las dos siguientes garantías:

$$\tilde{a} \geq a \quad (3.6)$$

$$\tilde{a} \leq a + \varepsilon \|a\|_1, \text{ con probabilidad, } p \geq 1 - \delta \quad (3.7)$$

La ecuación 3.6 garantiza que la estimación obtenida siempre será mayor o igual que el valor real. Esta sobreestimación podrá ocurrir cuando se hayan producido colisiones de hashes en las sucesivas actualizaciones, nunca se producirá una subestimación. La ecuación 3.7 expresa el valor que puede tomar dicho margen de sobreestimación. El valor de  $\|a\|_1$  representa la suma total de frecuencias observadas hasta el momento en el flujo de actualizaciones. A medida que el flujo de actualizaciones avanza, se incrementa  $\|a\|_1$ , por lo que el error de sobreestimación aumenta. Este error penalizará, sobre todo, a los elementos con bajas frecuencias ya que el error máximo de sobreestimación solo es sensible a la suma total de frecuencias, no a la frecuencia individual de los elementos.

Generalmente, se establecen valores muy pequeños de  $\delta$ , cercanos a 0,01, para poder garantizar, con una alta probabilidad, que el error de sobreestimación se mantenga dentro del margen establecido.

Es posible ajustar la precisión de nuestro count-min sketch variando los valores de  $\varepsilon$  y  $\delta$ , siendo conscientes que al aumentar la precisión, aumentará el espacio requerido. Esto último es debido a que la anchura y profundidad de la matriz se obtiene a partir de  $\varepsilon$  y  $\delta$ , como se pudo ver anteriormente. Por lo tanto, el espacio necesario para almacenar un count-min sketch pertenece al orden,  $O\left(\frac{e \ln\left(\frac{1}{\delta}\right)}{\varepsilon}\right)$ .

### 3.6.2.2. AMS Sketch

La estructura de datos probabilística AMS sketch, fue propuesta originalmente por Alon, Matias y Szegedy, de ahí su nombre, en 1996 [2]. Diferentes versiones de esta estructura han ido apareciendo con el paso del tiempo. La que se tratará en esta sección, utiliza funciones hash para mapear las sucesivas actualizaciones de los elementos, a diferencia de la propuesta original.

El propósito de este data sketch es el de calcular estadísticas agregadas sobre un flujo de actualizaciones de datos, en concreto, la norma  $L_2$ . Este tipo de estadísticas, proporcionan información sobre las características generales del conjunto de datos en su totalidad. Estas estadísticas ofrecen una visión global del comportamiento o la distribución de los datos, en lugar de centrarse en elementos individuales. El cálculo de la norma  $L_2$  sobre un vector que almacena las frecuencias de una serie de elementos nos dará una medida de la magnitud o tamaño del conjunto de datos en términos de sus frecuencias. En el mundo de la analítica de datos, optimización y aprendizaje automático se considera una medida fundamental que proporciona información sobre la estructura que subyace.

Aunque el propósito del AMS sketch sea aproximar el valor comentado anteriormente. Esta estructura nos permitirá realizar consultas de punto aproximadas, para obtener una estimación sobre la frecuencia de un elemento, tal y como se mostrará en la fase de estimación.

Al igual que en los data sketch comentados hasta el momento, esta estructura de datos mantiene una matriz compacta de contadores, de anchura  $w$ , y profundidad,  $d$ . Los valores de los parámetros  $\varepsilon$  y  $\delta$  establecen el espacio requerido por la estructura.

Además se garantiza con una probabilidad de  $1 - \delta$  que la estimación se encuentra dentro de un  $\varepsilon$ -error relativo del valor real.

Esta estructura necesita dos familias de hashes diferentes [10]. La primera de ella está formada por  $d$  funciones hash independientes entre sí. Estas funciones mapean los elementos de entrada,  $x_i$ , a valores en el rango  $\{1, 2, \dots, w\}$ . Para la segunda familia de  $d$  funciones hash, estas deben ser independientes cuatro a cuatro y mapear los elementos de entrada a valores  $\{-1, +1\}$ . Dichos conceptos se exponen en la sección 3.2.

### Fase de actualización

Al inicio, la matriz AMS que almacena contadores se encuentra inicializada a 0. Cuando se produce una actualización de la forma  $\langle x, c \rangle$ , donde  $x$  es el elemento cuya frecuencia deseamos modificar y  $c$ , dicha modificación o peso sobre el contador asociado. Este valor,  $c$ , puede tomar valores tanto positivos como negativos dependiendo del caso concreto y la finalidad de nuestros datos.

Se calculan las  $d$  funciones hash de la primera familia de hashes para la entrada  $x$ , de esta forma, obtenemos el índice de la celda o columna de cada fila de la matriz asociado al elemento  $x$ . Sobre cada una de las celdas obtenidas, se actualiza el contador, sumando el valor  $c$  multiplicado por el resultado de haber aplicado la función hash de la segunda familia sobre el elemento  $x$ .

Formalmente, para cada una de las  $d$  filas de la matriz con valores entre  $1 \leq j \leq d$ , se calcula  $h_j(x)$  y el resultado  $cg_j(x)$  se añade de la forma:

$$AMS[j, h_j(x)] = AMS[j, h_j(x)] + cg_j(x) \quad (3.8)$$

Procesar cada una de las actualizaciones conlleva un coste temporal lineal,  $O(d)$ , considerando el cálculo de las funciones hash de tiempo constante. La Figura 3.6, muestra gráficamente el procedimiento necesario para llevar a cabo una actualización.

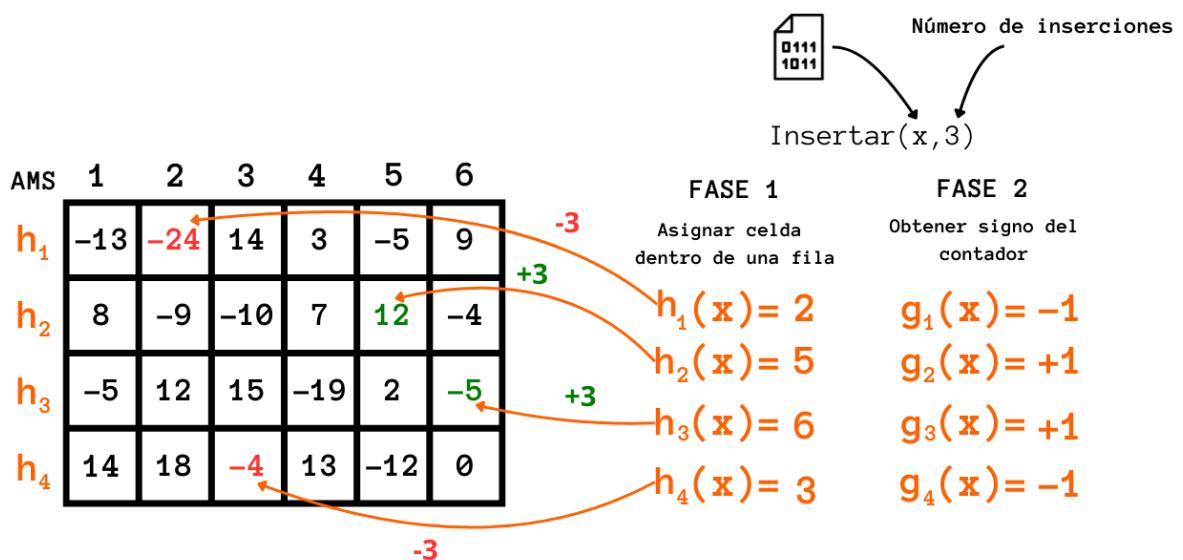


Figura 3.6: Fase de actualización en un AMS sketch

## Fase de estimación

Tal y como se comentó al inicio de la sección, este data sketch permite realizar una estimación de la norma  $L_2$ , en concreto  $L_2^2$ , del vector que almacena las frecuencias de una serie de elementos. Para obtener esta estimación [8], se calcula la suma de los elementos de cada fila,  $j$ , al cuadrado,  $\sum_{i=1}^w \text{AMS}[j, i]^2$ . Finalmente, se toma la mediana de los  $d$  sumatorios obtenidos previamente. El tiempo necesario para calcular esta estimación es lineal respecto al espacio ocupado por la matriz de sketch,  $O(wd)$ .

El AMS sketch nos permite realizar consultas de punto aproximadas sobre un vector de frecuencias, al igual que el count-min sketch o el count sketch. El procedimiento a seguir, es idéntico al descrito en 3.6.2.3. Ciento es, que si necesitamos una estructura de datos sobre la que únicamente vamos a realizar consultas de punto, es más recomendable utilizar alguno de los data sketch mencionados con anterioridad.

### 3.6.2.3. Count Sketch

El propósito de esta estructura de datos es similar al del count-min sketch. Realizar una estimación sobre la frecuencia individual de un elemento,  $x$ . La principal diferencia de ambos data sketch reside en las garantías de precisión proporcionada para la estimación [10]. La estructura de datos subyacente es la misma, una matriz de contadores con profundidad,  $d$ , y anchura,  $w$ .

Adicionalmente a la familia de funciones hash encargada de mapear cada elemento,  $x$ , a una celda dentro de cada fila. Se añade una nueva familia de funciones hash, encargada de mapear cada elemento a  $\{-1, +1\}$ , igual que ocurre en el AMS Sketch. Estas funciones adicionales, modificarán el proceso de actualización y estimación. Los requisitos que deben de cumplir las familias de funciones hash  $h(x)$  y  $g(x)$  son los descritos en la sección 3.6.2.2.

## Fase de actualización

La similitud de este data sketch y el AMS sketch se fundamenta en que para realizar una estimación sobre la frecuencia de un elemento concreto,  $f_x$ . Esta se puede realizar mediante el producto interno de dos distribuciones de frecuencias [11], una de las aplicaciones de un AMS sketch. En otras palabras, existe un caso concreto, para el que se puede usar un AMS sketch para estimar  $f_x$ .

Debido a lo anterior, llevar a cabo una actualización de la forma  $\langle x, c \rangle$ , el procedimiento a seguir es idéntico al descrito en la Figura 3.6. Para cada una de las  $j$  filas presentes en la matriz de sketch con,  $1 \leq j \leq d$ , se actualiza la celda  $C[j, h_j(x)]$  de la forma descrita a continuación.

$$C[j, h_j(x)] = C[j, h_j(x)] + g_j(x) \cdot c \quad (3.9)$$

## Fase de estimación

El proceso a seguir para obtener una estimación del valor real,  $f_x$ , es el siguiente. Para cada fila,  $j$ , de la matriz de sketch con  $1 \leq j \leq d$ . Calculamos los valores de las funciones hash  $h_j(x)$  y  $g_j(x)$ . Y obtenemos el valor,  $v_j$ , como  $g_j(x) \cdot C[j, h_j(x)]$ . La

estimación final,  $\tilde{f}_x$ , se obtiene a partir de la mediana de los  $d$  valores, previamente calculados [11]. La Figura 3.7 muestra el proceso seguido a la hora de realizar una consulta.

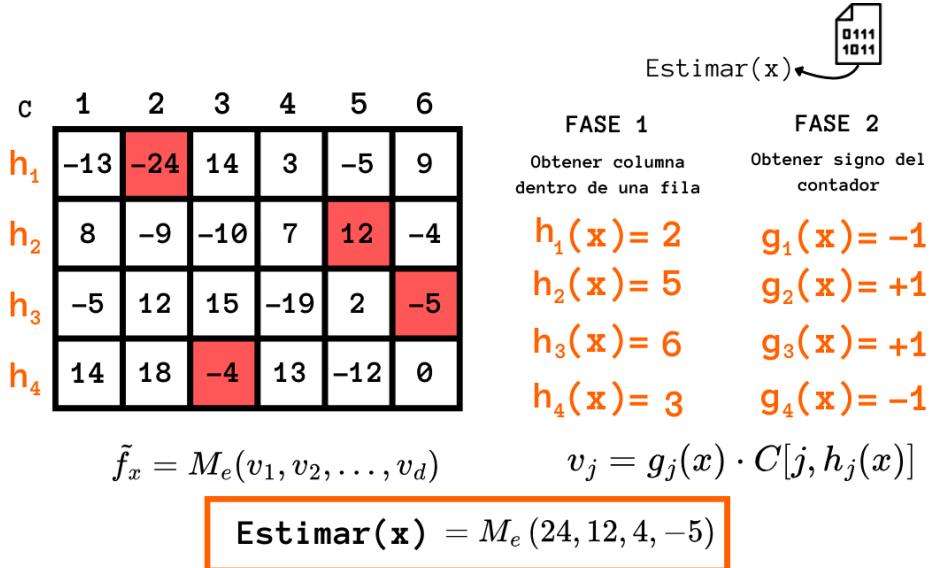


Figura 3.7: Consulta de punto sobre un Count Sketch

Como se comentó al inicio de esta sección, la diferencia respecto al Count-Min sketch reside en las garantías de precisión de cada estructura. Fijando los valores,  $d = \log \frac{4}{\delta}$  y  $w = \frac{2}{\epsilon^2}$ , se garantiza que la estimación,  $\tilde{f}_x$ , tenga un error máximo de  $\epsilon\sqrt{F_2} \leq \epsilon N$  con una probabilidad de al menos  $1 - \delta$  [11]. Siendo,  $F_2$ , la norma  $L_2^2$  del vector de frecuencias, explicada en 3.6.2.2. Para lograr esta garantía, las familias de funciones hash deben cumplir los requisitos expuestos anteriormente.

El espacio requerido por esta estructura de datos pertenece al orden  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$  y el tiempo necesario para llevar a cabo una consulta es de orden,  $O(\log \frac{1}{\delta})$ , en el peor de los casos.

### 3.6.3. Estimadores de cardinalidad

Al igual que ocurría con la estimación de frecuencias sobre conjuntos de datos de gran tamaño, determinar la cardinalidad de un conjunto con duplicados es un problema común y en aumento desde la expansión de los servicios de internet, donde miles de millones de búsquedas son realizadas diariamente por un número mucho menor de usuarios distintos. En este contexto, surge la necesidad de desarrollar algoritmos y estructuras de datos que puedan estimar la cardinalidad de un conjunto de datos en una cantidad de espacio sustancialmente menor que el número de elementos distintos y en tiempos razonables [35].

Las aplicaciones de esta estimación son muchas. Por ejemplo, si un sitio web que se dedica a la venta de productos necesita saber el número de clientes (direcciones IP diferentes) que han visitado un producto concreto, para detectar tendencias emergentes y estar preparado ante posibles problemas de inventario. O simplemente, se pretende

identificar segmentos de clientes únicos o grupos demográficos específicos para una personalización más efectiva de las estrategias de marketing y ventas. Los datos de visita de los usuarios tienden a crecer mucho. En un sitio web concurrido, una tabla de visitas diarias de un producto puede crecer hasta tener miles de millones de filas, y esta consulta particular podría llevar un tiempo.

Los algoritmos y estructuras de datos tradicionales, implementados por los sistemas de gestión de bases de datos más populares, apostaban por dar soluciones exactas a estos problemas. Con los volúmenes de datos masivos que se manejan actualmente, realizar consultas clásicas puede tomar días y tera-bytes de memoria. En este punto, surgen soluciones que tienen por objetivo ahorrar espacio a cambio de renunciar a algo de precisión. Casi siempre manteniéndose en el rango de unos pocos kilo-bytes mientras se alcanza la verdadera cardinalidad, con una pequeña tasa de error. A continuación, se proponen una serie de algoritmos o estructuras que tienen por objetivo abordar la problemática mencionada.

### 3.6.3.1. Flajolet-Martin Sketch

El algoritmo Flajolet-Martin sketch, es una técnica de estimación probabilística diseñada para estimar la cardinalidad de forma aproximada de un conjunto grande de elementos distintos, como un flujo de datos masivos. Fue uno de los primeros algoritmos diseñados para resolver este problema. Se propuso en [23] por Philippe Flajolet y G. Nigel Martin en el año 1984. Este algoritmo destaca por su sencillez y espacio necesario respecto a la precisión de los resultados generados. Ha servido como base para muchos otros algoritmos de estimación de la cardinalidad usados en la actualidad, como HyperLogLog (3.6.3.2).

La idea subyacente a este algoritmo se basa en que dada una buena distribución uniforme de números, la probabilidad de que el bit activo (bit que toma el valor 1) más a la derecha esté en la posición,  $\rho$ , 0 es  $\frac{1}{2}$ , la probabilidad de que esté en la posición 1 es  $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$ , en la posición 2 es  $\frac{1}{8}$  y así sucesivamente. En general, se puede decir que la probabilidad de que el bit activo más a la derecha, esté en la posición  $k$  en una distribución uniforme de números es

$$P[\rho = k] = 2^{-k-1} \quad (3.10)$$

Se puede observar que dicha probabilidad va disminuyendo en un factor de  $\frac{1}{2}$  con cada posición desde el bit menos significativo hasta el bit más significativo. Por lo tanto, si se registra la posición del bit activo más a la derecha,  $\rho$ , para cada elemento del conjunto de datos, Se espera que la probabilidad de que  $\rho = 0$  sea 0.5,  $\rho = 1$  sea 0.25, y así sucesivamente [26].

De esta forma, dicha probabilidad debería volverse nula cuando la posición del bit no activo más a la derecha,  $b$ , sea  $b > \log m$ , donde  $m$  es el número de elementos distintos en el conjunto de datos, el valor que se desea estimar. A partir de la posición de dicho bit,  $b$ , se puede intuir que el número de elementos únicos será aproximadamente  $2^b$ .

El procedimiento a seguir es muy sencillo. En primer lugar se necesita una función hash capaz de distribuir los elementos de manera uniforme. Para ello, se utiliza una

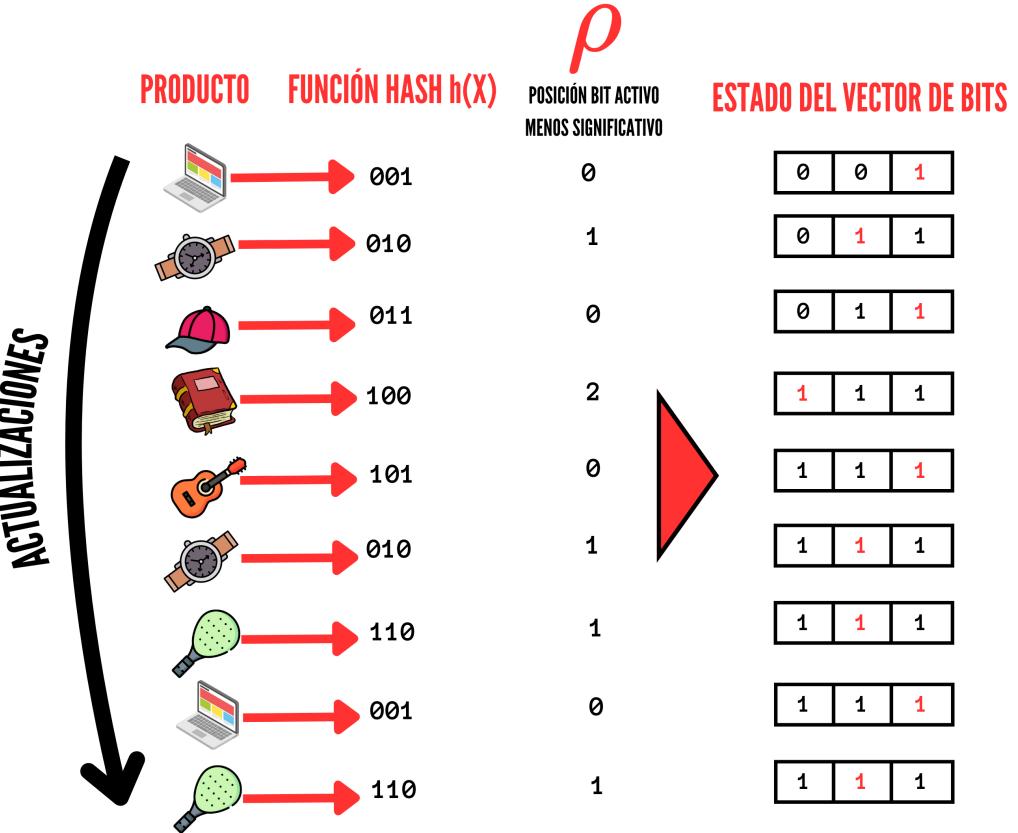


Figura 3.8: Flujo de actualización en un Flajolet-Martin sketch

función hash multiplicativa de la forma  $(ax + b) \bmod c$ , donde  $a$  y  $b$  son números impares y  $c$  es el límite del rango de dicha función hash. El valor  $c$  puede expresarse como  $2^L - 1$ , ya que el conjunto de enteros en el intervalo  $[0, \dots, 2^L - 1]$ , corresponde con el conjunto de cadenas binarias de  $L$  bits. Será necesario establecer un rango de hash lo suficientemente grande como para almacenar el máximo número posible de valores únicos, normalmente  $L = 64$  [23].

Cuando un nuevo elemento se recibe, similar a una actualización, este pasa a través de dicha función hash. Para el valor del hash obtenido, se localiza la posición del bit activo más a la derecha y se marca la posición correspondiente en el vector de bits como 1. Dicho vector o mapa de bits es la única estructura que utiliza el algoritmo de Flajolet-Martin. La Figura 3.8 muestra el estado del vector de bits en función al flujo de elementos.

En el momento en el que se deseé realizar una estimación, el vector de bits tendrá 1 en todas las posiciones correspondientes a la posición de cada bit activo más a la derecha para todos los elementos procesados hasta el momento. Se busca la posición,  $b$ , del bit más a la derecha con valor 0 en este vector de bits. Esta posición corresponde al bit más a la derecha que no hemos visto mientras procesamos los elementos [26]. Para obtener el valor de la estimación final,  $\tilde{a}$ , se necesita un factor de corrección adicional,  $\varphi \approx 0,77351$ , debido al amplio margen de error que tiene este algoritmo. Quedando la estimación final como:

$$\tilde{a} = \frac{2^b}{\varphi} \quad (3.11)$$

Debido al problema con el margen de error comentado anteriormente, se propone una mejora que consiste en utilizar diferentes funciones hash y combinar los resultados de las distintas ejecuciones, tomando la media o la mediana. Una gran característica de este algoritmo es que el resultado no depende del número de ocurrencias de un elemento concreto. En definitiva, esta técnica de sketch tiene un coste espacial  $O(\log m)$ , siendo  $m$  el número máximo de elementos distintos posibles.

### 3.6.3.2. HyperLogLog

HyperLogLog es un algoritmo de estimación de cardinalidad diseñado para contar la cantidad de elementos distintos en un conjunto de datos extremadamente grande. Aunque también puede verse como una estructura de datos probabilística. Este algoritmo se fundamenta en las propiedades probabilísticas y estadísticas de cadenas generadas aleatoriamente de forma uniforme, similares a las vistas en [3.6.3.1](#). Las funciones hash son la base de este algoritmo y aunque la implementación original utilizaba hashes de 32 bits, las implementaciones más recientes utilizan cadenas de 64 bits, para aumentar la precisión en aquellos casos que implican cardinalidades muy grandes [35].

Representamos nuestro conjunto de datos, sobre el cual queremos estimar una cardinalidad, como un conjunto con elementos repetidos ,  $S = \{x_1, x_2, \dots, x_n\}$ , que cuenta con un número  $n$  de elementos, de los cuales,  $k$ , son distintos. El valor de  $k$  es el que se desea conocer y con este algoritmo se conseguirá estimar, con un alto grado de precisión. Se necesita una función hash,  $h : U \rightarrow \{0, 1\}^L$ , siendo el  $L$ , el número de bits de nuestra cadena. Esta función hash se encarga de mapear, con una alta probabilidad, cada uno de nuestros elementos diferentes a una cadena de bits distinta. De esta forma, el número de hashes distintos tomará un valor muy cercano a  $k$ .

En las etapas primitivas de este algoritmo se optó por una técnica, denominada conteo probabilístico, que consistía en almacenar por cada uno de los hashes obtenidos, la posición del primer 1 comenzando por la derecha,  $p$ . Seleccionando el valor más alto,  $p_{max}$ , podríamos obtener una estimación de la cardinalidad,  $E = 2^{p_{max}}$ . Esta técnica se basaba en la idea de que si se logra obtener un hash inusual con muchos ceros finales, eso sería un indicador de la presencia de muchos otros hashes en el conjunto. Tener un elemento cuyo hash tenga  $p = p_{max}$ , implica una cardinalidad media de  $2^{p_{max}}$  [35]. Esta primera aproximación contaba con varios problemas:

1. Todas las estimaciones son potencias de 2, por lo que en muchas ocasiones, será imposible acercarse al valor real.
2. Existen valores atípicos que pueden producir un hash con muchos ceros a la derecha, y en consecuencia, disparar el valor estimado.

Con el fin de solventar estos problemas, se propuso una solución que divide la totalidad de los hashes que se pueden obtener en,  $m = 2^b$ , subconjuntos, donde  $b$  es el número de bits, empezando por la izquierda, que identifican a los hashes de cada uno de los subconjuntos. Tenemos entonces,  $m$  estimadores (valores  $p_{max}$ ). Sobre los cuales se calcula la media aritmética.

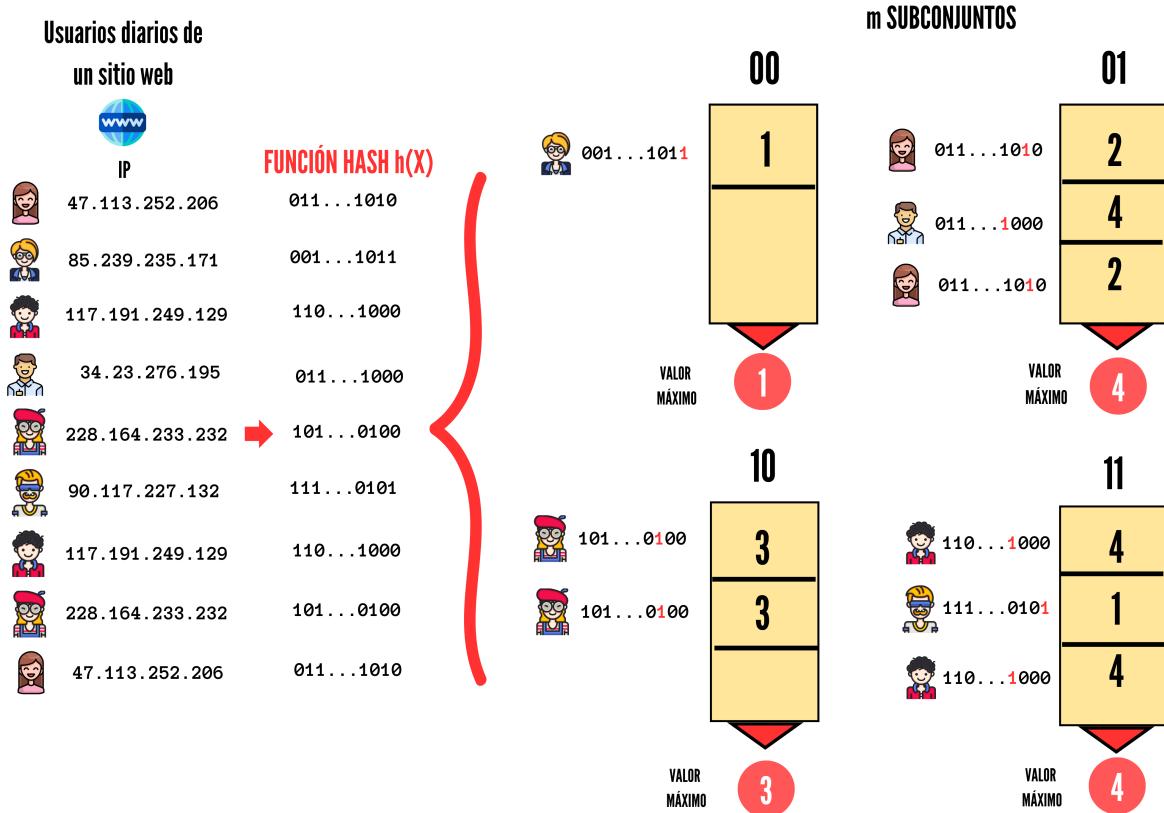


Figura 3.9: Distribución por subconjuntos en HyperLogLog

$$M_A = \frac{\sum_{i=1}^m p_{i\max}}{m} \quad (3.12)$$

La figura 3.9, muestra un ejemplo de esta estructuras de subconjuntos para un caso en el que se desea conocer los usuarios únicos de un sitio web. Hay que tener en cuenta que la estructura almacena únicamente los valores máximos de cada subconjunto.

Para obtener la estimación general, se necesita tener en cuenta todos los subconjuntos y sus respectivos estimadores. Adicionalmente, se incorpora una constante de normalización,  $\tilde{a}_m$  que busca corregir la sobreestimación introducida por el uso de variables aleatorias. Más adelante se discutirá el valor de esta constante.

$$E = \tilde{a}_m \cdot m \cdot 2^{M_A} \quad (3.13)$$

Sin embargo, la media aritmética es muy susceptible frente a valores atípicos. Una solución ante esto consiste en eliminar un número determinado de estimadores, aquellos con valores más altos, y realizar la estimación con el resto. Aunque la solución que resulta en el HyperLogLog utilizado actualmente, utiliza la media armónica en lugar de la media aritmética. Por lo tanto, se calcula la media de los subconjuntos,  $M_A$ .

$$M_A = \frac{m}{\sum_{i=1}^m 2^{-p_{i\max}}} \quad (3.14)$$

La estimación final, que nos dará una aproximación del número de elementos distintos existentes en un conjunto de datos es  $E = a_m \cdot m \cdot 2^{M_A}$ , donde el valor de  $a_m$  se obtiene a partir de la Ecuación 3.15 [23].

$$a_m = \frac{1}{2 \ln 2(1 + \frac{1}{m}(3 \ln 2 - 1) + O(m^{-2}))} \quad (3.15)$$

Aunque para valores de  $m$  lo suficientemente grandes,  $m \geq 128$ ,  $a_m = \frac{0,723}{(1 + \frac{1,079}{m})}$ . Pese a que esta última alternativa puede proporcionar peores resultados para casos con conjuntos de datos pequeños, a medida que los conjuntos de datos se hacen más grandes, proporciona una estimación menos sesgada y con un error relativo menor. Las diferentes pruebas que se muestran en la publicación [24], estiman un valor del error relativo de alrededor de  $\frac{1,04}{\sqrt{m}}$ , el cual es un valor ideal para aquellas aplicaciones que no necesitan una exactitud total respecto al valor real.

Además, es posible determinar el orden espacial de la estructura necesaria en función de un valor de cardinalidad máximo,  $k_{max}$ . Si determinamos dicho valor de forma correcta, se reducirá el riesgo de sufrir colisiones. Partiendo de dicho valor máximo, podemos definir la longitud de los hashes a utilizar como  $O(\log_2 k_{max})$ . Para representar los  $m$  estimadores,  $p_{imax}$ , de la totalidad de los subconjuntos, se necesita una estructura de  $O(m \log_2 \log_2 k_{max})$  bits. Un tamaño muy reducido respecto a la precisión de la estimación que proporciona.

### 3.6.3.3. Theta Sketch

Los Theta Sketch son estructuras de datos probabilísticas, similares a los HyperLogLog, que han estado ganando más aceptación recientemente. A diferencia de HyperLogLog, los Theta Sketch ofrecen la posibilidad de realizar estimaciones de cardinalidad sobre intersecciones de conjuntos de datos, obteniendo una aproximación dentro de un margen de error [13].

Un Theta Sketch se define como un par  $(\theta, S)$ , donde  $0 < \theta < 1$  es un valor umbral y  $S$  es el conjunto de todos los elementos únicos de la secuencia que tras aplicarles una función hash,  $0 < h(x) < 1$ , son menores que  $\theta$ . Inicialmente, cuando el data sketch se encuentra,  $\theta = 1,0$ . Después de que la estructura se haya llenado con  $k$  valores mínimos,  $\theta$  sigue siendo 1,0. Cuando el próximo valor único entrante debe ser insertado en el sketch, el valor mínimo en la posición,  $k + 1$ , se asigna a  $\theta$  y se elimina de la memoria.

La Figura 3.10 muestra el proceso a seguir cuando se inserta un elemento en un Theta Sketch, con  $k = 5$ . En un primer instante,  $S$  almacena los 5 valores únicos más pequeños computados hasta el momento y  $\theta = 0,6254$ . Cuando llega un elemento, este puede haber sido visto previamente o no. Se calcula la función hash sobre dicho elemento, si el valor resultante es mayor que  $\theta$ , este se descarta. Si por el contrario, es menor que  $\theta$ , se inserta el elemento en  $S$ . Al ser  $S$  un conjunto, no puede contener elementos repetidos, por lo que en caso de que el elemento ya existiera previamente,  $S$  se mantiene igual. En caso de haber añadido el elemento a  $S$ , se asigna un nuevo valor a  $\theta$ , el valor mínimo de  $S$  en la posición  $k + 1$ .

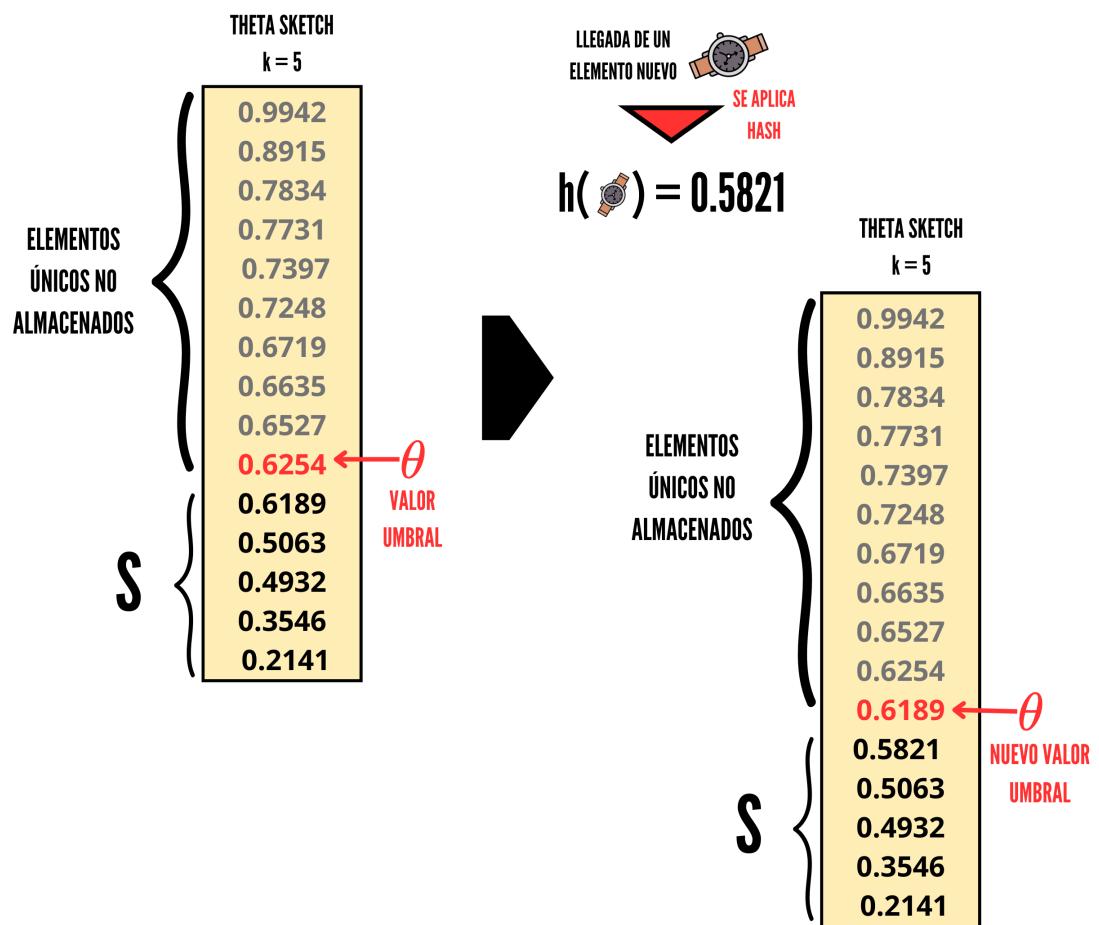


Figura 3.10: Actualización en un Theta Sketch

El valor,  $k$ , es un parámetro de configuración especificado por el usuario, que es utilizado para determinar la precisión objetivo y el tamaño máximo del data sketch. Para obtener una estimación de la cardinalidad del conjunto de datos,  $\tilde{a}$ , bastará con:

$$\tilde{a} = \frac{|S|}{\theta} \quad (3.16)$$

Como se introdujo anteriormente, la precisión obtenida en la estimación dependerá del valor que tome  $k$ . De tal forma que,  $\frac{\sigma_{\tilde{a}}}{\tilde{a}} < \frac{1}{\sqrt{k-1}}$  [44].

### 3.7. Comparativa de los data sketches expuestos

A partir de los diferentes data sketches expuestos hasta el momento y las propiedades presentadas en la sección 3.4, la Tabla 3.1 recoge un breve resumen de las propiedades de los mismos. Un detalle importante es el hecho de que no se ha usado la misma notación para hacer referencia a elementos comunes en todos los data sketches. Por ejemplo, en los data sketches utilizados para estimar la frecuencia de un elemento,  $d$ , hace referencia al número de funciones hash empleadas, mientras que en los filtros de Bloom se utiliza,  $k$ .

Para una correcta compresión de la Tabla, se recomienda revisar las secciones asociadas a cada data sketch, donde se define la notación empleada y se razonan los resultados obtenidos en términos de espacio y tiempo. También es importante destacar las métricas de error que se obtienen de cada una de las técnicas, estas no se han incluido en la Tabla debido a que varían bastante de una a otra.

	Espacio requerido	Tiempo de actualización	Tiempo de consulta	Tipo de estimación	Estado inicial
<b>Filtros de Bloom</b>	$O(m)$	$O(k)$	$O(k)$	Existencia en un conjunto	Mapa de bits o vector de contadores establecido a 0
<b>Count-min Sketch</b>	$O\left(\frac{e \ln\left(\frac{1}{\delta}\right)}{\epsilon}\right)$	$O(d)$	$O(d)$	Frecuencia	Matriz de Sketch establecida a 0
<b>AMS Sketch</b>	$O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$	$O(d)$	$O(wd)$	Norma $L_2^2$	Matriz de Sketch establecida a 0
<b>Count Sketch</b>	$O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$	$O(d)$	$O(\log \frac{1}{\delta})$	Frecuencia	Matriz de Sketch establecida a 0
<b>Flajolet-Martin Sketch</b>	$O(\log m)$	$O(1)$	$O(L)$	Cardinalidad	Mapa de bits establecido a 0
<b>HyperLogLog</b>	$O(m \log_2 \log_2 k_{max})$	$O(1)$	$O(m)$	Cardinalidad	$m$ contadores establecidos a 0
<b>Theta Sketch</b>	$O(k)$	$O( S )$	$O(1)$	Cardinalidad	$S = \emptyset$

Tabla 3.1

*Comparativa entre data sketches*

Una propiedad fundamental a la hora de evaluar el rendimiento de un data sketch es el margen de utilidad o error que este proporciona. Obtener una fórmula rigurosa para esta métrica es una tarea, en ciertos casos, complicada. La Tabla 3.2 recoge algunas métricas, junto a su fórmula, que permiten evaluar los diferentes métodos expuestos en el capítulo. A excepción del AMS Sketch, ya que la utilidad de este método depende directamente de la naturaleza de los datos. Para una mayor comprensión de la notación

Data Sketch	Métrica de utilidad/error	Fórmula
Filtro de Bloom	Tasa de falsos positivos	$f \approx (1 - e^{-\frac{nk}{m}})^k = e^{k \ln(1 - e^{-\frac{kn}{m}})}$
Count-Min Sketch	Margen de sobreestimación	$\tilde{a} \leq a + \varepsilon \ a\ _1$ , con probabilidad, $p \geq 1 - \delta$
Count Sketch	Error máximo	$\varepsilon \sqrt{F_2} \leq \varepsilon N$ , con probabilidad de, $1 - \delta$
Flajolet-Martin Sketch	Error relativo ( $\epsilon$ )	$\epsilon \approx \frac{1}{\sqrt{L}}$
HyperLogLog	Error relativo ( $\epsilon$ )	$\epsilon \approx \frac{1,04}{\sqrt{m}}$
Theta Sketch	Varianza de la estimación	$\frac{\sigma_{\tilde{a}}}{a} < \frac{1}{\sqrt{k-1}}$

Tabla 3.2

*Métricas de error/utilidad asociadas a los diferentes data sketches*

utilizada en las fórmulas, es recomendable revisar las secciones específicas de cada técnica.

## 4. Data sketches diferencialmente privados

---

A lo largo de todo el documento, se ha destacado la importancia que ha adquirido la privacidad de los datos en los sistemas de información actuales. De la misma forma que las empresas más importantes en el sector tecnológico han comenzado a utilizar estructuras de datos probabilísticas, en especial, los data sketches, para realizar análisis eficientes de la gran cantidad de datos generados por sus usuarios.

Es decir, nos encontramos en una situación en la que las empresas necesitan almacenar y, posteriormente, analizar la información de sus usuarios, una cantidad de información masiva que se envía continuamente. Pero, además, estas empresas deben garantizar la seguridad y privacidad de dichos datos.

A lo largo de este capítulo se presentarán una serie de algoritmos propuestos por empresas punteras como Apple, Google o Microsoft que tienen por objetivo garantizar la privacidad de los datos durante un análisis efectivo y eficiente. Mediante la combinación de data sketches y técnicas que aseguran la privacidad  $\varepsilon$ -diferencial. Esto supone un avance importante en la unión entre la protección de la información personal y la eficiencia, tomando protagonismo en diversas áreas como el análisis de datos o la inteligencia artificial.

### 4.1. Private Count Mean Sketch

Este algoritmo fue propuesto por el equipo de desarrollo de Apple en [15]. El objetivo final de este algoritmo es la obtención de un histograma de las frecuencias de una serie de elementos conocidos, pertenecientes a un dominio,  $D$ . A partir de un flujo de datos de  $n$  registros, en un momento determinado, la empresa desea conocer las estadísticas sobre aquellos elementos con mayor frecuencia.

Este algoritmo tiene dos bloques con funcionalidad bien diferenciada. El primero de ellos se ejecuta en el dispositivo de los usuarios y tiene por finalidad, asegurar que la información que se transmite hacia el servidor sea  $\varepsilon$ -diferencialmente privada.

Al servidor, llega la información de los usuarios privatizada, donde se suprime cualquier información personal como la dirección IP de origen. Para almacenar la información se utiliza un data sketch, muy similar al de 3.6.2.1. Las dimensiones de la matriz utilizada,  $M$ , es de  $k \times m$ , siendo,  $k$ , el número de funciones hash existentes. Este conjunto de funciones hash se seleccionan previamente a la ejecución del algoritmo y se comparten entre la parte del cliente y del servidor. Estas funciones hash mapean los elementos,  $D \rightarrow [m]$ . A continuación, se detalla el funcionamiento de cada uno de los dos bloques, apoyado en un pseudocódigo.

Para la parte del cliente, el Algoritmo 1 representa los pasos necesarios. En primer lugar se requiere una familia de funciones hash independientes 3 a 3 entre sí,  $H$ , previamente seleccionada, los parámetros  $\varepsilon$  y  $m$  y, por supuesto, el dato a privatizar,  $d$ . Se selecciona un valor aleatorio,  $j$  (Pasos 1 y 2), que servirá como posterior índice. Se

genera un vector de  $m$  elementos, con todos sus valores a  $-1$  (Paso 3). Posteriormente, se establece la posición  $h_j(d)$  del vector, previamente inicializado, con el valor 1 (Paso 4). Es decir se selecciona una función hash aleatoria de  $\mathcal{H}$  y se establece a 1 la posición devuelta por dicha función hash. Por último se genera un vector de Bernoulli de  $m$  elementos y se genera un nuevo vector  $\tilde{v}$  como resultado del producto, elemento a elemento, de los vectores  $v$  y  $b$  (Pasos 5 y 6). Como salida, el algoritmo envía al servidor el par  $\langle \tilde{v}, j \rangle$ .

---

**Algoritmo 1:** Cliente (Private Count Mean Sketch)

---

**Input:** Elemento  $d \in D$ ,  $\varepsilon$ ,  $m$ ,  $\mathcal{H}$

**Output:** Vector  $\tilde{v}$ , índice  $j$

- 1  $k \leftarrow |\mathcal{H}|$ ;
  - 2 Seleccionar un entero,  $j$ , aleatorio en el intervalo  $[0, k]$ ;
  - 3  $v \leftarrow \{-1\}^m$ ;
  - 4 Establecer el valor  $v_{h_j(d)} \leftarrow 1$ ;
  - 5 Generar el vector  $b \in \{-1, 1\}^m$ , con una probabilidad de  $b_x = 1$  igual a  $\frac{e^{\varepsilon/2}}{e^{\varepsilon/2} + 1}$ ;
  - 6  $\tilde{v} \leftarrow \{v_1 \cdot b_1, v_2 \cdot b_2, \dots, v_m \cdot b_m\}$ ;
- 

En el lado del servidor existe una matriz de sketch,  $M \in \mathbb{R}^{k \times m}$ , la cual se va actualizando en función de las sucesivas actualizaciones del flujo de datos. Esta matriz se inicializó con todos sus valores a 0. El Algoritmo 2 muestra el impacto de una actualización en la matriz,  $M$ . En primer lugar, se calcula una constante de privacidad  $c_e$ , que se utilizará, junto con el vector  $\tilde{v}$  recibido, para generar un nuevo vector que mantenga la utilidad de la información (Pasos 1 y 2). Por último, se incrementa la fila de la matriz de sketch correspondiente a la función hash,  $j$ , con los valores del vector generado en el paso anterior.

---

**Algoritmo 2:** Actualización matriz de sketch (Private Count Mean Sketch)

---

**Input:** Par  $\langle \tilde{v}, j \rangle, \varepsilon, k, m$

- 1  $c_e \leftarrow \frac{e^{\varepsilon/2} + 1}{e^{\varepsilon/2} - 1}$ ;
  - 2  $\tilde{x} \leftarrow k \cdot \left( \frac{c_e}{2} \cdot \tilde{v} + \frac{1}{2} \cdot \mathbf{1} \right)$ ;
  - 3 **for**  $i \in [m]$  **do**
  - 4    $M_{j,i} \leftarrow M_{j,i} + \tilde{x}_i$ ;
- 

Para obtener una estimación sobre la frecuencia de un elemento,  $d$ , bastaría con realizar el siguiente cálculo:

$$\tilde{f}(d) = \left( \frac{m}{m-1} \right) \left( \frac{1}{k} \sum_{i=1}^k M_{i,h_i(d)} - \frac{n}{m} \right) \quad (4.1)$$

donde el sumatorio representa un promedio de todas las funciones hash para dicho elemento y  $n$ , el número de actualizaciones sobre,  $M$ , hasta el momento. Para obtener el histograma de frecuencias de una serie de elementos pertenecientes a un subconjunto del dominio de estudio,  $\hat{D} \subseteq D$ , bastaría con calcular  $\tilde{f}(d), \forall d \in \hat{D}$ .

La ecuación 4.2, hace referencia a la varianza de la estimación sobre la frecuencia de un elemento concreto,  $f(d)$ . Esta varianza dependerá en gran medida de los valores seleccionados para cada uno de los hiperparámetros  $m$ ,  $k$  y  $\varepsilon$ , además del número de

elementos procesados hasta el momento,  $n$ . Una característica importante que puede influir en la utilidad de este algoritmo es la norma  $L_2$  de la distribución de los datos subyacente. Esta se relaciona con el término  $\sum_{d' \in D} f(d')^2$  y está estrechamente relacionada con la dispersión en las frecuencias de los elementos de nuestro conjunto de datos. Cuanto mayor sea este valor, se necesitará un aumento en el factor,  $k \times m$ , para garantizar una varianza reducida en la estimación, lo que ocasiona un aumento en el ancho de banda de los dispositivos.

$$\text{Var} [\tilde{f}(d)] \leq \left( \frac{m}{m-1} \right)^2 \times \left( \frac{e^{\varepsilon/2}}{(e^{\varepsilon/2}-1)^2} + \frac{1}{m} + \frac{\sum_{d' \in D} f(d')^2}{n \cdot k \cdot m} \right) \times n. \quad (4.2)$$

El equipo de analistas de Apple ha utilizado este algoritmo para obtener estadísticas sobre los emoticonos más populares en diferentes regiones del planeta, estableciendo los valores  $m = 1024$ ,  $k = 65,536$  y  $\varepsilon = 4$ , con el objetivo de mejorar la experiencia de los usuarios mejorando el teclado predictivo de sus dispositivos [15]. También ha sido utilizado para detectar un conjunto actualizado de sitios web para los cuales la mayoría de los usuarios prefieren la reproducción automática de multimedia en todo el mundo. O para determinar los tipos de datos de salud populares, para impulsar futuras mejoras en la aplicación de salud de su propio sistema. Este último caso de uso resulta bastante interesante al tratar con datos sobre la salud de sus usuarios, una situación en la que la privacidad de los mismos se vuelve indispensable.

## 4.2. Private Hadamard Count Mean Sketch

A partir del problema con el ancho de banda que puede afectar notablemente a la parte del cliente, comentado en la sección 4.1, el equipo de desarrollo de Apple propone una variante del algoritmo Count Mean Sketch, Hadamard Count Mean Sketch [15]. La idea propuesta consiste en reducir la cantidad de datos que el cliente necesita enviar al servidor. En lugar de enviar un vector de bits, el algoritmo propuesto permite que el cliente solo envíe un solo bit. Esto reduce significativamente la carga de comunicación y el ancho de banda requerido.

Para ello, el algoritmo utiliza la transformada de Hadamard. Esta transformada ayuda a difundir la información contenida en un vector disperso a lo largo de múltiples dimensiones. Un vector disperso es aquel que tiene la mayoría de sus elementos como ceros y solo unos pocos elementos diferentes de cero. Al transformar el vector usando la matriz de Hadamard, la información se distribuye de manera más uniforme a través del vector transformado. Esto asegura que, aunque solo se envíe un bit al servidor, este bit puede proporcionar una cantidad significativa de información sobre el vector original.

La transformada de Hadamard es una transformación lineal que se representa mediante la multiplicación de un vector por una matriz de Hadamard. La matriz de Hadamard es una matriz cuadrada que toma valores,  $\{-1, 1\}$ , y es ortogonal. Esta matriz se define recursivamente de la siguiente forma:

$$H_n = \begin{cases} H_1 = \begin{bmatrix} 1 \end{bmatrix}, & \text{si } n = 1 \\ \begin{bmatrix} H_{\frac{n}{2}} & H_{\frac{n}{2}} \\ H_{\frac{n}{2}} & -H_{\frac{n}{2}} \end{bmatrix}, & \text{si } n > 1 \text{ y } n \text{ es una potencia de 2} \end{cases} \quad (4.3)$$

El Algoritmo del lado del cliente, 3, es bastante similar al Algoritmo 1. En esta nueva versión, se añade un índice,  $l$ , que determina la posición del vector obtenido tras aplicar la transformada de Hadamard, que se va a privatizar y, posteriormente, enviar al servidor. Destacar que la notación empleada en este algoritmo es idéntica a la que se define en 4.1. A diferencia del caso anterior, el vector  $v$  se inicializa con el valor 0 (Paso 2), y se genera un nuevo vector  $w$ , a partir de la multiplicación de la matriz de Hadmard y  $v$  (Paso 4).

---

**Algoritmo 3:** Cliente (Hadamard Count Mean Sketch)

---

**Input:** Elemento  $d \in D$ ,  $\varepsilon$ ,  $k$ ,  $m$ ,  $H$ ,  $\mathcal{H}$

**Output:**  $\tilde{w}$ , índice  $j$ , índice  $l$

- 1 Seleccionar un entero,  $j$ , aleatorio en el intervalo  $[0, k]$ ;
  - 2  $v \leftarrow \{0\}^m$ ;
  - 3  $v_{h_j(d)} \leftarrow 1$ ;
  - 4  $w \leftarrow H_m v$ ;
  - 5 Seleccionar un entero,  $l$ , aleatorio en el intervalo  $[0, m]$ ;
  - 6 Seleccionar un entero,  $b \in \{-1, 1\}$ , con  $P[b = 1] = \frac{e^\varepsilon}{e^\varepsilon + 1}$ ;
  - 7  $\tilde{w} \leftarrow bw_l$ ;
- 

De la misma forma que ocurre en el algoritmo Private Count Mean Sketch, en el lado del servidor debe existir una matriz de sketch,  $M^H \in \mathbb{R}^{k \times m}$ , la cual se va modificando en función de las sucesivas actualizaciones del flujo de datos. Esta matriz se inicializó con todos sus valores a 0. El Algoritmo 4 muestra el impacto de una actualización en dicha matriz. A diferencia que en el Algoritmo 2, no se genera un vector, sino un elemento (Paso 2). Y por lo tanto, solo se modifica un elemento de la matriz,  $M^H$ , consiguiendo una mejora considerable en el coste temporal del algoritmo de actualización, al pasar de  $O(m)$  a  $O(1)$  (Paso 3).

---

**Algoritmo 4:** Actualizar matriz de sketch (Hadamard Count Mean Sketch)

---

**Input:** Tupla  $(\tilde{w}, j, l)$ ,  $\varepsilon$ ,  $k$ ,  $m$ ,  $H$

- 1  $c_e \leftarrow \frac{e^\varepsilon + 1}{e^\varepsilon - 1}$ ;
  - 2  $\tilde{x} \leftarrow k \cdot c_e \cdot \tilde{w}$ ;
  - 3  $M_{j,l}^H \leftarrow M_{j,l}^H + \tilde{x}$ ;
- 

Para consultar la estimación de la frecuencia de un elemento,  $d \in D$ , en primer lugar, debemos realizar una transformación sobre la matriz,  $M^H$ . El Algoritmo 5, describe el proceso seguido para obtener una estimación,  $\tilde{f}(d)$ . Debido a que la información original sufre modificaciones mediante la transformada de Hadamard para su optimizar su procesamiento, es necesario revertir esta transformación para obtener una representación original de los datos antes de calcular la estimación. Multiplicar la matriz,  $M^H$  por la transpuesta de la matriz de Hadamard es parte del proceso de inversión de la transformada de Hadamard (Paso 1).

---

**Algoritmo 5:** Calcular estimación (Hadamard Count Mean Sketch)

---

**Input:** Elemento  $d \in D$ ,  $M^H$ ,  $\varepsilon$ ,  $k$ ,  $m$ ,  $n$ ,  $H$ ,  $\mathcal{H}$

**Output:**  $\tilde{f}(d)$

1  $M^H \leftarrow M^H H_m^T$ ;

2  $\tilde{f}(d) = \left( \frac{m}{m-1} \right) \left( \frac{1}{k} \sum_{i=1}^k M_{i,h_i(d)}^H - \frac{n}{m} \right);$

---

La matriz,  $M^H$ , sobre la que el Algoritmo 5 trabaja, se puede ver como una variable auxiliar o local a la estimación, siendo una copia de la matriz global de sketch que va recogiendo las sucesivas actualizaciones, en el instante en el que se han procesado,  $n$ , actualizaciones. Aunque estos aspectos dependerán de la implementación concreta.

De la misma forma que en 4.1, es posible obtener un ecuación que determina la varianza de la estimación sobre la frecuencia de un elemento concreto.

$$\text{Var} [\tilde{f}(d)] \leq \left( \frac{m}{m-1} \right)^2 \times \left( \left( \frac{e^\varepsilon + 1}{e^\varepsilon - 1} \right)^2 + \frac{\sum_{d' \in D} f(d')^2}{n \cdot k \cdot m} \right) \times n \quad (4.4)$$

Siendo  $f(d)$ , la frecuencia real de un elemento,  $d$ . Se obtiene una varianza similar a la obtenida en 4.1. Sin embargo, el término que depende del parámetro de privacidad,  $\varepsilon$ , ha aumentado. El principal beneficio de este algoritmo es que el ancho de banda necesario en la comunicación cliente-servidor, no crece en función del parámetro,  $m$ .

Apple ha empleado este algoritmo para mejorar la experiencia del usuario en Safari. Con el fin de detectar los sitios web con mayor consumo de recursos, en iOS 11 y macOS High Sierra [15]. Se utiliza este algoritmo para identificar dos tipos de dominios: aquellos que sobrecargan la memoria y los que causan un exceso de consumo de energía debido al uso intensivo de la CPU. Sobre un conjunto de 250.000 dominios web, se ha aplicado este algoritmo, con los hiperparámetros,  $m = 32,768$ ,  $k = 1024$  y  $\varepsilon = 4$ . Utilizando solo 1 bit para mantener la privacidad de los usuarios. Los resultados obtenidos revelan que los sitios que más recursos consumen incluyen plataformas de transmisión de vídeo, tiendas en línea y sitios de noticias.

### 4.3. Private Sequence Fragment Puzzle

Los dos algoritmos expuestos hasta el momento, tienen por objetivo estimar la frecuencia de un elemento,  $d$ , dentro de un conjunto de datos conocido. La empresa Apple propone un algoritmo, que se apoya en el Count Mean Sketch, para obtener un conjunto de elementos populares, junto con sus valores estimados de frecuencia. Este algoritmo se denomina, Private Sequence Fragment Puzzle, ya que su funcionamiento guarda una similitud con las piezas de un rompecabezas.

Apple propuso este algoritmo con el objetivo de descubrir palabras que comúnmente los usuarios escriben, pero que no se encuentran en ningún diccionario [15]. Este tipo de palabras pueden corresponder con abreviaciones, expresiones populares, tendencias relacionadas con el momento o palabras de otro idioma. Durante dicho análisis, se utilizaron los valores para los parámetros,  $(m, m') = (1024, 1024)$ ,  $(k, k') = (2048, 2048)$ ,

y  $(\varepsilon, \varepsilon') = (2, 6)$ . El algoritmo fue desarrollado, en concreto, para analizar palabras de un máximo de 10 caracteres. En caso de que la palabra ocupe más caracteres se truncaría o, de lo contrario, se llenaría con espacios en blanco.

Sin embargo, este algoritmo puede ser utilizado en multitud de contextos, en los que interese conocer cadenas de texto populares pertenecientes a un dominio desconocido o demasiado extenso. Por ejemplo, podría ser utilizado para determinar, dentro de un sitio web de compras, qué artículos son los más buscados pese a no estar disponibles. Con el objetivo futuro de apostar por ellos e incluirlos en el inventario.

Como en todos los métodos vistos, se necesita un algoritmo que se ejecuta en el cliente, encargado de enviar a un servidor la información sensible privatizada. Este algoritmo trabaja con cadenas de texto de 10 caracteres, para las cuales se utiliza la notación,  $s$ . Todas las posibles cadenas de texto de 10 caracteres forman el dominio,  $D$ . Hay que tener en cuenta que,  $D$ , está formado por  $26^{10}$  combinaciones posibles, una cantidad intratable con las estructuras de datos convencionales. La notación,  $s[i : j]$ , hace referencia a los caracteres de  $s$  comenzando en el índice  $i$  hasta  $j$ .

El Algoritmo correspondiente a la parte del cliente, 6, recibe como entrada la cadena que se enviará al servidor, los parámetros de privacidad  $(\varepsilon, \varepsilon')$  y  $(m, m')$ , las familias de funciones hash 3-independientes, con las mismas características que en 4.1,  $(\mathcal{H}, \mathcal{H}')$  y una función hash,  $h : D \rightarrow [256]$ , que mapea las cadenas a un entero de 8 bits en el intervalo  $[0, 255]$ .

El hecho de que se necesiten el doble de parámetros de privacidad y familias de funciones hash, es debido a que se privatizan dos registros. Por un lado, el registro denominado secuencia que corresponde con la cadena,  $s$ . Y por otro lado, el fragmento, que surge de la concatenación de una subcadena de  $s$  de dos caracteres y la salida de la función hash para el valor  $s$ ,  $h(s) \parallel s[l : l + 1]$ , siendo,  $l$ , un índice impar entre 0 y 9. Para privatizar ambos registros, se hace uso del algoritmo correspondiente al cliente del Private Count Mean Sketch, cuya notación será,  $CMS_{cliente}$ .

---

#### **Algoritmo 6:** Cliente (Private Sequence Fragment Puzzle)

---

**Input:** Cadena  $s \in D$ ,  $(\varepsilon, \varepsilon')$ ,  $(m, m')$ ,  $(\mathcal{H}, \mathcal{H}')$ ,  $h$

**Output:** Tupla  $(CMS_{cliente}(r, \varepsilon', \mathcal{H}'), CMS_{cliente}(s, \varepsilon, \mathcal{H}), l)$

1 Seleccionar un entero aleatoriamente,  $l \in \{1, 3, 5, 7, 9\}$ ;

2  $r \leftarrow h(s) \parallel s[l : l + 1]$ ;

---

En este punto, es interesante destacar la propiedad de composición de la privacidad diferencial [19], que permiten combinar múltiples mecanismos privados mientras se controla el impacto acumulado en la privacidad total. Una propiedad muy útil en contextos donde se aplican múltiples operaciones sobre los datos, ya que permite mantener una garantía global sobre la privacidad. En el algoritmo previamente presentado, primero se utiliza un mecanismo para enviar el fragmento,  $\varepsilon'$ , y luego otro para enviar la secuencia completa,  $\varepsilon$ . La composición de estos dos mecanismos implica que el parámetro de privacidad total sea la suma de los parámetros individuales,  $(\varepsilon + \varepsilon')$ .

En la parte del servidor, donde llegan los registros privatizados, alberga una serie de matrices de sketch, de igual manera que en 4.1, sobre las que se trabaja a la hora de construir el diccionario con los elementos más frecuentes. Una matriz general,  $M$ ,

inicializada con los parámetros  $\varepsilon$ ,  $k$  y  $m$ , que se va actualizando mediante los registros privatizados,  $(CMS_{cliente}(s, \varepsilon, \mathcal{H})$ , que se denotan como  $\beta$ . Por otro lado, existen tantas matrices de sketch,  $M_l$ , como posibles valores de,  $l$ , existan. Estas últimas matrices se inicializan con los parámetros  $\varepsilon'$ ,  $k'$  y  $m'$ , y se actualizan a partir de los registros,  $CMS_{cliente}(r, \varepsilon', \mathcal{H}')$ , denotados por el símbolo,  $\alpha$ .

El Algoritmo encargado de realizar las sucesivas actualizaciones sobre las estructuras de datos descritas anteriormente, se describe en 7. A lo largo del Algoritmo, se hace uso de sucesivas llamadas al método encargado de actualizar la matriz de sketch en Private Count Mean Sketch, 2, cuya notación será,  $CMS_{servidor}$ . Se lleva a cabo la actualización de la matriz de sketch,  $M$ , con la cadena privatizada y de la matriz  $M_l$  pertinente, según el valor de  $l$ .

---

**Algoritmo 7:** Servidor (Private Sequence Fragment Puzzle)

---

**Input:** Tupla  $(\alpha, \beta, l), (\varepsilon, \varepsilon'), (m, m'), (k, k'), (\mathcal{H}, \mathcal{H}')$

- 1  $M \leftarrow CMS_{servidor}(\beta, \varepsilon, k, m);$
  - 2  $M_l \leftarrow CMS_{servidor}(\alpha, \varepsilon', k', m');$
- 

En un momento determinado, el equipo de analistas que se quiera beneficiar de este algoritmo, solicitará al equipo de desarrollo los resultados. Para ello, se propone un Algoritmo, 8, encargado de construir un diccionario,  $\chi$ , donde se almacena una estimación de las cadenas más enviadas por los usuarios, junto a la frecuencia estimada de estas.

En primer lugar, definimos como  $\tilde{f}$  y  $\tilde{f}_l$ , a los algoritmos encargados de realizar una consulta sobre las matrices de sketch  $M$  y  $M_l$ , respectivamente, definidos en 4.1 y 4.2. De ahí que el Algoritmo reciba como entrada los diferentes parámetros requeridos para la estimación. Adicionalmente, se recibe un valor umbral,  $T$ , que determinará la cantidad de cadenas estimadas a generar.

Inicialmente, se inicializa el diccionario como un conjunto vacío (Paso 1). Para cada uno de los valores de  $l$  posibles, se crea un conjunto,  $Q_l$ . A cada uno de dichos conjuntos, se le añaden los  $T$  registros  $(w||c)$  con mayor valor de frecuencia estimada, sobre la matriz  $M_l$  que corresponda (Pasos 2, 3 y 4). Esto puede verse como la construcción de un histograma auxiliar para todas las combinaciones posibles de un número entero en el intervalo  $[0, 255]$ , concatenado con todas las permutaciones de dos caracteres de un alfabeto,  $\Omega$ .

Posteriormente, para cada uno de los 256 valores posibles de  $w$ , se calcula el producto escalar de los subconjuntos  $\tilde{Q}_l \subseteq Q_l, \forall l \in \{1, 3, 5, 7, 9\}$ , formados por los registros que comparten el mismo valor del hash concatenado,  $w$  (Paso 7). Un proceso similar al de encajar las piezas de un rompecabezas puesto que se concatenan las diferentes subcadenas  $s[l : l + 1]$  que conformarían una supuesta secuencia,  $s$ , que comparten el valor  $h(s)$ . Finalmente, todas estas cadenas generadas se añaden al diccionario,  $\chi$  (Paso 9).

Como consecuencia de la operación producto escalar y las posibles colisiones de hashes, este algoritmo incluirá en el diccionario algunas palabras sin sentido, que nunca han sido enviadas por los usuarios. No obstante, estas palabras contaran con un valor de frecuencia,  $\tilde{f}$ , pequeño, por lo que acabarán siendo descartadas.

---

**Algoritmo 8:** Generar Diccionario (Private Sequence Fragment Puzzle)

---

**Input:** Matrices  $M$  y  $M_l$ ,  $\forall l \in \{1, 3, 5, 7, 9\}$ ,  $(\varepsilon, \varepsilon')$ ,  $(m, m')$ ,  $(k, k')$ ,  $(\mathcal{H}, \mathcal{H}')$ , n, umbral  $T$

**Output:** Diccionario  $\chi$  y frecuencias  $\tilde{f}(x)$ ,  $\forall x \in \chi$

```
1  $\chi \leftarrow \emptyset;$ 
2 for  $l \in \{1, 3, 5, 7, 9\}$  do
3    $Q_l \leftarrow \emptyset;$ 
4   Añadir a  $Q_l$  los  $T$  registros  $w||c$  con mayor valor de  $\tilde{f}_l(w||c)$ , para  $c \in \Omega^2$  y
     $w \in [256];$ 
5 for  $w \in [256]$  do
6    $Q_w \leftarrow \emptyset;$ 
7    $Q_w = \{q_1 \parallel \dots \parallel q_9 : w \parallel q_l \in Q_l \text{ para } l \in \{1, 3, 5, 7, 9\}\};$ 
8   for  $x \in Q_w$  do
9      $\chi \leftarrow \chi \cup \{x\};$ 
```

---

## 4.4. Randomized Aggregatable Privacy-Preserving Ordinal Response

El algoritmo Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR) [21], es una técnica avanzada diseñada para recolectar y analizar datos preservando la privacidad de los usuarios. Google ha utilizado este algoritmo en numerosos contextos, por ejemplo, para recopilar información sobre las configuraciones de Chrome de los usuarios (página de inicio, el motor de búsqueda y otras). Estas configuraciones son a menudo objetivo de software malicioso y se cambian sin el consentimiento de los usuarios, por lo que es fundamental conocer la distribución de estas configuraciones en un gran número de instalaciones de Chrome.

RAPPOR ofrece una solución mediante el uso de técnicas de privacidad  $\varepsilon$ -diferencial local que agregan ruido a los datos antes de ser recolectados. Esto asegura que los datos individuales no puedan ser fácilmente identificados o rastreados. Este algoritmo o técnica cuenta con dos bloques bien diferenciados, la recolección de informes y el análisis de los mismos. La etapa de recolección de informes consiste en una serie de pasos que aseguran la privacidad de los datos individuales mediante técnicas de perturbación aleatoria. Esta etapa se ejecuta en los diferentes dispositivos clientes implicados en el análisis. La etapa de análisis consiste en eliminar de forma aproximada el ruido introducido en las etapas de perturbación y estimar las frecuencias verdaderas. A continuación, se detallarán en profundidad las diferentes etapas.

El Algoritmo cliente recibe el valor real, el cual suele ser una cadena, denominado como  $v$  y los parámetros  $k$ ,  $h$ ,  $f$ ,  $p$  y  $q$ , explicados a continuación. El Algoritmo 9 muestra todo el proceso necesario para privatizar la información. En primer lugar, se crea un filtro de Bloom (Paso 1),  $B$ , de  $k$  bits y  $h$  funciones hash independientes entre sí (familia  $h$ -independiente), supongamos que estas han sido seleccionadas previamente, y se inserta el valor  $v$  en  $B$  (Paso 2). Se genera un nuevo vector de  $k$  bits,  $B'$ , de la

siguiente forma,

$$B'_i = \begin{cases} 1, & \text{con probabilidad } \frac{1}{2}f \\ 0, & \text{con probabilidad } \frac{1}{2}f \\ B_i, & \text{con probabilidad } 1 - f \end{cases} \quad (4.5)$$

donde  $f$  es un valor en el intervalo  $[0, 1]$ , introducido por el usuario. Esto corresponde con la denominada fase de perturbación permanente (Pasos 4-12), debido a que para el mismo cliente y valor  $v$ , el vector  $B'$  debe ser siempre el mismo. Por último se genera un vector de  $k$  bits adicional,  $S$ , inicializado a 0, donde la probabilidad de que cada uno de los bits de dicho vector tome un valor u otro dependerá del vector previamente definido  $B'$  y las probabilidades  $p$  y  $q$ ,

$$\text{Prob}(S_i = 1) = \begin{cases} q, & \text{si } B'_i = 1 \\ p, & \text{si } B'_i = 0 \end{cases} \quad (4.6)$$

Esta fase se denomina, fase de perturbación temporal ya que varía en cada ejecución (Pasos 13-21). El vector de bits privatizado  $S$  se envía al servidor, junto al resto de reportes privatizados. La elección de los valores de  $f$ ,  $p$  y  $q$ , es clave para determinar la utilidad de los datos de cara a su análisis. En el artículo original [21], se expone paso a paso la demostración que asegura que el algoritmo RAPPOR satisface la definición de privacidad  $\varepsilon$ -diferencial en sus dos etapas de perturbación.

En términos de privacidad  $\varepsilon$ -diferencial, el algoritmo RAPPOR satisface su definición en las etapas de perturbación permanente y temporal. Según [21], la etapa de perturbación permanente satisface una privacidad  $\varepsilon_\infty$ -diferencial, siendo:

$$\varepsilon_\infty = 2h \ln \left( \frac{1 - \frac{1}{2}f}{\frac{1}{2}f} \right). \quad (4.7)$$

Es destacable el hecho de que  $\varepsilon_\infty$  no depende del valor de  $k$ . Es cierto que un valor menor de  $k$ , o una mayor tasa de colisiones de bits en el filtro de Bloom, a veces mejora la privacidad, pero, por sí solo, no es suficiente ni necesario para proporcionar privacidad  $\varepsilon$ -diferencial. Esta etapa de perturbación introduce en el algoritmo un ruido sistemático en los datos para ocultar la información real. Este proceso de ruido asegura que, si un atacante intenta inferir información sensible a partir de los datos procesados, no pueda obtener detalles precisos sobre la entrada original. Esta etapa actúa como una capa de protección distorsiona los datos reales, haciendo que cualquier intento de analizar patrones específicos en los datos sea menos efectivo.

Durante la fase de perturbación temporal, el conocimiento de un atacante sobre un informe  $B$  debe provenir directamente de un único informe  $S$  generado al aplicar la aleatorización en dos fases, proporcionando así un mayor nivel de protección de la privacidad sobre conocimiento completo de  $B$ . Debido a la aleatorización en dos etapas, la probabilidad de observar un bit activo en un informe depende, directamente, tanto de  $q$  y  $p$  como de  $f$ . En este punto, se definen dos probabilidades clave,  $q^*$  y  $p^*$ , que derivan en la obtención del parámetro de privacidad  $\varepsilon$ . La probabilidad  $q^*$  corresponde con la probabilidad de observar en  $S$  un bit activo cuando el bit subyacente del filtro de

---

**Algoritmo 9:** Cliente (RAPPOR)

---

**Input:** Elemento  $v$ ,  $k$ ,  $f$ ,  $p$ ,  $q$  y conjunto de hashes,  $\mathcal{H}$

**Output:**  $S$

```
1  $B \leftarrow \{0\}^k;$ 
2 for  $h \in H$  do
3    $| B_{h(v)} \leftarrow 1;$ 
4 for  $i \leftarrow 0$  to  $k - 1$  do
5   Generar un número aleatorio  $r$  en el rango  $[0, 1)$ ;
6   if  $r < \frac{f}{2}$  then
7      $| B'_i \leftarrow 1;$ 
8   else
9     if  $r < f$  then
10       $| B'_i \leftarrow 0;$ 
11    else
12       $| B'_i \leftarrow B_i;$ 
13  $S \leftarrow \{0\}^k;$ 
14 for  $i \leftarrow 0$  to  $k - 1$  do
15   Generar un número aleatorio  $r$  en el rango  $[0, 1)$ ;
16   if  $B'_i = 1$  then
17     if  $r < q$  then
18        $| S_i \leftarrow 1;$ 
19   else
20     if  $r < p$  then
21        $| S_i \leftarrow 1;$ 
```

---

Bloom se encontraba establecido. Por el contrario,  $p^*$  corresponde con la probabilidad de observar un bit activo en  $S$  cuando el bit original del filtro de Bloom no estaba establecido. El cálculo de ambas probabilidades se describe a continuación.

$$q^* = P(S_i = 1 \mid B_i = 1) = \frac{1}{2}f(p + q) + (1 - f)q. \quad (4.8)$$

$$p^* = P(S_i = 1 \mid B_i = 0) = \frac{1}{2}f(p + q) + (1 - f)p. \quad (4.9)$$

A partir de los valores de estas probabilidades, se establece que la etapa de perturbación temporal o instantánea satisface una privacidad  $\varepsilon_1$ -diferencial, siendo:

$$\varepsilon_1 = h \log \left( \frac{q^*(1 - p^*)}{p^*(1 - q^*)} \right). \quad (4.10)$$

Esta etapa se enfoca en mitigar los riesgos asociados con la recopilación y análisis de datos a lo largo del tiempo, especialmente por parte de un rastreador longitudinal. Un rastreador longitudinal es un sistema que puede seguir a los usuarios durante un período prolongado y tratar de identificar patrones o comportamientos persistentes. La etapa de perturbación temporal introduce variaciones adicionales en los datos en cada instancia de respuesta, de modo que los datos individuales no puedan ser fácilmente correlacionados con la actividad de un usuario específico a lo largo del tiempo.

El objetivo del análisis de los reportes privatizados es aprender qué elementos están presentes en la población muestrada y cuáles son sus frecuencias correspondientes. Debido al uso de filtros de Bloom y la agregación de ruido aleatorio, la decodificación requiere técnicas estadísticas sofisticadas. Previamente al envío de los reportes, cada cliente se asigna aleatoriamente a uno de las  $m$  cohortes. Las cohortes son grupos de sujetos que comparten una definitoria, en este caso, cada cohorte implementa diferentes conjuntos de  $h$  funciones hash para sus filtros de Bloom, reduciendo así la posibilidad de colisiones. Por lo tanto, cada cliente debe incluir su número de cohorte en cada informe enviado al servidor.

El algoritmo de análisis es complejo y puede variar en ciertos aspectos según la implementación. Para una mayor comprensión, se ha dividido el algoritmo de análisis en diferentes etapas. En primer lugar, en el Algoritmo 10 se construye una matriz de contadores (Paso 2),  $c$ , donde cada fila corresponde a cada uno de los posibles cohortes y en cada columna  $c_{i,j}$  se almacena el número de veces que el bit  $j$  se encuentra activo en los diferentes reportes correspondientes a la cohorte  $i$  (Pasos 5, 6 y 7). Además,  $N$  almacena el número de reportes correspondientes a cada cohorte (Paso 4). A continuación, se realiza una estimación de los bits que realmente estaban activos en los filtros de Bloom originales (Pasos 8-11), a partir del conteo anterior y las probabilidades de perturbación. Por último se aplana por filas la matriz de contadores estimados, dando como resultado el vector  $Y$  (Paso 12).

La siguiente etapa del análisis consiste en construir una matriz dispersa  $X$ , de tamaño  $km \times |M|$ , donde  $M$  es el conjunto de elementos candidatos bajo consideración (Paso 1). Estos elementos candidatos forman un subconjunto del dominio de estudio y deberán ser seleccionados previamente a la ejecución del algoritmo. RAPPOR no

---

**Algoritmo 10:** Análisis RAPPOR (Vector de contadores estimados)

---

**Input:**  $D = \{\langle S_1, m_1 \rangle, \langle S_2, m_2 \rangle, \dots, \langle S_n, m_n \rangle\}$ ,  $m, k, f, p, q$

**Output:**  $Y$

```
1  $N \leftarrow \{0\}^m;$ 
2  $c \leftarrow \{0\}^{m \times k};$ 
3 for  $\langle S, m \rangle \in D$  do
4    $N_m \leftarrow N_m + 1;$ 
5   for  $j \leftarrow 0$  to  $k - 1$  do
6     if  $S_j = 1$  then
7        $c_{m,j} \leftarrow c_{m,j} + 1;$ 
8    $t \leftarrow \{0\}^{m \times k};$ 
9   for  $i \leftarrow 0$  to  $m - 1$  do
10    for  $j \leftarrow 0$  to  $k - 1$  do
11       $t_{ij} \leftarrow \frac{c_{ij} - (p + \frac{1}{2}f(q - \frac{1}{2}fp))N_j}{(1-f)(q-p)};$ 
12  $Y \leftarrow [t_{11}, t_{12}, \dots, t_{1k}, t_{21}, t_{22}, \dots, t_{2k}, \dots, t_{m1}, t_{m2}, \dots, t_{mk}]^T$ 
```

---

ha sido diseñado para trabajar sobre un dominio desconocido, como podía ser el caso en 4.3. El Algoritmo 11 describe el proceso de construcción de la matriz  $X$ , por cada candidato posible se utilizan  $m$  filtros de Bloom, utilizando las mismas funciones hash empleadas en el cliente. Cada punto corresponde a un bit en una cohorte. Una celda en esa fila toma el valor 1, si dicho bit tomase el valor 1 al aplicar la función hash correspondiente, sobre dicho elemento candidato (Pasos 3-7). Por último, se traspone la matriz  $X$  para cumplir con las dimensiones definidas (Paso 8).

---

**Algoritmo 11:** Análisis RAPPOR (Matriz de elementos candidatos)

---

**Input:**  $M, m, k, \tilde{H} = \{H_0, H_1, \dots, H_{m-1}\}$

**Output:**  $X$

```
1  $X \leftarrow \{0\}^{|M| \times km};$ 
2  $i \leftarrow 0;$ 
3 for  $v \in M$  do
4   for  $c \leftarrow 0$  to  $m - 1$  do
5     for  $h \in \tilde{H}_c$  do
6        $X_{i,h(v)+k \cdot m} \leftarrow 1;$ 
7    $i \leftarrow i + 1;$ 
8  $X \leftarrow X^T;$ 
```

---

Finalmente, según el artículo original, se ajusta un modelo de regresión Lasso,  $Y \sim X$ , que actúa como un filtro para continuar con los elementos candidatos que tienen coeficientes no nulos, para posteriormente ajustar una regresión lineal de mínimos cuadrados ordinarios con las variables seleccionadas. Sin embargo, Google realiza esto cuando el número de candidatos es alto en comparación al número de cohortes y funciones hash empleadas [22], lo cual puede ser difícil de determinar. Esto ha sido

despreciado por diferentes implementaciones, que han optado por realizar una única regresión Lasso.

En esta etapa final del análisis de los reportes, se busca determinar aquellos elementos candidatos que realmente aparecieron utilizando la información de qué bits tomarían el valor 1 para cada candidato, información contenida en la matriz  $X$ . Utilizando los conteos estimados,  $Y$ , en lugar de los observados, bastaría con resolver un sistema de ecuaciones lineales para encontrar la frecuencia de cada candidato. Es por esto, que se utiliza una regresión lineal. Los coeficientes del modelo de regresión entrenado indican la frecuencia de aparición estimada de cada uno de los candidatos.

Respecto a la elección de los valores de los parámetros, no existe ninguna regla bien definida. En el artículo original de Google [21], se presentan una serie de experimentos diferentes, en los que varían la naturaleza de los datos y la elección de los parámetros. Por ejemplo, se realizó un experimento para medir la precisión frente a una distribución normal con los parámetros,  $q = 0,75$ ,  $p = 0,5$ ,  $\epsilon = \ln(3)$  y  $f = 0$ , para 10000 registros. Este experimento se asemeja a uno de los realizados en la sección 5.5.3.

## 4.5. Mecanismo dBitFlip para la estimación de histogramas

Como se ha podido ver a lo largo de todo el documento, la recopilación y el análisis de datos de telemetría de los dispositivos de los usuarios es una práctica común realizada por muchas empresas punteras. Pese a que esta recopilación se lleva a cabo con el fin de mejorar la experiencia del usuario, puede plantear riesgos significativos para la privacidad de los usuarios. Los algoritmos de privacidad  $\epsilon$ -diferencial local que han surgido recientemente proporcionan garantías de privacidad muy fuertes para una única ronda de recopilación, pero se degradan rápidamente cuando la telemetría se recopila regularmente. Esto supone un gran problema para aquellas empresas que desean realizar una recopilación repetida, como las estadísticas de uso diario de una aplicación.

Con el objetivo de solventar este problema, el equipo de investigadores de Microsoft ha propuesto [16] una serie de mecanismos que implementan privacidad  $\epsilon$ -diferencial local para la obtención de histogramas. Estos mecanismos proporcionan garantías formales de privacidad para la recopilación continuada de datos.

Antes de detallar el algoritmo, es necesario definir el contexto de estudio. Supongamos que existen un número  $n$  de usuarios, y cada usuario en el tiempo  $t$  tiene un contador o registro privado (de valor entero o real) con valor  $x_i \in [0; m]$ . Esto podría extrapolarse a un caso puntual, sin necesidad de ser un registro continuo en el tiempo, como podría ser la visita al médico de un paciente o una encuesta sobre intención de voto.

Ajeno a cualquier cliente, existe un servidor que actuará como recolector de datos para recopilar los valores de los contadores  $\{x_i\}_{i \in [n]}$  en un instante determinado para realizar análisis estadísticos. Por ejemplo, para la obtención de un histograma de

frecuencias que permita analizar los elementos más frecuentes por los usuarios, junto con la distribución de los mismos, en multitud de contextos.

En el Algoritmo del cliente, 12, un usuario  $i$  selecciona aleatoriamente  $d$  elementos sin reemplazo de  $[k]$ , denotados por  $j_1, j_2, \dots, j_d$  (Pasos 2-5). Estos elementos son independientes del valor correspondiente,  $x_i$ .

Cuando los usuarios se disponen a enviar los datos al servidor, cada uno envía un vector,  $b_i = [(j_1, b_{i,j_1}), (j_2, b_{i,j_2}), \dots, (j_d, b_{i,j_d})]$  (Paso 12), donde  $b_{i,j_p}$  es un bit aleatorio 0-1, con la probabilidad:

$$\Pr(b_{i,j_p} = 1) = \begin{cases} \frac{e^{\epsilon/2}}{e^{\epsilon/2} + 1} & \text{si } x_i = j_p \\ \frac{1}{e^{\epsilon/2} + 1} & \text{si } x_i \neq j_p \end{cases} \quad (4.11)$$

para  $p = 1, 2, \dots, d$  (Pasos 6-11).

---

**Algoritmo 12:** Cliente dBitFlip

---

**Input:** Valor del contador  $x_i$  del usuario  $i$ , dominio  $[k]$ ,  $\epsilon$ ,  $d$   
**Output:** Vector privatizado  $b_i$

```

1  $S \leftarrow \emptyset;$ 
2 while  $|S| < d$  do
3   Seleccionar aleatoriamente  $j \in [k];$ 
4   if  $j \notin S$  then
5      $S \leftarrow S \cup \{j\};$ 
6 for  $p \leftarrow 1$  to  $d$  do
7    $b_{i,j_p} \leftarrow 0;$ 
8   if  $x_i = j_p$  then
9      $b_{i,j_p} \leftarrow 1$  con probabilidad  $\frac{e^{\epsilon/2}}{e^{\epsilon/2} + 1};$ 
10  else
11     $b_{i,j_p}(t) \leftarrow 1$  con probabilidad  $\frac{1}{e^{\epsilon/2} + 1};$ 
12  $b_i \leftarrow [(j_1, b_{i,j_1}), (j_2, b_{i,j_2}), \dots, (j_d, b_{i,j_d})];$ 

```

---

El servidor recibe los diferentes vectores  $b_i$  privatizados y estima el histograma  $h$  para cada elemento  $v \in [k]$ , de la forma:

$$\hat{h}(v) = \frac{k}{nd} \sum_{\substack{b_{i,v} \\ \text{recibidos}}} \frac{b_{i,v} \cdot (e^{\epsilon/2} + 1) - 1}{e^{\epsilon/2} - 1} \quad (4.12)$$

En términos de precisión, la intuición es que para cada  $v \in [k]$ , hay aproximadamente  $nd/k$  usuarios respondiendo con un bit,  $b_{i,v}$ , siendo  $n$  el número total de usuarios. Para una recopilación de datos de una sola ronda por usuario, el mecanismo dBitFlip garantiza la privacidad  $\epsilon$ -diferencial para cada usuario. Al recibir los  $d$  bits  $f_{i,j_p}$  de cada usuario  $i$ , el servidor puede entonces estimar el histograma  $\hat{h}_t$ . Con una probabilidad de al menos  $1 - \delta$ , según [16] se puede afirmar que:

$$\max_{v \in [k]} |h(v) - \hat{h}(v)| \leq \sqrt{\frac{5k}{nd}} \cdot \frac{e^{\varepsilon/2} + 1}{e^{\varepsilon/2} - 1} \cdot \sqrt{\log \frac{6k}{\delta}} \leq O\left(\sqrt{\frac{k \log(k/\delta)}{\varepsilon^2 nd}}\right). \quad (4.13)$$

Cuando la situación requiere una recolección continuada de datos, donde se están recopilando datos de contadores de usuarios repetidamente, la memoización puede ser útil para evitar la redundancia en el procesamiento de los datos. La memoización es una técnica de optimización, usada en programación, donde se almacenan los resultados de cálculos previos para evitar recalcularlos cuando se necesita el mismo resultado nuevamente. Por ejemplo, si un usuario mantiene su valor de contador privado constante durante varios días seguidos, la memoización permite al usuario precalcular y almacenar sus respuestas para todos los posibles valores del contador,  $x_i$ . Entonces, en lugar de tener que recalcular y enviar la respuesta cada día, el usuario simplemente utiliza la respuesta memoizada correspondiente al valor actual del contador.

Esta práctica puede ser especialmente beneficiosa en términos de eficiencia computacional, ya que reduce la carga de cálculo y transmisión de datos repetitivos. Sin embargo, en el contexto de la privacidad, la memoización puede provocar fugas de privacidad si no se maneja adecuadamente. Para evitar tales fugas, se almacenan las posibles respuestas basándonos en el algoritmo descrito anteriormente. Cada usuario ejecuta el Algoritmo 12 para cada  $v \in [k]$ , y almacena la respuesta en un diccionario  $f_d : [k] \rightarrow \{0, 1\}^d$ . Un usuario siempre enviará  $f_d(v)$  si el valor de su contador es  $v$ . De esta forma, varios elementos se mapean al mismo resultado, lo que ayuda a evitar las fugas de privacidad. Esta protección se vuelve más fuerte cuando  $d = 1$ . Respecto a valores reales propuestos por Microsoft para los parámetros  $\varepsilon$  y  $d$ , en el artículo original [16], se realizan experimentos variando notablemente el valor de estos parámetros. Para  $d$ , se evaluó el rendimiento con los valores  $\{1, 2, 4, k\}$ . Por otro lado, se utilizaron valores de  $\varepsilon \leq 2$ .

# 5. Evaluación y análisis de las técnicas implementadas

---

A lo largo del capítulo anterior, se han presentado y explicado cinco algoritmos dedicados a la estimación de frecuencias sobre un flujo de datos masivo. En concreto, *Private Count Mean Sketch*, *Private Hadamard Count Mean Sketch*, *Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR)* y *dBitFlip* tienen como objetivo estimar las frecuencias de una serie de elementos pertenecientes a un dominio conocido, similar a la generación de un histograma de estimaciones. Por otro lado, el algoritmo *Private Sequence Fragment Puzzle* tiene la capacidad de generar el conjunto de elementos más frecuentes en un flujo de datos, sin conocer el dominio al que pertenecen dichos elementos.

En este capítulo se han implementado dichas técnicas con la finalidad de presentar un análisis exhaustivo de las mismas. Se presentarán los diferentes datasets generados para evaluar el rendimiento de las técnicas implementadas. Se evaluarán los algoritmos mediante la ejecución de diferentes experimentos, en términos de precisión, eficiencia y complejidad, proporcionando una visión integral de su efectividad y aplicabilidad.

## 5.1. Datasets empleados para el análisis

Para evaluar el rendimiento de los algoritmos implementados, se han generado varios conjuntos de datos sintéticos que simulan diferentes escenarios y condiciones.

Se creó un dataset de palabras que, en castellano, se consideran anglicismos para simular un flujo de usuarios en una red de mensajería. Este dataset incluye términos seleccionados como anglicismos comunes, con aproximadamente un 10 % del dataset compuesto por cadenas de caracteres aleatorios que representan errores tipográficos. Estos datos se utilizarán específicamente para evaluar el algoritmo *Sequence Fragment Puzzle*. Los nombres de estos datasets vienen determinados por su temática *anglicismo*, seguidos de su tamaño. Se han preparado versiones del dataset con 750 (**anglicismo\_750**), 2500 (**anglicismo\_2500**), 50000 (**anglicismo\_50k**) y 1 millón (**anglicismo\_1M**) de registros.

La Figura 5.1 muestra las diferentes cadenas o elementos presentes en el dataset y sus frecuencias reales (para la versión de 50000 registros). La categoría *Otros* hace referencia al ruido añadido al dataset con cadenas generadas de forma aleatoria con una probabilidad reducida.

Con el objetivo de evaluar el resto de métodos, se han generado datasets relacionados con distribuciones de números aleatorios enteros. Esto incluye una distribución exponencial y una distribución normal con media 12 y desviación estándar 2. Estos datasets han sido diseñados con diferentes números de registros, para evaluar cómo los algoritmos manejan diferentes escalas de datos y distribuciones. Los nombres de

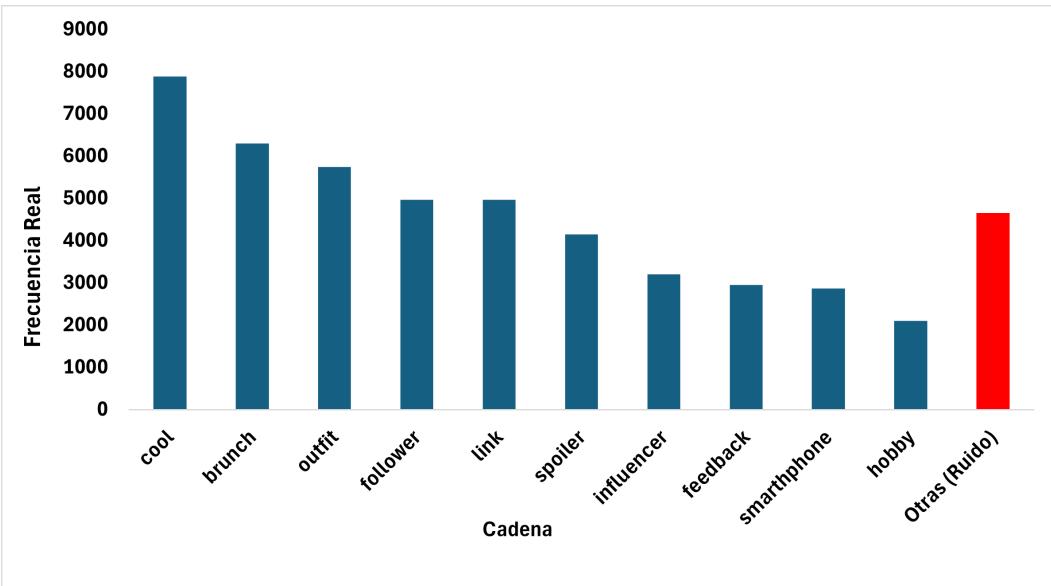


Figura 5.1: Estructura interna del dataset de palabras consideradas anglicismos.

los diferentes datasets recogen información acerca del tipo de distribución y el número de registros de cada uno, siendo **exp\_distrib\_50k** y **exp\_distrib\_500k** para los datasets correspondientes a las distribuciones exponenciales de tamaño 50000 y 500000, respectivamente. Por otro lado, se ha generado una serie de datasets referentes a la distribución normal mencionada. Dichos datasets reciben los nombres **norm\_distrib\_50**, **norm\_distrib\_300**, **norm\_distrib\_750**, **norm\_distrib\_900**, **norm\_distrib\_2500**, **norm\_distrib\_15k**, **norm\_distrib\_60k**, **norm\_distrib\_200k**, **norm\_distrib\_350k**, **norm\_distrib\_1M**, y **norm\_distrib\_2.4M**. Completando una colección de datasets de tamaño altamente variado, entre 50 y 2,4 millones de registros.

De forma similar al caso anterior, las Figuras 5.2 y 5.3 muestran gráficamente los elementos del dominio, así como su proporción, en ambos datasets generados de forma sintética. Para estos casos se ha optado por datasets numéricos por su sencillez a la hora de generarlos sintéticamente, así como su mejor visualización frente a otros tipos de datos. No obstante, todos los algoritmos implementados son capaces de soportar cualquier tipo de datos.

Todos estos datasets sintéticos han sido generados utilizando scripts en Python. Los archivos están en formato CSV y cada registro inicialmente incluye un identificador único de 10 caracteres alfanuméricos que representa el usuario que envía el elemento. Este identificador se elimina antes de la ejecución del algoritmo para enfocarse únicamente en los datos relevantes. Se podrá ejecutar cualquiera de los algoritmos descrito con un dataset diferente, siempre que este cumpla con una estructura similar a los presentados en esta sección. Para ello, se recomienda revisar el código fuente referente a la creación de los dataset sintéticos.

Esta variedad de datasets permite una evaluación completa y detallada de los algoritmos implementados, cubriendo desde datos textuales con errores potenciales hasta datos numéricos con distribuciones variadas. La generación sintética asegura un control preciso sobre las condiciones de prueba, proporcionando resultados significativos

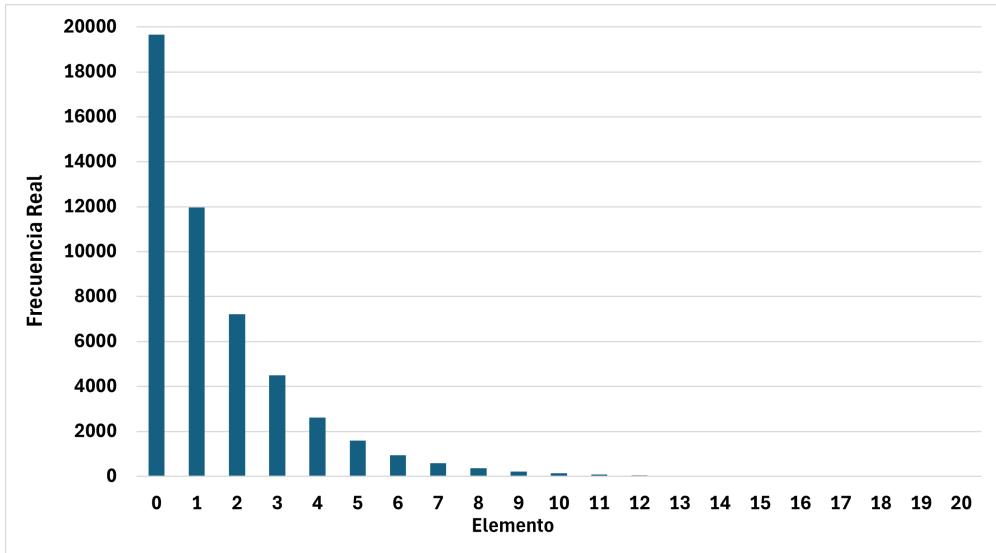


Figura 5.2: Dataset de números enteros generados aleatoriamente a partir de una distribución exponencial.

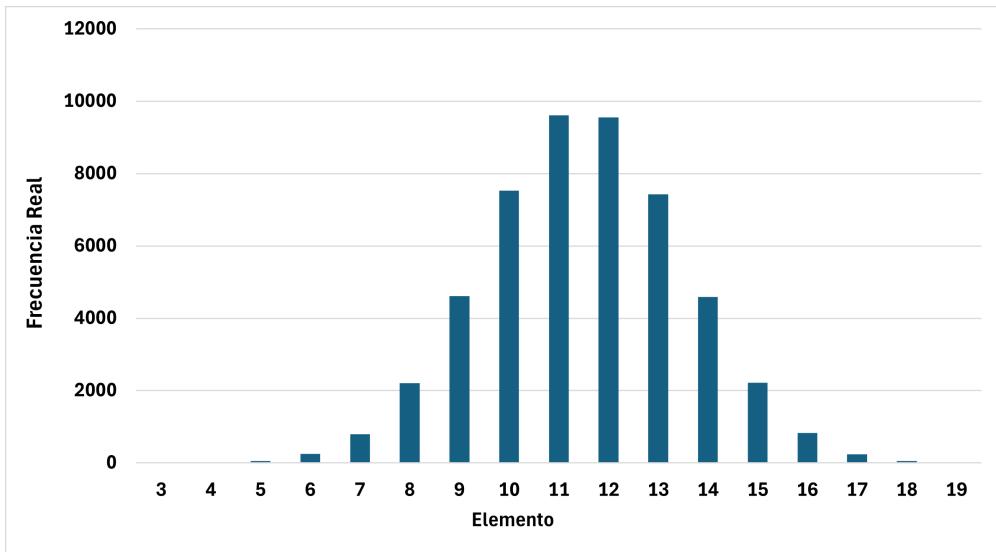


Figura 5.3: Dataset de números enteros generados aleatoriamente a partir de una distribución normal ( $\mu = 12, \sigma = 2$ ).

sobre el rendimiento y la robustez de los algoritmos en diferentes escenarios simulados.

## 5.2. Implementación de los métodos

En esta sección se detalla la implementación práctica de los algoritmos mencionados anteriormente, incluyendo una breve descripción de cada uno, el entorno de desarrollo utilizado y las especificaciones del hardware. A continuación se presenta una breve descripción de cada uno de los algoritmos implementados y sus propiedades fundamentales, a modo de recordatorio:

- **Private Count Mean Sketch:** Algoritmo diseñado para estimar frecuencias de elementos en un flujo de datos mientras se preserva la privacidad  $\varepsilon$ -diferencial. Utiliza estructuras de datos probabilísticas propias del sketching, combinadas con la adición de ruido para asegurar la privacidad de los datos individuales.
- **Private Hadamard Count Mean Sketch:** Algoritmo que busca mejorar el enfoque del *Count Mean Sketch* aplicando transformaciones Hadamard para reducir el ancho de banda necesario para la transmisión de información entre el cliente y servidor, mientras se mantiene la calidad de la información transmitida.
- **RAPPOR:** Técnica propuesta por la empresa Google que permite la recolección y agregación de datos sensibles mientras preserva la privacidad de los individuos. Es particularmente útil para estimar distribuciones de datos, así como detectar tendencias, en grandes poblaciones.
- **dBitFlip:** Algoritmo que emplea sucesivas técnicas de aleatorización a nivel de bit para preservar la privacidad mientras posibilita la estimación de frecuencias de elementos en un dominio conocido. Es un método altamente eficiente en términos de comunicación y procesamiento. Sin embargo, sus garantías en términos de privacidad son, a priori, peores respecto al resto de métodos.
- **Private Sequence Fragment Puzzle:** Algoritmo que se diferencia de los anteriores al no requerir conocimiento previo del dominio de los elementos. Este algoritmo se enfoca en identificar un conjunto de elementos repetidos en numerosas ocasiones en un flujo de datos, a la vez que se mantiene la privacidad de la información enviada por los usuarios. Es una técnica ideal para situaciones en las que el dominio de estudio sea desconocido o intratable.

Cada método se ha implementado de forma independiente entre sí, siguiendo el pseudocódigo descrito en el capítulo anterior, adaptándolo para trabajar con los diferentes tipos de datasets generados y su correspondiente invocación desde la terminal. Todos los métodos han sido implementados en el lenguaje de programación Python debido a su versatilidad y las librerías disponibles para cálculos numéricos y manipulación de datos. Se utilizaron las siguientes librerías principales:

- **NumPy y Random:** Para operaciones numéricas eficientes y manejo de matrices, esencial en los data sketches. Así como, herramientas para generar números aleatorios, necesarios en la mayoría de métodos.

- **Pandas** y **csv**: Para manipulación y análisis de datos estructurados. Así como, la lectura de datos desde archivos CSV y escritura en este formato.
- **scikit-learn**: Para el uso de modelos de regresión en el método RAPPOR y extracción de métricas de evaluación para el resto de métodos.
- **matplotlib**, **Tabulate** y **Time**: Para crear multitud de gráficas para visualizar los resultados del análisis de forma clara y comprensible. Para la impresión de tablas en formato de texto tabulado para una mayor legibilidad en la consola. Así como, realizar las diferentes mediciones de tiempo correspondientes.

Todo el código desarrollado junto con los datasets utilizados, y las instrucciones para ejecutar los algoritmos puede encontrarse en el repositorio de GitHub bajo el siguiente enlace: <https://github.com/Antonio-Requena/DP-Sketching-Algorithms>. En el repositorio se incluye una guía detallada de instalación, configuración del entorno y cómo ejecutar cada uno de los algoritmos implementados. De esta forma asegura la reproducibilidad y transparencia de los experimentos realizados, permitiendo a otros investigadores revisar y validar los resultados obtenidos.

### 5.3. Especificaciones del entorno de ejecución

Las diferentes técnicas han sido desarrolladas y ejecutados en un equipo MacBook Air (M1, 2020). Este modelo está equipado con el chip Apple M1, que incluye una CPU de 8 núcleos. Estos núcleos están divididos en 4 núcleos de alto rendimiento y 4 núcleos de eficiencia, lo que permite un equilibrio óptimo entre poder de procesamiento y eficiencia energética. Los núcleos de alto rendimiento están diseñados para manejar tareas intensivas y demandantes, proporcionando una ejecución rápida y fluida de aplicaciones complejas y algoritmos avanzados. Por otro lado, los núcleos de eficiencia se encargan de tareas menos exigentes, ayudando a conservar la energía y prolongar la duración de la batería.

Dicho equipo cuenta con 16 GB de memoria RAM unificada, lo cual facilita el manejo eficiente de grandes volúmenes de datos. En cuanto al almacenamiento, el dispositivo está equipado con un SSD de alta velocidad. Este tipo de almacenamiento no solo ofrece grandes capacidades de almacenamiento, sino que también asegura tiempos de acceso extremadamente rápidos y una alta velocidad de lectura y escritura de datos. Esto es especialmente beneficioso para el desarrollo de las diferentes técnicas, ya que reduce significativamente los tiempos de carga.

El sistema operativo macOS Sonoma (14.5) proporciona un entorno estable y optimizado para el desarrollo y ejecución de software. No obstante, el entorno de desarrollo integrado (IDE) utilizado, ha sido Visual Studio Code.

## 5.4. Métricas de Evaluación

Para evaluar el rendimiento y la precisión de los métodos implementados, se utilizarán diversas métricas que se describen a continuación. Estas métricas se dividen en dos categorías generales, por un lado se medirán los tiempos de ejecución de las diferentes técnicas. Por otro lado, se presentan diferentes medidas de error en la estimación de frecuencias propuestas, para medir la utilidad y eficiencia de los métodos.

Los tiempos de ejecución se medirán en dos partes diferenciadas:

- **Tiempos de ejecución del cliente:** Se medirán los tiempos que tarda la parte cliente en ejecutar los métodos correspondientes. Esto incluye la recolección y preprocesamiento de datos antes de enviarlos al servidor.
- **Tiempos de ejecución del servidor:** Se medirán los tiempos de ejecución de las operaciones que se realizan en el servidor, cuando estas se puedan separar claramente de las operaciones del cliente. Esto incluye el procesamiento de los datos recibidos y la ejecución de algoritmos de estimación y análisis.

Para evaluar la precisión en la estimación de frecuencias, se utilizarán las siguientes métricas de error. En estas métricas,  $n$  representa el número de registros en el dataset y  $m$  el tamaño del dominio, es decir, el número de elementos cuya frecuencia se estimará. Los valores reales de frecuencia se denotan como  $f_i$  y las frecuencias estimadas como  $\hat{f}_i$ .

- **Número de errores en la estimación:** Se define como el sumatorio de las diferencias absolutas entre el valor real ( $f_i$ ) y el valor estimado ( $\hat{f}_i$ ) para cada elemento del dominio:

$$\text{Número de errores} = \sum_{i=1}^m |f_i - \hat{f}_i|$$

- **Media del número de errores:** Es el promedio de los errores de estimación, calculado como:

$$\text{Media de errores} = \frac{1}{m} \sum_{i=1}^m |f_i - \hat{f}_i|$$

- **Error porcentual:** Es la media del número de errores expresada como porcentaje del número total de registros procesados:

$$\text{Error porcentual} = \left( \frac{\text{Media de errores}}{n} \right) \times 100$$

- **Error Cuadrático Medio (MSE):** Se calcula como la media de los cuadrados de las diferencias entre los valores reales y los estimados:

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^m (f_i - \hat{f}_i)^2$$

- **Raíz del Error Cuadrático Medio (RMSE):** Es la raíz cuadrada del MSE, proporcionando una medida de error en las mismas unidades que los datos originales:

$$\text{RMSE} = \sqrt{\frac{1}{m} \sum_{i=1}^m (f_i - \hat{f}_i)^2}$$

La métrica MSE se caracteriza por penalizar aquellas predicciones que tienen un valor muy dispar con el valor esperado, pudiendo generar una visión distorsionada de la utilidad de los métodos. Por ello, RMSE se utiliza para mitigar la existencia de errores muy elevados, obteniendo un resultado más interpretable.

- **MSE y RMSE normalizados:** Estas métricas se normalizan dividiendo por el rango de los valores reales ( $\max(f_i) - \min(f_i)$ ), proporcionando una medida de error relativa:

$$\text{MSE normalizado} = \frac{\text{MSE}}{\max(f_i) - \min(f_i)}$$

$$\text{RMSE normalizado} = \frac{\text{RMSE}}{\max(f_i) - \min(f_i)}$$

- **Coeficiente de correlación de Pearson (r):** Mide la relación lineal entre las frecuencias reales y las estimadas, pero no proporciona información sobre el tamaño del error. Un valor alto de  $r$  indica que las frecuencias estimadas tienden a seguir la misma tendencia que las frecuencias reales, pero no garantiza una precisión absoluta en las estimaciones. Puede considerarse un indicador muy útil para un estudio de este tipo ya que logra una evaluación más general.

$$r = \frac{\sum_{i=1}^m (f_i - \bar{f})(\hat{f}_i - \bar{\hat{f}})}{\sqrt{\sum_{i=1}^m (f_i - \bar{f})^2 \sum_{i=1}^m (\hat{f}_i - \bar{\hat{f}})^2}}$$

donde  $\bar{f}$  y  $\bar{\hat{f}}$  son las medias de los valores reales y estimados, respectivamente.

#### 5.4.1. Inconvenientes respecto a la medición del error

Es importante considerar algunas problemáticas que pueden surgir al utilizar las métricas de error anteriormente descritas, especialmente en contextos donde se trata con elementos del dominio que tienen frecuencias notablemente altas.

En muchos casos, la distribución de frecuencias de los elementos del dominio puede ser muy desigual, con algunos elementos presentando frecuencias muy altas mientras que otros tienen frecuencias mucho menores. Esta desigualdad puede causar varios problemas:

- **Minimización del error relativo en elementos con alta Frecuencia:** Para elementos con frecuencias muy altas, un error absoluto puede parecer pequeño en comparación con su valor real. Por ejemplo, un error de 100 en una frecuencia real de 10,000 representa solo un 1% de error relativo. Sin embargo, este mismo error absoluto podría ser significativo para elementos con frecuencias más bajas.

- **Impacto desproporcionado en las métricas de error:** Las métricas como el Error Cuadrático Medio (MSE) y la Raíz del Error Cuadrático Medio (RMSE) pueden estar fuertemente influenciadas por errores en elementos con altas frecuencias, debido a que estos errores tienen un mayor peso en el cálculo de los promedios cuadrados. Esto puede llevar a una subestimación del error en elementos con frecuencias bajas.
- **Dificultad en la interpretación del error promedio:** La media de los errores y el error porcentual pueden ser difíciles de interpretar en distribuciones con alta varianza en las frecuencias, ya que estos valores no reflejan necesariamente el impacto del error en elementos con diferentes frecuencias.

Por otro lado, las métricas como el MSE y el RMSE son muy sensibles a los *outliers*, es decir, a los elementos con errores de estimación extremadamente grandes. Estos *outliers* pueden distorsionar la percepción del rendimiento del modelo, haciendo que parezca menos preciso de lo que realmente es para la mayoría de los elementos. Puede ocurrir que los elementos con frecuencias más bajas tengan errores relativos muy altos y distorsionen la eficacia real de las técnicas.

Para mitigar algunas de estas problemáticas, se pueden utilizar versiones normalizadas de las métricas de error. La normalización por el rango de los valores reales, como en el MSE y RMSE normalizados, puede ayudar a proporcionar una medida de error relativa más equilibrada. Sin embargo, incluso estas versiones normalizadas tienen limitaciones y deben interpretarse con cuidado. Por ello, se ha apostado por un enfoque combinado que utilice varias métricas y tenga en cuenta la distribución de los datos.

## 5.5. Experimentos

En esta sección, se presentan los resultados obtenidos a partir de una serie de experimentos diseñados para evaluar y analizar el rendimiento de las cinco técnicas descritas a lo largo del documento. Con el objetivo de analizar de manera empírica la eficiencia, efectividad y robustez de cada algoritmo bajo diferentes configuraciones de parámetros y datasets. Se trata de un total de cuatro experimentos, uno por cada algoritmo. A excepción de los algoritmos *Private Count Mean Sketch* y *Private Hadamard Count Mean Sketch*, que debido a la naturaleza de sus parámetros, se evaluarán de forma conjunta, a modo de comparativa. El resto de experimentos analizan, de manera aislada, el resto de algoritmos presentados. Pese a no ser del todo justo llevar a cabo una comparativa de los resultados obtenidos entre experimentos diferentes, se mantendrá un contexto y propósito generalizado entre todos estos.

Para garantizar una evaluación justa y exhaustiva, se han definido métricas específicas que permiten medir aspectos clave como el tiempo de ejecución, separando por operaciones en el cliente o en el servidor, la precisión, y la capacidad de representar la naturaleza de los datos reales. Durante los diferentes experimentos, se han empleado diferente datasets, los cuales han sido descritos en 5.1. A lo largo de esta sección, se detallarán los procedimientos experimentales realizados, incluyendo las configuraciones de parámetros y datasets escogidos. Se presentarán los resultados obtenidos y se

realizará un análisis crítico de estos. Este análisis servirá para identificar las fortalezas y debilidades de cada algoritmo, proporcionando una base fundamentada para su posterior aplicabilidad en contextos prácticos.

### 5.5.1. Experimento 1. Comparativa entre los algoritmos Private Count Mean Sketch y Private Hadamard Count Mean Sketch

Este experimento tiene por objetivo realizar un análisis comparativo exhaustivo entre dos de los algoritmos de privacidad de datos descritos a lo largo de la memoria, Private Count Mean Sketch (PCMS) y Private Hadamard Count Mean Sketch (PHCMS). Al utilizar los mismos parámetros, ambos algoritmos operan bajo las mismas condiciones, lo que permite una comparación más justa y directa. Los resultados obtenidos reflejan diferencias en la metodología del algoritmo en lugar de diferencias en los parámetros. Esta ventaja ocurre únicamente en estos dos algoritmos.

Entre los datasets empleados en este experimento destaca uno que representa una distribución exponencial de un tamaño de 50000 registros, **exp\_distrib\_50k**. Adicionalmente a los introducidos en la sección 5.1, se ha empleado un dataset correspondiente a una distribución normal ( $\mu = 12, \sigma = 2$ ) de 200000 registros, **norm\_distrib\_200k**, para evaluar el impacto del parámetro  $\varepsilon$  en los resultados. Además, se ha utilizado una colección de datasets de características similares al anterior, pero de tamaño variable entre 50 y 2 millones de registros, con el objetivo de analizar el rendimiento de ambas técnicas en función del tamaño de la entrada. Dichos datasets corresponden con **norm\_distrib\_50**, **norm\_distrib\_300**, **norm\_distrib\_750**, **norm\_distrib\_900**, **norm\_distrib\_2500**, **norm\_distrib\_15k**, **norm\_distrib\_60k**, **norm\_distrib\_350k**, **norm\_distrib\_1M**, y **norm\_distrib\_2\_4M**.

Las Tablas 5.1 y 5.3 muestran los resultados obtenidos, en términos de coste temporal, de la ejecución de ambas técnicas. De la misma forma, las Tablas 5.2 y 5.4 recogen una serie de métricas clave para evaluar la precisión de ambos algoritmos. Se han establecido unos parámetros fijos de  $\varepsilon = 2$  y  $N$ (Tamaño del dataset) = 50000 registros. Los parámetros que varían de una ejecución a otra son  $k$  y  $m$ .  $k$  corresponde con el número de funciones hash empleadas y tomará los valores  $[16, 128, 2^{10}, 2^{15}, 2^{16}]$ . El parámetro  $m$  determina el rango de valores posibles devueltos por las funciones hash y toma los valores  $[16, 64, 256, 2^{10}]$ . Respecto al análisis del coste temporal se han obtenido los tiempos de ejecución de la parte del cliente, encargada de privatizar el valor real  $d$ , el tiempo de ejecución asociado a la actualización de la matriz de sketch y el tiempo necesario para realizar la estimación de un elemento.

En ambos algoritmos el tiempo de ejecución del cliente depende directamente del valor del parámetro  $m$ , un resultado que concuerda con la implementación, ya que en ambos se genera un vector privatizado de dicho tamaño. Ambos mecanismos obtienen tiempos similares para valores pequeños de  $m$ . A medida que  $m$  aumenta, el tiempo de ejecución del cliente en PHCMS aumenta notablemente. Esto es debido a que en dicho algoritmo se realiza una multiplicación matricial a partir de la matriz de Hadamard, cuya dimensión es de  $m \times m$ . En cuanto al tiempo de actualización la matriz de sketch,

PARÁMETROS		COSTE TEMPORAL		
k	m	Cliente	Servidor (Actualizar matriz)	Servidor (Estimación)
16	16	0.0100 ms	0.0114 ms	0.0250 ms
16	64	0.0103 ms	0.0219 ms	0.0228 ms
16	256	0.0139 ms	0.0670 ms	0.0229 ms
16	1024	0.0308 ms	0.2828 ms	0.0251 ms
128	16	0.0094 ms	0.0107 ms	0.1651 ms
128	64	0.0104 ms	0.0219 ms	0.1726 ms
128	256	0.0138 ms	0.0668 ms	0.1715 ms
128	1024	0.0265 ms	0.2503 ms	0.1754 ms
1024	16	0.0095 ms	0.0107 ms	1.3214 ms
1024	64	0.0102 ms	0.0217 ms	1.3175 ms
1024	256	0.0139 ms	0.0666 ms	1.3611 ms
1024	1024	0.0269 ms	0.2522 ms	1.4236 ms
32768	16	0.0099 ms	0.0108 ms	42.8604 ms
32768	64	0.0107 ms	0.0220 ms	44.5811 ms
32768	256	0.0144 ms	0.0673 ms	46.4648 ms
32768	1024	0.0277 ms	0.2579 ms	46.4400 ms
65536	16	0.0096 ms	0.0105 ms	82.6702 ms
65536	64	0.0102 ms	0.0207 ms	84.6401 ms
65536	256	0.0135 ms	0.0621 ms	84.5558 ms
65536	1024	0.0247 ms	0.2270 ms	83.7617 ms

Tabla 5.1

Coste temporal asociado a la ejecución del algoritmo Private Count Mean Sketch, en función de los parámetros m y k. (dataset: exp\_distrib\_50k y  $\varepsilon = 2$ )

PARÁMETROS		MÉTRICAS DE ERROR						
k	m	Media Errores	Error porcentual	MSE	RMSE	MSE (Normalizado)	RMSE (Normalizado)	C.C. Pearson
16	16	953.30	1.91 %	1829894.93	1352.74	93.12	9.65	0.6985
16	64	560.87	1.12 %	388747.16	623.50	19.78	4.45	0.7154
16	256	322.07	0.64 %	370550.66	608.73	18.86	4.34	0.7049
16	1024	243.65	0.49 %	116354.10	341.11	5.92	2.43	0.7138
128	16	471.91	0.94 %	313140.62	559.59	15.94	3.99	0.7192
128	64	213.29	0.43 %	66383.90	257.65	3.38	1.84	0.7104
128	256	273.42	0.55 %	100291.45	316.69	5.10	2.26	0.7049
128	1024	202.29	0.40 %	51674.57	227.32	2.63	1.62	0.7149
1024	16	277.43	0.55 %	117633.44	342.98	5.99	2.45	0.6914
1024	64	149.47	0.30 %	30064.45	173.39	1.53	1.24	0.7119
1024	256	167.89	0.34 %	35351.40	188.02	1.80	1.34	0.7222
1024	1024	167.27	0.33 %	39297.22	198.24	2.00	1.41	0.7133
32768	16	204.12	0.41 %	70043.14	264.66	3.56	1.89	0.7145
32768	64	216.90	0.43 %	67969.30	260.71	3.46	1.86	0.7170
32768	256	212.49	0.42 %	73979.55	271.99	3.76	1.94	0.7072
32768	1024	203.12	0.41 %	51167.91	226.20	2.60	1.61	0.7143
65536	16	248.13	0.50 %	94997.33	308.22	4.83	2.20	0.7360
65536	64	196.26	0.39 %	53686.37	231.70	2.73	1.65	0.7089
65536	256	168.17	0.34 %	44235.42	210.32	2.25	1.50	0.7168
65536	1024	194.14	0.39 %	59239.17	243.39	3.01	1.74	0.7166

Tabla 5.2

Métricas de error asociadas a la ejecución del algoritmo Private Count Mean Sketch, en función de los parámetros m y k. (dataset: exp\_distrib\_50k y  $\varepsilon = 2$ )

PARÁMETROS		COSTE TEMPORAL		
k	m	Cliente	Servidor (Actualizar matriz)	Servidor (Estimación)
16	16	0.0087 ms	0.0023 ms	0.0345 ms
16	64	0.0113 ms	0.0024 ms	0.0351 ms
16	256	0.0507 ms	0.0024 ms	0.0391 ms
16	1024	0.6846 ms	0.0029 ms	0.0354 ms
128	16	0.0087 ms	0.0024 ms	0.2206 ms
128	64	0.0113 ms	0.0024 ms	0.3815 ms
128	256	0.0507 ms	0.0029 ms	0.3716 ms
128	1024	0.6570 ms	0.0028 ms	0.3787 ms
1024	16	0.0088 ms	0.0024 ms	1.6807 ms
1024	64	0.0114 ms	0.0027 ms	2.5006 ms
1024	256	0.0509 ms	0.0035 ms	2.7016 ms
1024	1024	0.7008 ms	0.0031 ms	2.6171 ms
32768	16	0.0096 ms	0.0026 ms	58.6125 ms
32768	64	0.0122 ms	0.0027 ms	60.3469 ms
32768	256	0.0523 ms	0.0033 ms	65.8501 ms
32768	1024	0.6551 ms	0.0038 ms	65.6921 ms
65536	16	0.0097 ms	0.0026 ms	113.4329 ms
65536	64	0.0125 ms	0.0028 ms	117.4876 ms
65536	256	0.0525 ms	0.0036 ms	126.0397 ms
65536	1024	0.6811 ms	0.0054 ms	125.7506 ms

Tabla 5.3

*Coste temporal asociado a la ejecución del algoritmo Private Hadamard Count Mean Sketch, en función de los parámetros m y k. (dataset: exp\_distrib\_50k y  $\varepsilon = 2$ )*

el PHCMS obtiene tiempos bastante inferiores al PCMS ya que únicamente actualiza un valor de la matriz, mientras que en PCMS el número de valores a actualizar es  $m$ . Finalmente, el tiempo requerido para realizar una estimación en ambos algoritmos se comporta de forma similar y está directamente relacionada con el parámetro  $k$ . Por lo general, el PHCMS tiene tiempos de estimación ligeramente superiores que pueden ser provocados por pequeñas diferencias en la implementación.

En términos del error, todas las métricas recogidas, a excepción del coeficiente de correlación de Pearson, reflejan una mayor precisión por parte del algoritmo PCMS. Todas estas métrica indican que las predicciones obtenidas por el algoritmo PCMS están más cerca de los valores reales. El incremento de los parámetros  $k$  y  $m$  supone un incremento de la precisión en ambos algoritmos, aunque este incremento es bastante mayor en PHCMS. Una característica que se aprecia en la Tabla 5.4, es que la elección de  $m$  influye bastante en el error. Valores pequeños de este parámetro empeoran notablemente el rendimiento.

Respecto al coeficiente de correlación de Pearson, ambos algoritmos obtienen resultados similares que muestran que las estimaciones reflejan correctamente las tendencias presentes en los datos reales. Las estimaciones están muy alineadas con los valores reales. A medida que la frecuencia real de un elemento aumenta, la frecuencia estimada también lo hace de manera proporcional.

Con el objetivo de evaluar el impacto del parámetro de privacidad  $\varepsilon$  en ambos algoritmos se han recogido los resultados de un conjunto de ejecuciones, los cuales se pueden ver en las Tablas 5.5 y 5.6. Los parámetros  $k$  y  $m$  se han fijado a partir de los resultados obtenidos previamente y la ventaja computacional que estos suponen frente

PARÁMETROS		MÉTRICAS DE ERROR						
k	m	Media Errores	Error porcentual	MSE	RMSE	MSE (Normalizado)	RMSE (Normalizado)	C.C. Pearson
16	16	4164.79	8.33 %	32794319.65	5726.63	1668.92	40.85	0.6640
16	64	2655.63	5.31 %	15715151.22	3964.23	799.75	28.28	0.6761
16	256	1928.71	3.86 %	13889392.96	3726.85	706.84	26.59	0.7283
16	1024	1757.84	3.52 %	12123228.19	3481.84	616.96	24.84	0.7142
128	16	3777.00	7.55 %	24434758.85	4943.15	1243.50	35.26	0.7351
128	64	2045.57	4.09 %	14697986.57	3833.80	747.99	27.35	0.7026
128	256	1694.28	3.39 %	11437428.81	3381.93	582.06	24.13	0.7150
128	1024	1813.78	3.63 %	12808624.22	3578.91	651.84	25.53	0.7146
1024	16	3509.35	7.02 %	21851493.75	4674.56	1112.04	33.35	0.6993
1024	64	2100.27	4.20 %	13921635.57	3731.17	708.48	26.62	0.7119
1024	256	1883.67	3.77 %	14071775.77	3751.24	716.12	26.76	0.7190
1024	1024	1757.98	3.52 %	11668119.17	3415.86	593.80	24.37	0.7129
32768	16	3715.77	7.43 %	24030291.90	4902.07	1222.92	34.97	0.6981
32768	64	2157.01	4.31 %	14790649.96	3845.86	752.70	27.44	0.7144
32768	256	1812.54	3.63 %	12192833.16	3491.82	620.50	24.91	0.7222
32768	1024	1782.65	3.57 %	12578034.69	3546.55	640.10	25.30	0.7205
65536	16	3657.30	7.31 %	22539980.40	4747.63	1147.07	33.87	0.7032
65536	64	2177.42	4.35 %	14933179.67	3864.35	759.96	27.57	0.7073
65536	256	1799.16	3.60 %	12494845.85	3534.80	635.87	25.22	0.7148
65536	1024	1746.27	3.49 %	12912105.08	3593.34	657.10	25.63	0.7164

Tabla 5.4

Métricas de error asociadas a la ejecución del algoritmo Private Hadamard Count Mean Sketch, en función de los parámetros m y k. (dataset: exp\_distrib\_50k y  $\varepsilon = 2$ )

a valores mayores. Se han eliminado las métricas asociadas al coste temporal debido a que la elección de  $\varepsilon$  no implica un cambio en estas.

Para ambos algoritmos, se cumple que el error aumenta en cada caso para ambos algoritmos a medida que el parámetro  $\varepsilon$  disminuye, algo tiene sentido ya que a valores más pequeño, mayor privacidad se está exigiendo, lo que conlleva una pérdida de valor en la información. La Figura 5.4, representa esta caída de la media de error para valores relativamente pequeños de  $\varepsilon$ .

PARÁMETROS		MÉTRICAS DE ERROR					
$\varepsilon$	Media Errores	Error porcentual	MSE	RMSE	MSE (Normalizado)	RMSE (Normalizado)	
0.05	13772.49	6.89 %	304703800.08	17455.77	7916.85	88.98	
0.1	6477.56	3.24 %	57279284.94	7568.31	1488.24	38.58	
0.5	1668.19	0.83 %	3964441.50	1991.09	103.00	10.15	
2	351.34	0.18 %	193480.20	439.86	5.03	2.24	
4	203.72	0.10 %	53573.61	231.46	1.39	1.18	
8	163.01	0.08 %	35293.25	187.86	0.92	0.96	
12	94.02	0.05 %	16578.24	128.76	0.43	0.66	
48	95.94	0.05 %	17364.27	131.77	0.45	0.67	
128	99.31	0.05 %	12871.82	113.45	0.33	0.58	

Tabla 5.5

Resultados de ejecución del algoritmo Private Count Mean Sketch sobre un dataset que representa una distribución normal ( $\mu = 12, \sigma = 2$ ) (Dataset: norm\_distrib\_200k, k = 1024 y m = 256)

Ambos algoritmos han sido sometidos a diferentes ejecuciones con los mismos parámetros de entrada  $\varepsilon$ , m y k, variando la cantidad de registros del dataset empleado, con el objetivo de analizar la evolución del error porcentual a medida que aumenta

PARÁMETROS	MÉTRICAS DE ERROR					
	$\varepsilon$	Media Errores	Error porcentual	MSE	RMSE	MSE (Normalizado)
0.05	30363.91	15.18 %	1485927346.13	38547.73	38607.55	196.49
0.1	18283.35	9.14 %	559494940.74	23653.65	14536.87	120.57
0.5	13318.63	6.66 %	307511372.39	17536.00	7989.80	89.39
2	7651.61	3.83 %	131303682.21	11458.78	3411.55	58.41
4	3196.45	1.60 %	22555998.19	4749.32	586.05	24.21
8	536.26	0.27 %	494910.95	703.50	12.86	3.59
12	299.61	0.15 %	123988.99	352.12	3.22	1.79
48	348.13	0.17 %	202429.97	449.92	5.26	2.29
128	275.96	0.14 %	124533.48	352.89	3.24	1.80

Tabla 5.6

Resultados de ejecución del algoritmo Private Hadamard Count Mean Sketch sobre un dataset que representa una distribución normal ( $\mu = 12, \sigma = 2$ ) (Dataset: norm\_distrib\_200k,  $k = 1024$  y  $m = 256$ )

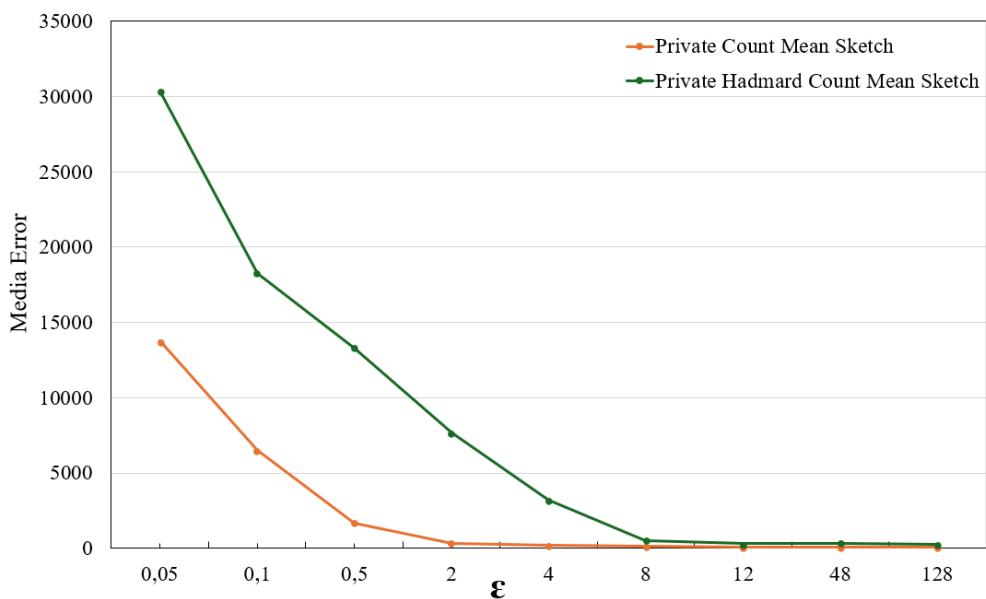


Figura 5.4: Gráfica comparativa de la evolución de la media de error en ambos algoritmos en función del valor de  $\varepsilon$

el tamaño de los datos. Para todas las ejecuciones se ha optado por valores correspondientes a una distribución normal, de características estadísticas similares a las utilizadas en todo el experimento.

La Figura 5.5 muestra gráficamente el decremento del error porcentual a medida que el volumen de datos aumenta, en ambos algoritmos. Al aumentar la cantidad de registros procesados, la matriz de sketch contiene una mayor cantidad de información, lo que disminuye las deficiencias propias de una estructura de datos basada en funciones hash y valores privatizados. En términos de la fórmula empleada a la hora de realizar una estimación,  $N$  tiene una relación directa. Para valores pequeños de  $N$ , la corrección aplicada a la estimación media es reducida. Esto junto a la falta de información al contar con pocos datos, produce un error proporcional mayor. A partir de  $N \geq 50000$  se puede apreciar como la curva del error tiende a aplanarse, obteniendo en el algoritmo PCMS valores cercanos a 0.

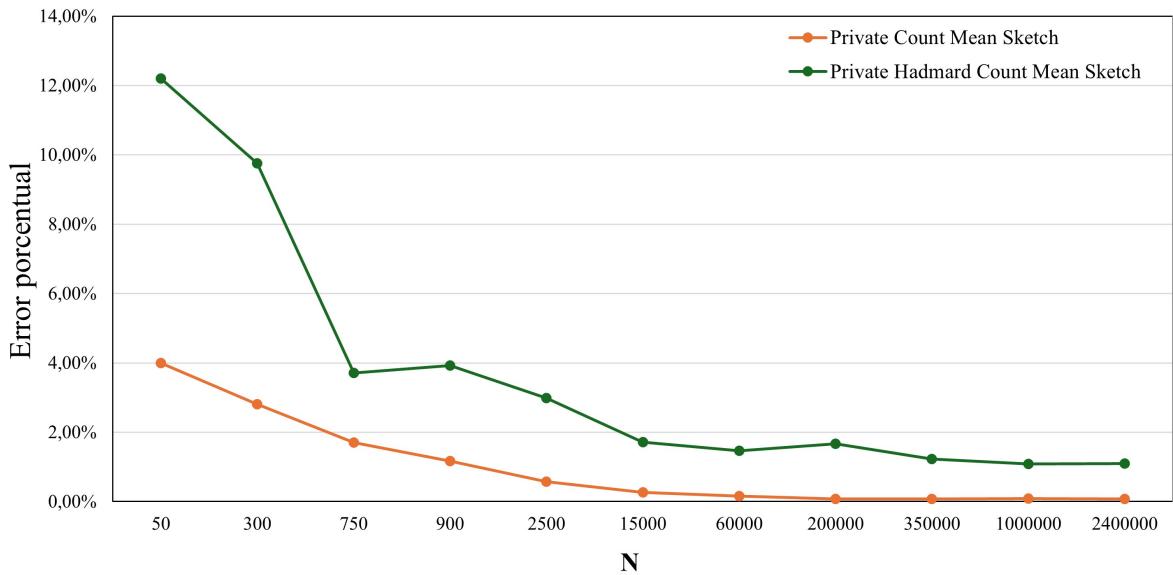


Figura 5.5: Evolución del error porcentual en ambos algoritmos en función del incremento del tamaño del dataset ( $N$ ). Parámetros empleados  $\varepsilon = 4$ ,  $k = 1024$  y  $m = 256$ .

Una diferencia notable y que conlleva que el algoritmo PHCMS obtenga resultados menos precisos que PCMS, es el hecho de que el cliente envía al servidor un único bit de información. Esto se trata de una premisa teórica, ya que dependiendo de la implementación, pese a ser una única unidad de información la que se envía, esta ocupa más de 1 bit. No obstante, durante las ejecuciones descritas anteriormente se ha medido el coste de trasmisión de la información enviado por un cliente, en kilo-bytes.

Los resultados obtenidos pueden verse en las Tablas 5.7 y 5.8. El coste de trasmisión depende directamente del valor del parámetro  $m$  debido a que cuando un elemento se privatiza, el cliente envía al servidor un vector de  $m$  elementos. Como en el algoritmo Private Hadamard Count Mean Sketch siempre se envía un elemento de dicho vector privatizado, entre otros elementos, el coste de trasmisión no varía.

<b>m</b>	<b>Coste Trasmisión</b>
16	268.00 kB
64	652.00 kB
256	2184.00 kB
512	4236.00 kB
1024	8332.00 kB

Tabla 5.7

*Coste de transmisión (Ancho de Banda) de los datos enviados por el cliente en Private Count Mean Sketch en función del parámetro m*

<b>m</b>	<b>Coste Trasmisión</b>
16	88.00 kB
64	88.00 kB
256	88.00 kB
512	88.00 kB
1024	88.00 kB

Tabla 5.8

*Coste de transmisión (Ancho de Banda) de los datos enviados por el cliente en Private Hadamard Count Mean Sketch en función del parámetro m*

Los resultados obtenidos a lo largo de todo el experimento sugieren que, aunque el algoritmo Private Hadamard Count Mean Sketch tiende a ser más eficiente en términos de coste temporal y coste de trasmisión, Private Count Mean Sketch ofrece una precisión significativamente mayor en sus estimaciones. De cara a la implantación de alguno de estos algoritmos en un sistema, se debe considerar la importancia relativa de la eficiencia temporal en comparación a la precisión de la estimación, en el contexto de aplicación específico.

### 5.5.2. Experimento 2. Análisis de la capacidad del algoritmo Private Secuence Fragment Puzzle para la estimación de un diccionario desconocido.

Este experimento tiene por objetivo analizar el comportamiento del algoritmo Private Secuence Fragment Puzzle según la variación de sus parámetros de entrada. El dataset empleado a lo largo del análisis es **anglicismo\_50k** con 50000 registros, cuyos elementos y frecuencias pueden verse en la Figura 5.1. En un primer momento, se han fijado los parámetros  $\varepsilon = 2$ ,  $\varepsilon' = 6$  y el umbral  $T = 10$ . La elección de los valores  $\varepsilon$  y  $\varepsilon'$  se justifica en consecuencia de su influencia sobre el algoritmo Private Count Mean Sketch, demostrada en el experimento anterior. Estos parámetros no presentan un impacto significativo en el tiempo de ejecución necesario para generar el diccionario. Por el contrario, el análisis debe enfocarse en el conjunto de parámetros que sí afectan de manera directa el rendimiento temporal del algoritmo. Además, anteriormente se demostró que a partir de valores mayores que 1, dicho parámetro influye de manera similar.

Por otro lado, la elección del umbral  $T$  está relacionado con el número de cadenas estimadas que se generan. A mayor valor de  $T$ , se incluirán cadenas que probablemente aporten ruido al diccionario generado. Cuando  $T$  toma valores pequeños el error va a ser menor, ya que se incluirán en el diccionario aquellas combinaciones de subcadenas con una mayor frecuencia estimada. En términos de tiempo, el incremento de  $T$  conlleva un incremento en el tiempo necesario para generar el diccionario, ya que se generan un mayor número cadenas candidatas. Debido a que se conoce el impacto de este parámetro en el rendimiento del algoritmo, no resulta oportuno incluir variaciones de este parámetro en las ejecuciones, ya que se trata de un algoritmo con gran cantidad de parámetros de entrada y tiempos de ejecución bastante altos.

Antes de proceder con el análisis de los resultados, es necesario destacar que pese a que las métricas empleadas son las mismas que las del resto de experimentos, sería injusto comparar los resultados obtenidos con algoritmos evaluados anteriormente. Pese a ser algoritmos de propósito similar, estimar frecuencias a partir de un flujo masivo de datos, hay que tener en cuenta el hecho de conocer o no el dominio de los elementos. En términos de tiempo, Private Sequence Fragment Puzzle es un algoritmo mucho más costoso que el resto puesto que internamente utiliza sucesivas instancias del algoritmo Private Count Mean Sketch. Respecto a las métricas de error, hay que tener en cuenta que para este algoritmo se comparan las frecuencias estimadas de las cadenas candidatas respecto a la frecuencia real de estas. Al seleccionar un valor reducido de  $T$ , se obtienen un conjunto de cadenas reducido que además, tiende a coincidir con las cadenas con mayor frecuencia real, por lo que el error es bajo en comparación a otros algoritmos, que adicionalmente estiman la frecuencia de elementos con baja ocurrencia en el flujo de datos.

Durante las sucesivas ejecuciones realizadas, los parámetros  $k$  y  $k'$  han tomado valores pertenecientes al conjunto  $\{32, 64, 256, 1024, 2048\}$ , y los parámetros  $m$  y  $m'$   $\{16, 32, 64, 256, 1024\}$ . La Tabla 5.9 recoge los tiempos de ejecución del algoritmos Sequence Fragment Puzzle en función de los valores que toman los parámetros  $k$ ,  $k'$ ,  $m$  y  $m'$ . El tiempo que tarda un cliente en seleccionar una subcadena, realizar de forma consecutiva dos enmascaramientos como en el algoritmo Private Count Mean Sketch, y enviarlo al servidor, es invariante de estos parámetros de estudio y se aproxima a los 0,02 milisegundos. El tiempo que el servidor tarda en realizar las actualizaciones pertinentes sobre las diferentes matrices de sketch implicadas, está directamente proporcionado con el valor que toman los parámetros  $m$  y  $m'$ , al igual que ocurre en los algoritmos probados en 5.5.1. La operación mas costosa, con diferencia, de todo el algoritmo es la generación del diccionario con cadenas candidatas. El parámetro con mayor impacto en esta operación es  $k'$ , tal y como se puede apreciar en la Figura 5.6. Esto es debido a que en una etapa de la generación del diccionario se realizan estimaciones sobre matrices de sketch, inicializadas con  $k = k'$ . El numero de estimaciones es muy elevado, se utilizan un alto número de combinaciones posibles de un número hasta 256 y una subcadena de tamaño 2. Según la implementación, el tiempo necesario para realizar una estimación en una matriz de sketch es del orden  $O(k)$ , lo que deriva en tiempos muy altos, del orden de minutos, cuando  $k'$  toma valores grandes, a partir de 1024.

La Tabla 5.10 recoge los resultados en términos de error recogidos por la sucesivas ejecuciones. Hay que tener en cuenta el valor umbral  $T = 10$ , elegido de forma común. Al ser este valor pequeño, el algoritmo se comporta de manera más estricta en el

PARÁMETROS				COSTE TEMPORAL		
<i>k</i>	<i>k'</i>	<i>m</i>	<i>m'</i>	Cliente	Servidor (Actualizar Matriz)	Servidor (Generar Diccionario)
32	32	16	16	0.0154 ms	0.0163 ms	34.3462 s
32	32	32	32	0.0157 ms	0.0220 ms	34.4705 s
32	64	16	16	0.0153 ms	0.0163 ms	67.9959 s
32	64	32	32	0.0157 ms	0.0220 ms	67.8417 s
64	32	32	32	0.0159 ms	0.0220 ms	34.4804 s
64	64	16	16	0.0155 ms	0.0163 ms	67.7929 s
64	64	32	32	0.0159 ms	0.0221 ms	67.8589 s
256	256	64	64	0.0165 ms	0.0333 ms	268.5827 s
256	256	256	256	0.0212 ms	0.1028 ms	1916.9922 s
256	256	1024	1024	0.0372 ms	0.3863 ms	270.6663 s
256	1024	64	64	0.0165 ms	0.0335 ms	1096.0001 s
256	1024	256	256	0.0214 ms	0.1026 ms	1233.2381 s
256	256	1024	1024	0.0373 ms	0.3865 ms	272.6547 s
256	256	64	64	0.0173 ms	0.0342 ms	270.2836 s
256	256	256	256	0.0211 ms	0.1028 ms	270.4174 s
256	256	1024	1024	0.0374 ms	0.3891 ms	275.6048 s
1024	256	64	64	0.0167 ms	0.0338 ms	269.9949 s
1024	256	256	256	0.0213 ms	0.1029 ms	268.2480 s
1024	256	1024	1024	0.0378 ms	0.3897 ms	272.5353 s
1024	1024	64	64	0.0166 ms	0.0335 ms	1100.7083 s
1024	1024	256	256	0.0215 ms	0.1038 ms	1103.9649 s
1024	1024	1024	1024	0.0377 ms	0.3866 ms	1104.8575 s
1024	2048	64	64	0.0183 ms	0.0350 ms	2200.8712 s
1024	2048	256	256	0.0214 ms	0.1031 ms	2201.7253 s
1024	2048	1024	1024	0.0379 ms	0.3869 ms	2267.5923 s
2048	256	64	64	0.0167 ms	0.0331 ms	268.1428 s
2048	256	256	256	0.0212 ms	0.1029 ms	269.1762 s
2048	256	1024	1024	0.0377 ms	0.3867 ms	271.1787 s
2048	1024	64	64	0.0168 ms	0.0335 ms	1097.4131 s
2048	1024	256	256	0.0215 ms	0.1033 ms	1096.5119 s
2048	1024	1024	1024	0.0379 ms	0.3875 ms	1088.18862 s
2048	2048	64	64	0.0168 ms	0.0334 ms	2195.0177 s
2048	2048	256	256	0.0215 ms	0.1027 ms	2203.6124 s
2048	2048	1024	1024	0.0378 ms	0.3870 ms	2266.6762 s

Tabla 5.9

Tiempo de ejecución obtenido tras sucesivas ejecuciones del algoritmo Private Sequence Fragment Puzzle ( $\varepsilon = 2$  y  $\varepsilon' = 6$ , dataset: anglicismo\_50k)

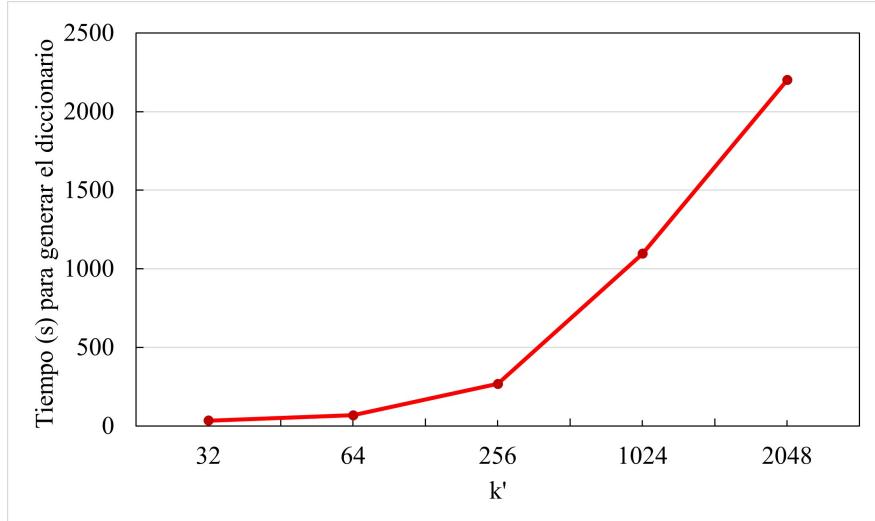


Figura 5.6: Evolución del tiempo de ejecución para la generación del diccionario frente al incremento de  $k'$ ,

sentido de que para proponer una cadena candidata, las subcadenas que componen a esta deben obtener valores muy altos de frecuencia, en consecuencia, dichas cadenas deben contar con una ocurrencia bastante alta en el flujo de datos. Dicha tabla muestra que el incremento de los cuatro parámetros variables supone una leve disminución de todas las medidas de error. No obstante, esta disminución del error no supone una gran mejora en comparación al coste computacional, en tiempo y espacio, necesario.

Al aumentar el valor que toman los parámetros  $k$ ,  $k'$ ,  $m$  y  $m'$ , el conjunto de cadenas candidatas estará soportado por unas estructuras de datos más robustas, por lo que los resultados obtenidos se acercarán más a la realidad. No obstante, el conjunto de cadenas candidatas está acotado superiormente por el valor de  $T$ . De nada sirve aumentar notablemente estos parámetros para un valor de  $T$  muy reducido. La Figura 5.7, muestra un ejemplo del resultado obtenido al aumentar en gran medida el valor del parámetro  $T$ . En este resultado se incluyen cadenas ruidosas (*fogfower 0 lilj*), cuya aparición en el flujo puede ser escasa o inexistente, de ahí que los valores de frecuencia asociados sean mínimos o inconsistentes.

La Tabla 5.11 muestra el impacto producido por el incremento del parámetro  $T$ , en dos métricas clave de cara a obtener resultados, el número de cadenas candidatas generadas por el algoritmo y la suma de los errores. Este error total representa la suma de los valores absolutos de las diferencias entre la frecuencia real y la frecuencia estimada de cada cadena candidata generada. Para aquellas cadenas consideradas como ruido, es decir, no tienen ocurrencia en el flujo de datos original, se incluye en la suma de errores el valor absoluto de la frecuencia estimada. Al incrementar  $T$ , el número de cadenas candidatas se dispara, en consecuencia, se incrementa el número de cadenas ruidosas que impactan negativamente en el error total. La Figura 5.8 muestra una tendencia exponencial en el número de cadenas candidatas al incrementar  $T$ . Esta tendencia subyace, de igual manera, en el número total de errores.

La elección final de los parámetros  $k$ ,  $k'$ ,  $m$ ,  $m'$  y  $T$  dependerá, en gran medida, del propósito del sistema. Ante una situación en la que el dominio subyacente, pese a

PARÁMETROS				MÉTRICAS DE ERROR					
<i>k</i>	<i>k'</i>	<i>m</i>	<i>m'</i>	Errores (Media)	Error (Porcentual)	MSE	RMSE	MSE (Normalizado)	RMSE (Normalizado)
32	32	16	16	817.10	1.63 %	944392.23	971.80	118.87	10.90
32	32	32	32	443.48	0.89 %	434089.41	658.85	54.64	7.39
32	64	16	16	572.82	1.15 %	374456.49	611.93	47.13	6.87
32	64	32	32	425.37	0.85 %	277670.57	526.94	34.95	5.91
64	32	32	32	375.42	0.75 %	175284.34	418.67	22.06	4.70
64	64	16	16	502.25	1.00 %	345158.33	587.50	43.44	6.59
64	64	32	32	398.63	0.79 %	247868.21	512.41	38.11	6.12
256	256	64	64	170.01	0.34 %	42016.29	204.98	5.29	2.30
256	256	256	256	185.08	0.37 %	45326.22	212.90	5.70	2.39
256	256	1024	1024	149.92	0.30 %	32783.68	181.06	4.13	2.03
256	1024	64	64	206.32	0.41 %	57649.53	240.10	7.26	2.69
256	1024	256	256	221.11	0.44 %	62343.87	249.69	7.85	2.80
256	256	1024	1024	184.33	0.37 %	46735.08	216.18	5.88	2.43
256	256	64	64	225.74	0.45 %	74942.38	273.76	9.43	3.07
256	256	256	256	251.68	0.50 %	85555.96	292.50	10.77	3.28
256	256	1024	1024	122.79	0.25 %	21682.24	147.25	2.73	1.65
1024	256	64	64	114.36	0.23 %	20184.49	142.07	2.54	1.59
1024	256	256	256	215.25	0.43 %	68650.45	262.01	8.64	2.94
1024	256	1024	1024	178.66	0.36 %	38229.39	195.52	4.81	2.19
1024	1024	64	64	263.57	0.53 %	105706.29	325.13	13.30	3.65
1024	1024	256	256	235.87	0.47 %	79997.67	282.84	10.07	3.17
1024	1024	1024	1024	93.20	0.19 %	15246.63	123.48	1.92	1.39
1024	2048	64	64	272.05	0.54 %	100926.90	317.69	12.70	3.56
1024	2048	256	256	150.09	0.30 %	32779.30	181.05	4.13	2.03
1024	2048	1024	1024	261.25	0.52 %	95474.62	308.99	12.02	3.47
2048	256	64	64	189.47	0.38 %	51129.84	226.12	6.44	2.54
2048	256	256	256	189.90	0.38 %	51598.12	227.15	6.49	2.55
2048	256	1024	1024	170.84	0.34 %	44621.32	211.24	5.62	2.37
2048	1024	64	64	167.46	0.33 %	41192.71	202.96	5.18	2.28
2048	1024	256	256	173.55	0.35 %	42470.75	206.08	5.35	2.31
2048	1024	1024	1024	149.23	0.30 %	31383.16	177.15	3.95	1.99
2048	2048	64	64	163.87	0.33 %	34383.37	185.43	4.33	2.08
2048	2048	256	256	184.44	0.37 %	48534.99	220.31	6.11	2.47
2048	2048	1024	1024	127.27	0.25 %	27886.84	166.99	3.51	1.87

Tabla 5.10

Métricas de error obtenidas tras sucesivas ejecuciones del algoritmo Private Sequence Fragment Puzzle ( $\varepsilon = 2$  y  $\varepsilon' = 6$ , dataset: anglicismo\_50k ).

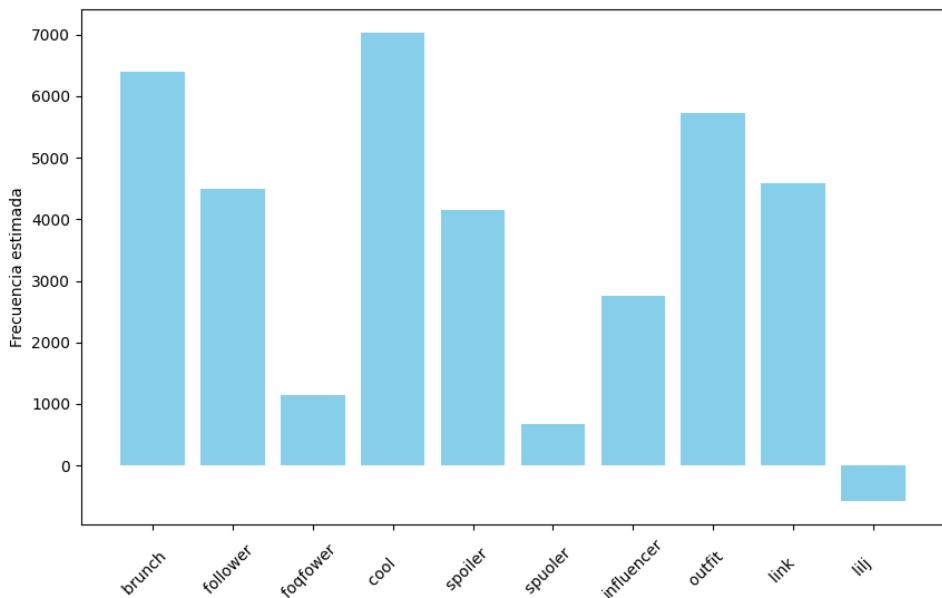


Figura 5.7: Ejemplo de diccionario con falsos positivos generado tras ejecutar el algoritmo Sequence Fragment Puzzle ( $T = 80$ )

T	Errores (Total)	Cadenas Candidatas
10	2197.74	9
20	2392.20	14
30	3674.96	18
40	2561.47	14
50	3245.75	16
60	3382.21	17
80	6297.41	32
100	31293.09	152
150	40144.91	217
200	27521.63	130
250	72485.52	395
300	204383.99	1038
400	469032.36	2361
500	1953907.47	9716

Tabla 5.11

Total de errores y número de cadenas candidatas generadas al ejecutar el algoritmo Sequence Fragment Puzzle ( $\varepsilon = 2$ ,  $\varepsilon' = 6$ ,  $k = 256$ ,  $k' = 256$ ,  $m = 64$  y  $m' = 64$ ) variando el valor que toma el umbral  $T$ .

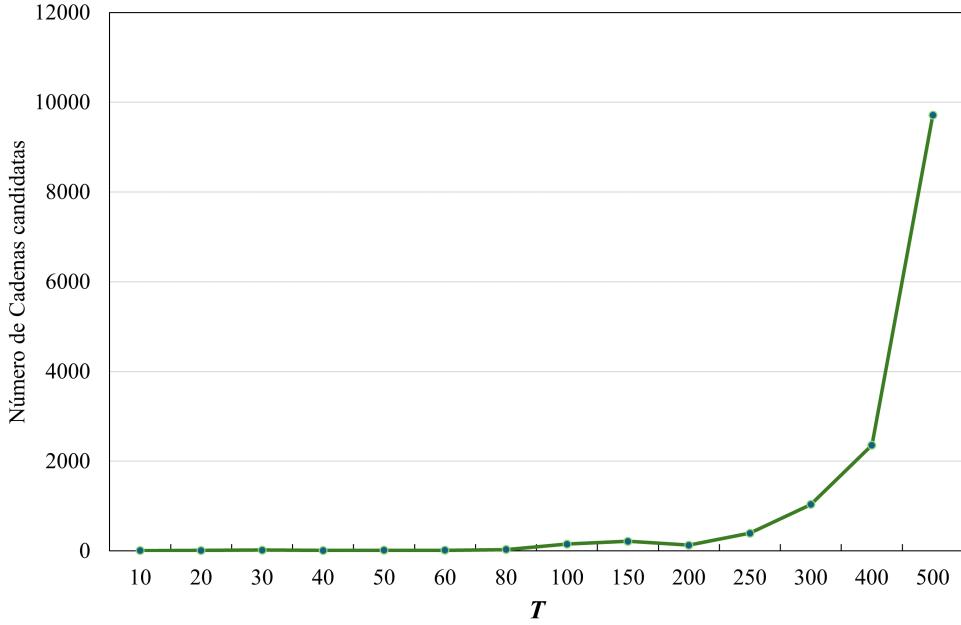


Figura 5.8: Evolución del número de cadenas candidatas generadas en función del parámetro  $T$ .

su desconocimiento, se presupone que es grande y con una distribución equilibrada, y se desea obtener un gran número de cadenas candidatas fiables, es recomendable optar por valores altos en todos los parámetros. A mayor  $T$ , las cadenas generadas contarán con un mayor nivel de fiabilidad u ocurrencia dentro del flujo real de datos, según la magnitud del resto de parámetros. Resulta interesante destacar que para un número pequeño de registros ( $N < 1000$ ), siempre que el valor de  $T$  sea reducido (en torno a 10), el algoritmo no es capaz de generar ninguna cadena candidata. Por lo tanto, se puede concluir que es un algoritmo que necesita de una gran cantidad de registros para producir resultados fiables.

### 5.5.3. Experimento 3. Evaluación del rendimiento del algoritmo RAPPOR

El presente experimento tiene por objetivo analizar el rendimiento del algoritmo RAPPOR, descrito en la sección 4.4, en términos de coste temporal y error. A lo largo del experimento se ha empleado el dataset **exp\_distrib\_500k** y el conjunto de datasets correspondientes a una distribución normal, pero de tamaño variable, **norm\_distrib\_50**, **norm\_distrib\_300**, **norm\_distrib\_750**, **norm\_distrib\_900**, **norm\_distrib\_2500**, **norm\_distrib\_15k**, **norm\_distrib\_60k**, **norm\_distrib\_350k**, **norm\_distrib\_1M**, **norm\_distrib\_2.4M**. Estos últimos se emplearán para medir el impacto del número de registros del dataset en los resultados. Por otro lado, se variarán los valores que toman los parámetros de ejecución  $m$ ,  $k$ ,  $f$ ,  $p$  y  $q$ , con el objetivo de analizar el comportamiento del algoritmo en diferentes casos de estudio.

En primer lugar, se han fijado los valores de los parámetros  $f = 0,5$ ,  $p = 0,5$ ,

PARÁMETROS		COSTE TEMPORAL	
m	k	Cliente (Por usuario)	Servidor (Estimar Frecuencias)
16	1	0.0094 ms	356.7209 ms
16	2	0.0103 ms	350.1508 ms
16	8	0.0275 ms	348.7728 ms
16	32	0.2876 ms	359.1962 ms
16	64	1.1642 ms	376.7619 ms
64	1	0.0314 ms	1259.7759 ms
64	2	0.0322 ms	1255.9581 ms
64	8	0.0493 ms	1246.2239 ms
64	32	0.3141 ms	1247.7219 ms
64	64	1.1832 ms	1259.4380 ms
128	1	0.0610 ms	2457.6943 ms
128	2	0.0620 ms	2451.7698 ms
128	8	0.0791 ms	2443.9161 ms
128	32	0.3424 ms	2435.5068 ms
128	64	1.2124 ms	2458.1549 ms
512	1	0.2337 ms	9697.1409 ms
512	2	0.2370 ms	9659.9059 ms
512	8	0.2536 ms	9638.4468 ms
512	32	0.5165 ms	9641.1283 ms
512	64	1.3897 ms	9623.8241 ms

Tabla 5.12

Resultados obtenidos, en términos de coste temporal, tras la ejecución del algoritmo RAPPOR según los parámetros  $m$  y  $k$ . ( $f = 0,5$ ,  $p = 0,5$ ,  $q = 0,75$ , dataset: `exp_distrib_500k`)

$q = 0,75$  y dataset: `exp_distrib_500k`, y se han realizado sucesivas ejecuciones variando los parámetros  $m$  y  $k$ . A modo de recordatorio,  $m$  corresponde al número de bits de los filtros de Bloom empleados y  $k$  corresponde con el número de funciones hash, de una familia  $k$ -independiente, utilizadas. La Tabla 5.12 muestra el tiempo requerido por el algoritmo para tomar un dato sin privatizar y enmascararlo, así como el tiempo necesario por el algoritmo para obtener el histograma de frecuencias estimadas. El tiempo del cliente depende directamente del número de bits empleados en el filtro de Bloom y del número de funciones hash utilizadas. El tiempo necesario para estimar el histograma se ve afectado principalmente por el parámetro  $m$ , llegando a tiempos en torno a 9 segundos cuando  $m = 512$ . Este resultado tiene sentido ya que durante la etapa de análisis, se construyen diferentes estructuras de datos matriciales, cuyas dimensiones dependen del valor de dicho parámetro.

La Tabla 5.13 muestra la precisión obtenida por el algoritmo, en las condiciones descritas anteriormente. Los resultados obtenidos muestran la importancia del parámetro  $k$  en la precisión de los resultados, cuando  $m$  toma valores pequeños. No obstante, cuando  $k > m$ , la precisión obtenida empeora bastante. A medida que el valor de  $m$  aumenta, los resultados tienden a ser más precisos independientemente del valor de  $k$ . Notándose apenas diferencia entre seleccionar 8 o 32 funciones hash cuando  $m \geq 64$ , pese a la diferencia existente en el tiempo necesario para privatizar un dato. Una característica destacable es que el aumento de  $m$  no provoca un impacto sobre la precisión cuando  $k = 1$ . En otras palabras, independientemente del número de bits empleados, cuando únicamente se utiliza 1 función hash, los resultados muestran

PARÁMETROS		MÉTRICAS DE ERROR						
m	k	Media Errores	Error Porcentual	MSE	RMSE	MSE (Normalizado)	RMSE (Normalizado)	Coefficiente Correlación Pearson
16	1	23332.42	4.67 %	4404018432.28	66362.78	22383.37	149.61	0.7974
16	2	3926.89	0.79 %	23616092.93	4859.64	120.03	10.96	0.8059
16	8	10934.73	2.19 %	218537208.72	14783.00	1110.71	33.33	0.8394
16	32	8256.33	1.65 %	305884232.10	17489.55	1554.65	39.43	0.9425
16	64	19809.57	3.96 %	2356144295.48	48540.13	11975.08	109.43	-0.1498
64	1	23356.90	4.67 %	4418888258.63	66474.72	22458.95	149.86	0.7974
64	2	1330.88	0.27 %	3734087.62	1932.38	18.98	4.36	0.8207
64	8	775.30	0.16 %	1002331.37	1001.17	5.09	2.26	0.8220
64	32	430.14	0.09 %	429842.89	655.62	2.18	1.48	0.8191
64	64	435.36	0.09 %	549690.75	741.41	2.79	1.67	0.8172
128	1	23230.30	4.65 %	4342320644.68	65896.29	22069.80	148.56	0.7974
128	2	941.11	0.19 %	1673550.41	1293.66	8.51	2.92	0.8174
128	8	631.77	0.13 %	771631.40	878.43	3.92	1.98	0.8218
128	32	395.25	0.08 %	250669.74	500.67	1.27	1.13	0.8169
128	64	333.24	0.07 %	235078.22	484.85	1.19	1.09	0.8193
512	1	23270.83	4.65 %	4366741762.25	66081.33	22193.92	148.98	0.7974
512	2	859.87	0.17 %	1972988.51	1404.63	10.03	3.17	0.8091
512	8	609.13	0.12 %	835055.20	913.81	4.24	2.06	0.8206
512	32	405.17	0.08 %	238397.56	488.26	1.21	1.10	0.8164
512	64	227.83	0.05 %	92162.36	303.58	0.47	0.68	0.8173

Tabla 5.13

Resultados obtenidos, en términos de error, tras la ejecución del algoritmo RAPPOR según los parámetros m y k. ( $f = 0.5$ ,  $p = 0.5$ ,  $q = 0.75$ , dataset: exp\_distrib\_500k)

una pérdida notable de precisión. Pese a que a mayores valores de  $m$  y  $k$ , se obtiene una mayor precisión, hay que sopesar el coste temporal asociado a este incremento, y analizar si este compensa dado el caso concreto.

La Tabla 5.15 muestra los resultados obtenidos de sucesivas ejecuciones del algoritmo RAPPOR, variando los parámetros probabilísticos  $f$ ,  $p$  y  $q$ , los cuales influyen directamente en las etapas de perturbación permanente ( $\varepsilon_{PP}$ ) y temporal ( $\varepsilon_{PT}$ ). Estos parámetros están directamente relacionados con el valor de  $\varepsilon$ , en el contexto de la privacidad  $\varepsilon$ -diferencial y se obtienen mediante las Ecuaciones 4.7 y 4.10, presentadas en la sección 4.4. No se ha incluido información acerca del coste temporal del algoritmo puesto que estos parámetros no tienen influencia en dichas métricas. Se han fijado los valores de los parámetros  $m = 128$  y  $k = 8$ , y el dataset empleado ha sido exp\_distrib\_500k.

El parámetro  $f$  se hace notar en la etapa del perturbación permanente. Cuando este parámetro toma valores pequeños, se obtiene una mayor precisión en los resultados. A medida que este valor aumenta, el nivel de ruido añadido se incrementa y la precisión empeora, a la vez que el valor  $\varepsilon_{PP}$  disminuye. Esto implica que los resultados sean bastante robustos frente a las alteraciones en los datos individuales, lo que proporciona mayor privacidad a las personas cuyos datos están en la base. Los parámetros  $p$  y  $q$  influyen en los resultados de acuerdo a su ocurrencia en la implementación de la etapa de perturbación temporal. A medida que  $q$  aumenta, la probabilidad de que los bits activos del filtro de Bloom mantengan su estado se incrementa. Añadiendo una menor cantidad de ruido, y, en consecuencia, menor nivel de privacidad y mayor precisión en los resultados. Con el parámetro  $p$  ocurre algo diferente, a medida que este aumenta, la probabilidad de que los bits inactivos del filtro de Bloom mantengan su estado disminuye, provocando un impacto inverso a  $q$ . No obstante, para esta etapa de

perturbación temporal también influye el valor de  $f$ , de forma similar que en la etapa de perturbación permanente.

Por último y con el objetivo de analizar el impacto del número de registros en los resultados, se han realizado sucesivas ejecuciones del algoritmo fijando los parámetros  $m = 128$ ,  $k = 8$ ,  $f = 0,5$ ,  $p = 0,5$  y  $q = 0,75$ , y variando el número de registros de los datasets empleados. Siendo estos, **norm\_distrib\_50**, **norm\_distrib\_300**, **norm\_distrib\_750**, **norm\_distrib\_900**, **norm\_distrib\_2500**, **norm\_distrib\_15k**, **norm\_distrib\_60k**, **norm\_distrib\_350**, **norm\_distrib\_1M** y **norm\_distrib\_2.4M**. En términos de eficiencia temporal, el tiempo necesario para estimar el histograma se ve influido más que por el número de registros, por la dimensión del dominio de estudio. En este caso, y por cómo se han generado los diferentes datasets sintéticos, al incrementar  $N$ , aumenta el número de elementos del dominio. Esto ocurre al tratarse de una distribución probabilística. Respecto al error obtenido, al aumentar el número de registros del dataset, se observa una mejora progresiva en la precisión de los resultados hasta valores del orden de decenas de miles de registros. A partir de este punto, el incremento en la precisión tiende a estabilizarse, mostrando una clara disminución en el ritmo de mejora.

DATASET	COSTE TEMPORAL Servidor (Estimar frecuencias)	MÉTRICAS ERROR	
		Error Porcentual	Coeficiente Correlación Pearson
50	1.1539 ms	11.43 %	-0.1409
300	2.4641 ms	5.68 %	0.4270
750	4.5850 ms	2.96 %	0.2433
900	5.6601 ms	3.43 %	0.7587
2500	13.2110 ms	2.08 %	0.7684
15000	74.1279 ms	0.81 %	-0.1719
60000	293.0222 ms	0.31 %	0.5256
200000	974.1979 ms	0.21 %	0.6830
350000	1703.8269 ms	0.16 %	0.8382
1000000	4864.1129 ms	0.10 %	0.7696
2400000	12126.6320 ms	0.06 %	0.7407

Tabla 5.14

*Resultados obtenidos tras la ejecución del algoritmo RAPPOR según el tamaño del dataset. ( $m = 128$ ,  $k = 8$ ,  $f = 0,5$ ,  $p = 0,5$ ,  $q = 0,75$ , dataset: norm\_distrib\_N)*

Una característica destacable de este algoritmo, la cual comparte con la mayoría de algoritmos descritos, es la dificultad para estimar de forma precisa la frecuencia de los elementos con menor ocurrencia en el flujo de datos. Las Figuras 5.9 y 5.10 muestran gráficamente esta problemática. La salida mostrada corresponde a la ejecución del algoritmo sobre el dataset **exp\_distrib\_1M**, de 1 millón de registros pertenecientes a una distribución exponencial. Pese a que existe un margen de error presente en todos los elementos del dominio, para elementos con un valor alto de frecuencia real, Figura 5.9, esta diferencia apenas se percibe y, en esencia, no modifica la realidad. Al observar la Figura 5.10, se puede ver como los resultados asociados a elementos de baja frecuencia real tienden a distorsionarse. Esto provoca que algunos elementos prácticamente inexistentes en el flujo de datos real, obtengan frecuencias estimadas del orden de miles o, por el contrario, que elementos con una frecuencia real baja pero notable, no estén presentes en el histograma estimado.

PARÁMETROS			MÉTRICAS ERROR		PRIVACIDAD $\varepsilon$ -DIFERENCIAL	
f	p	q	Media Errores	Error Porcentual	$\varepsilon_{PP}$	$\varepsilon_{PT}$
0.05	0.25	0.5	284.89	0.06 %	58.62	3.62
0.05	0.25	0.75	96.46	0.02 %	58.62	7.18
0.05	0.25	0.95	123.21	0.02 %	58.62	12.62
0.05	0.5	0.25	270.69	0.05 %	58.62	3.62
0.05	0.5	0.75	244.31	0.05 %	58.62	3.62
0.05	0.5	0.95	212.09	0.04 %	58.62	9.33
0.05	0.75	0.25	187.45	0.04 %	58.62	7.18
0.05	0.75	0.5	292.14	0.06 %	58.62	3.62
0.05	0.75	0.95	364.02	0.07 %	58.62	5.97
0.05	0.95	0.25	86.95	0.02 %	58.62	12.62
0.05	0.95	0.5	137.88	0.03 %	58.62	9.33
0.05	0.95	0.75	324.73	0.06 %	58.62	5.97
0.25	0.05	0.5	142.27	0.03 %	31.13	6.61
0.25	0.05	0.75	138.12	0.03 %	31.13	8.72
0.25	0.05	0.95	113.98	0.02 %	31.13	11.39
0.25	0.5	0.05	257.77	0.05 %	31.13	6.61
0.25	0.5	0.75	547.83	0.11 %	31.13	2.83
0.25	0.5	0.95	243.72	0.05 %	31.13	6.61
0.25	0.75	0.05	142.64	0.03 %	31.13	8.72
0.25	0.75	0.5	385.05	0.08 %	31.13	2.83
0.25	0.75	0.95	410.31	0.08 %	31.13	4.43
0.25	0.95	0.05	109.12	0.02 %	31.13	11.39
0.25	0.95	0.5	217.62	0.04 %	31.13	6.61
0.25	0.95	0.75	336.01	0.07 %	31.13	4.43
0.5	0.05	0.25	459.04	0.09 %	17.58	2.82
0.5	0.05	0.75	180.90	0.04 %	17.58	5.35
0.5	0.05	0.95	148.58	0.03 %	17.58	6.74
0.5	0.25	0.05	521.59	0.10 %	17.58	2.82
0.5	0.25	0.75	349.93	0.07 %	17.58	3.55
0.5	0.25	0.95	224.38	0.04 %	17.58	5.35
0.5	0.75	0.05	246.91	0.05 %	17.58	5.35
0.5	0.75	0.25	377.73	0.08 %	17.58	3.55
0.5	0.75	0.95	431.95	0.09 %	17.58	2.82
0.5	0.95	0.05	145.93	0.03 %	17.58	6.74
0.5	0.95	0.25	181.11	0.04 %	17.58	5.35
0.5	0.95	0.75	348.57	0.07 %	17.58	2.82
0.75	0.05	0.25	1508.08	0.30 %	8.17	1.37
0.75	0.05	0.5	544.75	0.11 %	8.17	1.98
0.75	0.05	0.95	328.58	0.07 %	8.17	3.18
0.75	0.25	0.05	1428.36	0.29 %	8.17	1.37
0.75	0.25	0.5	1000.17	0.20 %	8.17	0.93
0.75	0.25	0.95	643.87	0.13 %	8.17	2.57
0.75	0.5	0.05	644.72	0.13 %	8.17	1.98
0.75	0.5	0.25	976.63	0.20 %	8.17	0.93
0.75	0.5	0.95	598.64	0.12 %	8.17	1.98
0.75	0.95	0.05	511.23	0.10 %	8.17	3.18
0.75	0.95	0.25	531.85	0.11 %	8.17	2.57
0.75	0.95	0.5	506.28	0.10 %	8.17	1.98
0.95	0.05	0.25	6126.11	1.23 %	1.6	0.27
0.95	0.05	0.5	2429.59	0.49 %	1.6	0.39
0.95	0.05	0.75	2614.86	0.52 %	1.6	0.51
0.95	0.25	0.05	5984.93	1.20 %	1.6	0.27
0.95	0.25	0.5	6449.31	1.29 %	1.6	0.19
0.95	0.25	0.75	3368.70	0.67 %	1.6	0.35
0.95	0.5	0.05	2549.66	0.51 %	1.6	0.39
0.95	0.5	0.25	5477.86	1.10 %	1.6	0.19
0.95	0.5	0.75	8233.22	1.65 %	1.6	0.19
0.95	0.75	0.05	1746.03	0.35 %	1.6	0.51
0.95	0.75	0.25	2855.46	0.57 %	1.6	0.35
0.95	0.75	0.5	4452.01	0.89 %	1.6	0.19

Tabla 5.15

Métricas de error y resultados de privacidad obtenidos tras las sucesivas ejecuciones del algoritmo RAPPOR, variando los parámetros  $f$ ,  $p$  y  $q$ . ( $m = 128$ ,  $k = 8$ , dataset: exp\_distrib\_500k)

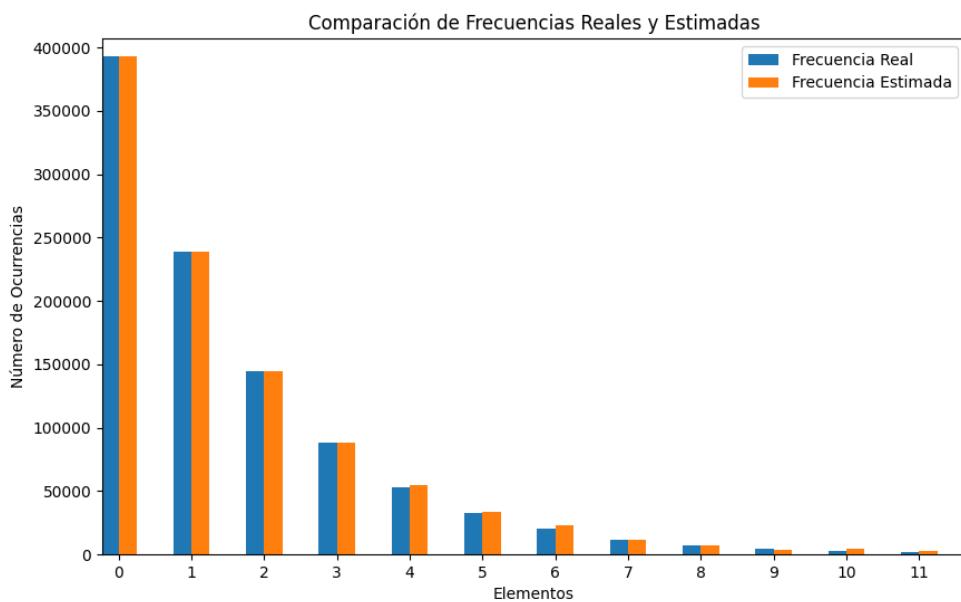


Figura 5.9: Comparativa entre los resultados obtenidos del algoritmo RAPPOR para los elementos de mayor frecuencia real.

Por este motivo, y de cara a realizar una estimación precisa, es recomendable conocer el dominio subyacente a los datos y seleccionar un subconjunto de estudio apropiado. Formado por elementos con un nivel de ocurrencia alto en el flujo original, en lugar del dominio completo.

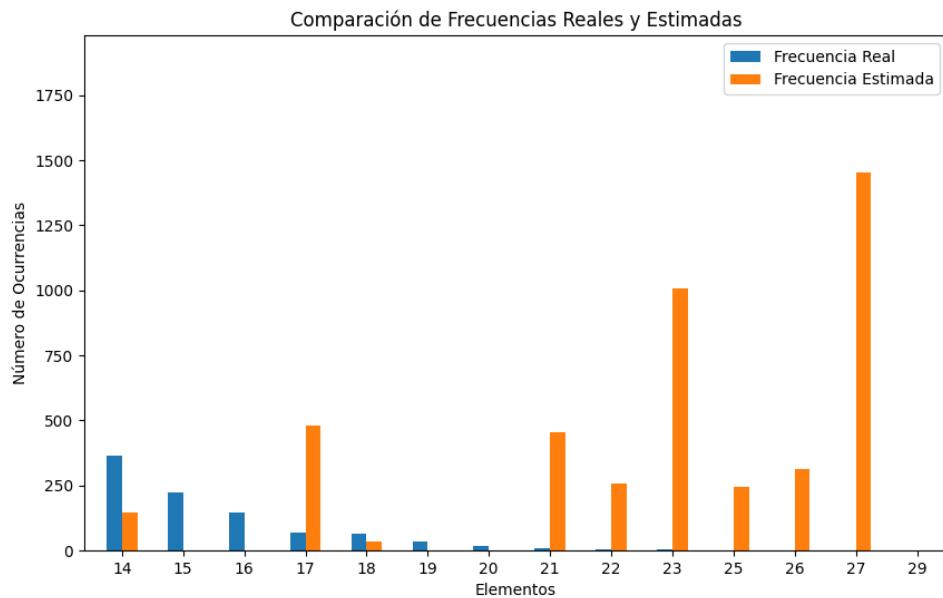


Figura 5.10: Comparativa entre los resultados obtenidos del algoritmo RAPPOR para los elementos de menor frecuencia real

#### 5.5.4. Experimento 4. Evaluación del rendimiento del algoritmo dBitFlip

El algoritmo dBitFlip corresponde al último de los cinco algoritmos de estudio. Este experimento se realiza con la finalidad de evaluar el rendimiento del algoritmo, descrito en 4.5, en términos de eficiencia temporal y precisión. Para una primera parte del experimento se ha empleado el dataset **exp\_distrib\_500k**. El resto del experimento se ha realizado con el conjunto de datasets correspondientes a una distribución normal, pero de tamaño variado, **norm\_distrib\_50**, **norm\_distrib\_300**, **norm\_distrib\_750**, **norm\_distrib\_900**, **norm\_distrib\_2500**, **norm\_distrib\_15k**, **norm\_distrib\_60k**, **norm\_distrib\_350**, **norm\_distrib\_1M**, **norm\_distrib\_2\_4M**, al igual que en dos de los experimentos anteriores.

En una primera fase se ha realizado un análisis del impacto de los parámetros de entrada,  $d$  y  $\varepsilon$ , en el tiempo de ejecución del algoritmo y precisión de los resultados. El parámetro  $d$  corresponde con el número de elementos del dominio sobre los que los clientes enviarán información al servidor, independientemente del valor real a enviar. Por otro lado,  $\varepsilon$  corresponde con el grado de privacidad de los reportes enviados por los clientes.

La Tabla 5.16 muestra el coste temporal del algoritmo en función de diferentes configuraciones de parámetros de entrada posibles. Puede apreciarse que el tiempo necesario para privatizar el dato de un cliente depende directamente del valor que toma  $d$ , en esencia  $O(d)$ . Este algoritmo cuenta con uno de los mejores tiempos para la etapa de enmascaramiento comentados hasta el momento. Esto se debe a que su implementación es de bajo costo computacional, con su correspondiente impacto en la precisión de los resultados y pérdida de privacidad asociada. El tiempo requerido por el algoritmo para estimar el histograma depende del número de paquetes privatizados,  $N$ , y de la dimensión de estos,  $d$ . Como en este caso se ha usado el mismo dataset para todas las ejecuciones, se aprecia el impacto directo del parámetro  $d$  en esta métrica.

La Tabla 5.17 recoge métricas acerca de la precisión de los resultados, obtenidas de las ejecuciones del algoritmo dBitFlip descritas anteriormente. Cuando  $\varepsilon$  toma valores menores de 0,5, la precisión cae notablemente, aunque incrementar  $d$  puede mitigar el impacto de dicho parámetro. Un valor de  $\varepsilon$  de esa magnitud implica un alto nivel en la privacidad de la información. No obstante, una media de errores alta, acompañada de un coeficiente de correlación de Pearson cercano a 0, es síntoma de una estimación errónea y que no representa la realidad. Los mejores resultados en balance privacidad-utilidad, se alcanzan para valores de  $\varepsilon$  en el intervalo [1, 2]. A partir del incremento de este parámetro, la precisión de los resultados mejora sustancialmente, a la vez que la privacidad va desapareciendo. Este es un algoritmo que lleva una pérdida de privacidad asociada al parámetro  $d$ , seleccionando valores altos de dicho parámetro y realizando múltiples envíos desde un mismo cliente. Además, el parámetro  $d$  impacta directamente en los tiempos de ejecución del algoritmo, y depende del tamaño del dominio de estudio, que en ocasiones puede ser desconocido. Por lo que no es recomendable escoger valores altos, superiores a 8, para este algoritmo.

La Figuras 5.11 y 5.12 muestran de forma gráfica que pese a que al aumentar el parámetro  $d$  la métrica de error disminuye, la curvatura descrita se ve acotada supe-

PARÁMETROS		COSTE TEMPORAL	
d	$\varepsilon$	Cliente (Por usuario)	Servidor (Estimar frecuencias)
1	0.05	0.0013 ms	164.4199 ms
2	0.05	0.0017 ms	307.0028 ms
4	0.05	0.0027 ms	582.9020 ms
8	0.05	0.0050 ms	1130.2509 ms
12	0.05	0.0070 ms	1680.3532 ms
16	0.05	0.0088 ms	2170.4540 ms
24	0.05	0.0127 ms	3243.9859 ms
1	0.5	0.0013 ms	163.7101 ms
2	0.5	0.0017 ms	303.6048 ms
4	0.5	0.0027 ms	574.1460 ms
8	0.5	0.0050 ms	1096.0798 ms
12	0.5	0.0071 ms	1629.2729 ms
16	0.5	0.0088 ms	2211.2291 ms
24	0.5	0.0131 ms	3386.0788 ms
1	1	0.0013 ms	165.2882 ms
2	1	0.0017 ms	302.0689 ms
4	1	0.0027 ms	572.3050 ms
8	1	0.0050 ms	1111.5940 ms
12	1	0.0070 ms	1623.2800 ms
16	1	0.0088 ms	2182.7400 ms
24	1	0.0127 ms	3219.0211 ms
1	2	0.0013 ms	163.2888 ms
2	2	0.0017 ms	307.4191 ms
4	2	0.0026 ms	567.5032 ms
8	2	0.0050 ms	1114.6109 ms
12	2	0.0070 ms	1613.8561 ms
16	2	0.0087 ms	2135.5691 ms
24	2	0.0127 ms	3223.1631 ms
1	4	0.0012 ms	161.2287 ms
2	4	0.0017 ms	302.7751 ms
4	4	0.0026 ms	574.3780 ms
8	4	0.0049 ms	1111.9721 ms
12	4	0.0069 ms	1614.5751 ms
16	4	0.0087 ms	2148.3901 ms
24	4	0.0126 ms	3297.9131 ms
1	8	0.0012 ms	162.6780 ms
2	8	0.0017 ms	301.5459 ms
4	8	0.0026 ms	565.9142 ms
8	8	0.0049 ms	1101.3517 ms
12	8	0.0069 ms	1638.6259 ms
16	8	0.0086 ms	2150.8639 ms
24	8	0.0125 ms	3187.8390 ms
1	16	0.0013 ms	161.3371 ms
2	16	0.0017 ms	310.6091 ms
4	16	0.0026 ms	571.9481 ms
8	16	0.0049 ms	1115.6809 ms
12	16	0.0069 ms	1604.5542 ms
16	16	0.0087 ms	2159.6799 ms
24	16	0.0124 ms	3190.5758 ms
1	32	0.0013 ms	161.4542 ms
2	32	0.0017 ms	307.3800 ms
4	32	0.0026 ms	560.6258 ms
8	32	0.0049 ms	1103.2040 ms
12	32	0.0069 ms	1634.1581 ms
16	32	0.0086 ms	2124.9831 ms
24	32	0.0126 ms	3254.9820 ms

Tabla 5.16

Resultados obtenidos, en términos de coste temporal, tras la ejecución del algoritmo dBitFlip según los parámetros d y  $\varepsilon$ . (dataset: exp\_distrib\_500k)

PARÁMETROS		MÉTRICAS DE ERROR						
d	$\varepsilon$	Media Errores	Error Porcentual	MSE	RMSE	MSE (Normalizado)	RMSE (Normalizado)	Coeficiente Correlación Pearson
1	0.05	146831.04	29.37 %	30827359339.80	175577.22	156679.71	395.83	0.1853
2	0.05	79551.30	15.91 %	9441644276.33	97168.12	47987.05	219.06	0.4823
4	0.05	52283.46	10.46 %	4043140073.66	63585.69	20549.21	143.35	0.6024
8	0.05	42312.46	8.46 %	3318877823.88	57609.70	16868.16	129.88	0.4881
12	0.05	43476.74	8.70 %	2743601015.94	52379.39	13944.32	118.09	0.5738
16	0.05	28178.92	5.64 %	1183081461.21	34395.95	6013.00	77.54	0.5686
24	0.05	22912.77	4.58 %	704989389.10	26551.64	3583.10	59.86	0.5909
1	0.5	11238.77	2.25 %	193240806.96	13901.11	982.14	31.34	0.7123
2	0.5	6595.06	1.32 %	55521966.52	7451.31	282.19	16.80	0.7802
4	0.5	5011.20	1.00 %	43069595.82	6562.74	218.90	14.80	0.8445
8	0.5	4344.02	0.87 %	25765919.66	5076.01	130.95	11.44	0.8069
12	0.5	4236.08	0.85 %	27780038.24	5270.68	141.19	11.88	0.8047
16	0.5	3062.61	0.61 %	12207655.04	3493.95	62.05	7.88	0.8132
24	0.5	2320.50	0.46 %	8462973.35	2909.12	43.01	6.56	0.8105
1	1	6873.57	1.37 %	64570046.13	8035.55	328.18	18.12	0.7992
2	1	4902.77	0.98 %	36446221.85	6037.07	185.24	13.61	0.8173
4	1	2391.23	0.48 %	8982912.95	2997.15	45.66	6.76	0.8089
8	1	1160.76	0.23 %	2143507.61	1464.07	10.89	3.30	0.8187
12	1	1545.87	0.31 %	4162229.69	2040.15	21.15	4.60	0.8268
16	1	1415.08	0.28 %	3119701.97	1766.27	15.86	3.98	0.8217
24	1	1047.65	0.21 %	1673609.06	1293.68	8.51	2.92	0.8140
1	2	3066.24	0.61 %	13756709.89	3709.00	69.92	8.36	0.8185
2	2	2323.71	0.46 %	7899569.67	2810.62	40.15	6.34	0.8154
4	2	1427.10	0.29 %	3687136.04	1920.19	18.74	4.33	0.8186
8	2	674.82	0.13 %	604449.76	777.46	3.07	1.75	0.8192
12	2	1007.56	0.20 %	1405393.36	1185.49	7.14	2.67	0.8200
16	2	642.60	0.13 %	659006.45	811.79	3.35	1.83	0.8168
24	2	540.36	0.11 %	410353.89	640.59	2.09	1.44	0.8187
1	4	1624.41	0.32 %	3859122.16	1964.46	19.61	4.43	0.8179
2	4	978.95	0.20 %	2124153.98	1457.45	10.80	3.29	0.8252
4	4	635.03	0.13 %	590340.40	768.34	3.00	1.73	0.8150
8	4	448.01	0.09 %	374191.76	611.71	1.90	1.38	0.8157
12	4	462.53	0.09 %	342287.96	585.05	1.74	1.32	0.8162
16	4	406.79	0.08 %	240433.91	490.34	1.22	1.11	0.8187
24	4	267.95	0.05 %	91830.85	303.04	0.47	0.68	0.8171
1	8	617.97	0.12 %	952979.99	976.21	4.84	2.20	0.8148
2	8	427.93	0.09 %	393571.18	627.35	2.00	1.41	0.8200
4	8	214.43	0.04 %	75556.31	274.88	0.38	0.62	0.8196
8	8	211.00	0.04 %	74828.57	273.55	0.38	0.62	0.8181
12	8	119.98	0.02 %	39661.52	199.15	0.20	0.45	0.8180
16	8	111.98	0.02 %	19359.87	139.14	0.10	0.31	0.8184
24	8	101.95	0.02 %	14924.33	122.17	0.08	0.28	0.8184
1	16	271.12	0.05 %	233830.72	483.56	1.19	1.09	0.8193
2	16	196.36	0.04 %	113576.90	337.01	0.58	0.76	0.8186
4	16	154.36	0.03 %	92920.85	304.83	0.47	0.69	0.8187
8	16	88.07	0.02 %	29825.80	172.70	0.15	0.39	0.8182
12	16	50.49	0.01 %	10621.16	103.06	0.05	0.23	0.8187
16	16	46.02	0.01 %	6855.38	82.80	0.03	0.19	0.8178
24	16	17.68	0.01 %	886.96	29.78	0.01	0.07	0.8183
1	32	176.55	0.04 %	152253.90	390.20	0.77	0.88	0.8191
2	32	252.86	0.05 %	304131.96	551.48	1.55	1.24	0.8179
4	32	144.57	0.03 %	81790.04	285.99	0.42	0.64	0.8191
8	32	97.15	0.02 %	58083.80	241.01	0.30	0.54	0.8175
12	32	67.56	0.01 %	31493.89	177.47	0.16	0.40	0.8190
16	32	66.37	0.01 %	18399.73	135.65	0.09	0.31	0.8177
24	32	22.21	0.01 %	1730.67	41.60	0.01	0.09	0.8182

Tabla 5.17

Resultados obtenidos, en términos de error, tras la ejecución del algoritmo dBitFlip según los parámetros  $d$  y  $\varepsilon$ . (dataset: exp\_distrib\_500k)

riormente por aquellas con valor de  $\varepsilon$  menor. Este hecho tiende a difuminarse a medida que el valor de  $\varepsilon$  aumenta notablemente, Figura 5.12, donde las curvas se entrelazan, mostrando una cantidad de error mínima, junto con unos niveles de privacidad prácticamente inexistentes.

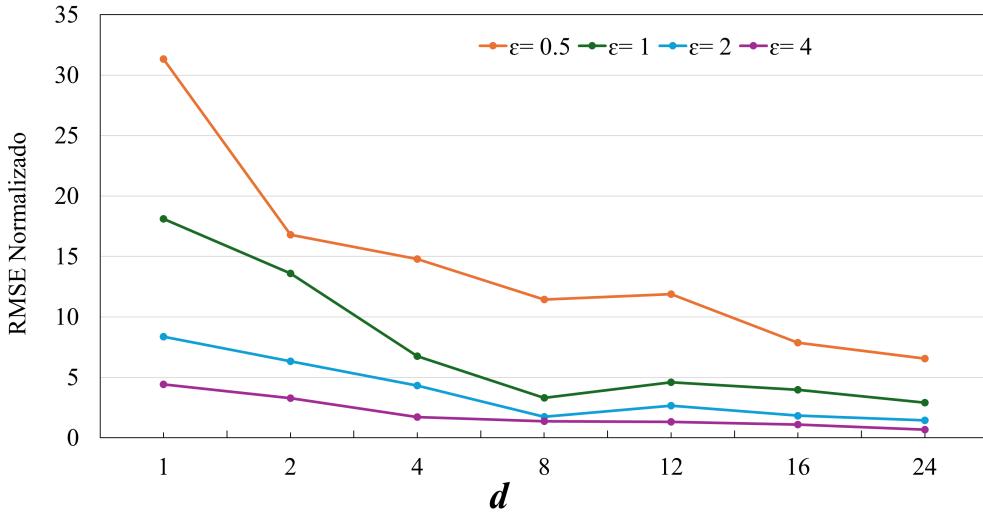


Figura 5.11: Curvatura descrita por el RMSE normalizado obtenido en el algoritmo dBitFlip, en función de los parámetros  $d$  y  $\varepsilon$  (Valores de  $\varepsilon$  reducidos).

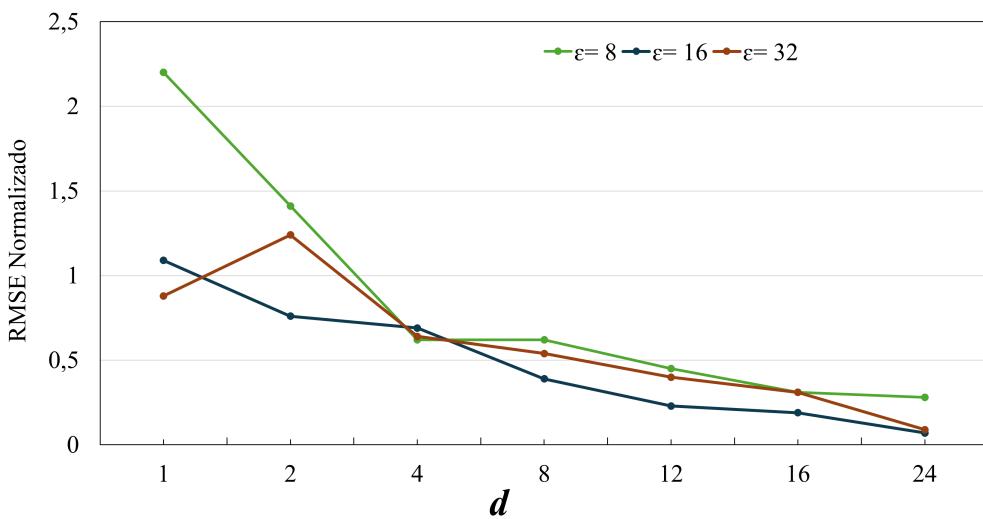


Figura 5.12: Curvatura descrita por el RMSE normalizado obtenido en el algoritmo dBitFlip, en función de los parámetros  $d$  y  $\varepsilon$  (Valores de  $\varepsilon$  altos).

En ultima instancia, se han realizado sucesivas ejecuciones del algoritmo fijando los parámetros  $d = 4$  y  $\varepsilon = 1$ , variando el número de registros de los datasets empleados. Se ha utilizado el conjunto de dataset formado por **norm\_distrib\_50**, **norm\_distrib\_300**, **norm\_distrib\_750**, **norm\_distrib\_900**, **norm\_distrib\_2500**,

**norm\_distrib\_15k, norm\_distrib\_60k, norm\_distrib\_350k, norm\_distrib\_1M** y **norm\_distrib\_2.4M**. Como se comentó anteriormente, el tiempo necesario para estimar el histograma depende directamente de los parámetros  $d$  y  $N$ . En definitiva, el proceso consiste en ir desenmascarando los diferentes paquetes privatizados, incrementando el valor de una serie de contadores en base a unas reglas. Esto dependerá de la implementación y se podría proponer un enfoque en el que a medida que llegue un nuevo dato privatizado, se actualicen los contadores pertinentes. A este tiempo se le añadiría el cálculo final de la estimación en base al valor de los contadores, cuyo coste temporal es despreciable. De la Tabla 5.18 se puede extraer el tiempo aproximado para actualizar los contadores debido a la llegada de un nuevo dato,  $t_{cont}$ . Teniendo en cuenta que este valor depende directamente de  $m$ , se puede concluir que  $t_{cont} \approx 0,00124$  milisegundos, cuando  $d = 4$ .

DATASET	COSTE TEMPORAL	ERROR
N	Servidor (Estimar frecuencias)	Error Porcentual
50	0.0610 ms	37.11 %
300	0.5691 ms	19.32 %
750	0.8571 ms	13.66 %
900	1.0812 ms	12.82 %
2500	2.8019 ms	5.17 %
15000	17.5188 ms	1.99 %
60000	67.1439 ms	1.08 %
200000	226.4423 ms	0.86 %
350000	432.4000 ms	0.51 %
1000000	1144.1531 ms	0.30 %
2400000	2733.6490 ms	0.24 %

Tabla 5.18

*Impacto del número de registros del dataset empleado en los resultados del algoritmo dBitFlip. ( $d = 4$ ,  $\varepsilon = 1$ , dataset: norm\_distrib\_N)*

Respecto a la precisión de los resultados, se cumple una característica encontrada en todos los métodos descritos hasta el momento. A medida que el número de registros aumenta, el algoritmo puede suavizar mejor el impacto del ruido y mejorar la precisión de las estimaciones. Si profundizamos en la fórmula encargada de calcular la frecuencia estimada de un elemento:

$$\tilde{h}(x) = \frac{|D|}{Nd} \cdot \text{Contadores}(x) \quad (5.1)$$

Siendo  $D$  el dominio de estudio. Aunque los contadores individuales están sujetos a ruido debido a las técnicas de privatización empleadas, un mayor número de registros permite que el efecto del ruido en la estimación final se reduzca. A medida que  $N$  crece, la forma en que el ruido es distribuido sobre un mayor número de registros puede afectar la precisión, pero en términos generales, obteniendo resultados más prácticos. El efecto de la privatización tiende a ser más manejable con datasets más grandes, siempre y cuando el ajuste de parámetros se haga adecuadamente.

# 6. Conclusiones

---

A lo largo del capítulo 5, se han obtenido conclusiones acerca de los experimentos realizados, con la finalidad de evaluar el rendimiento de cada técnica e intentar realizar una comparativa entre estas, aunque ya se mencionó la dificultad para realizar esta comparativa. No obstante, existen algunos aspectos que resultan interesante destacar en el presente capítulo. Estas conclusiones, abordan temas no comentados, o descritos con suficiente detalle, hasta el momento.

## Reducción de la complejidad espacial frente a métodos tradicionales.

Durante todo el proyecto, se ha expuesto el hecho de que estos algoritmos innovadores son sustancialmente mejores, en términos de ahorro espacial, respecto a las estructuras de datos tradicionales. A continuación, y con la finalidad de reforzar esta idea, se propone una demostración, o ejemplo teórico con datos reales, de esto.

Se estima que la popular empresa Amazon ofrece más de 350 millones de productos en total, aunque este número varía y puede ser mayor en función de la expansión continua de su inventario y la inclusión de nuevos productos. Imagínese que se desea mantener un conteo aproximado del número de veces que se compra cada producto, para poder analizar los productos más vendidos, tendencias crecientes, o descubrir y eliminar del mercado aquellos productos menos solicitados.

Si se utilizara una estructura de datos tradicional, como un vector de enteros, el espacio requerido dependerá directamente del número de productos disponibles, un número masivo y en constante crecimiento. Supongamos que implementamos una solución en el lenguaje Python, donde los números enteros (*int*), ocupan un espacio inicial de 4 bytes, aunque puede aumentar a medida que se aumenta el rango de valores. En estas condiciones, la estructura de datos empleada ocupará,

$$4 \text{ bytes} \times 350 \text{ Millones de productos} \sim 1,3 \text{ GB.} \quad (6.1)$$

La opción contraria es utilizar una estructura de datos probabilística de las descritas a lo largo del proyecto, por ejemplo, el algoritmo Private Count Mean Sketch, sección 4.1. Fijando un valor para los parámetros de entrada como los empleados en casos reales por la empresa Apple [15],  $k = 65536$ ,  $m = 1024$  y  $\varepsilon = 4$ , se obtendría una estructura de dimensiones  $k \times m$ , cuyo espacio requerido, en las mismas condiciones que antes, será,

$$4 \text{ bytes} \times k \times m \sim 256 \text{ MB.} \quad (6.2)$$

Este resultado indica una reducción del espacio requerido en un factor aproximado de  $\frac{1,3}{0,25} = 5,2$ . Esta comparativa carecería de sentido sin destacar el hecho de que la segunda estructura de datos empleada, lleva consigo una perdida en la precisión de los resultados, la cual deriva de la Ecuación 4.2. Este límite en la precisión de los resultados dependerá de numerosos factores, tales como la distribución de los datos reales, la cantidad de registros procesados o los parámetros seleccionados. Además el resultado de

esta ecuación depende, en gran medida, del parámetro de privacidad  $\varepsilon$ . Si aumentamos este parámetro, la privacidad asociada disminuiría, asemejándose a una situación en la que en un servidor, donde se mantiene una estructura de datos tradicional, van llegando los datos sin privatizar. En ese caso, la varianza de la estimación frente al valor real se reduciría notablemente.

Uno de los principales beneficios que comparten las técnicas descritas, es la capacidad para preservar la privacidad de los datos. Apoyándose en el marco de la privacidad  $\varepsilon$ -diferencial para asegurar que las estadísticas obtenidas no revelen información sensible sobre los datos individuales. Por lo general, las soluciones tradicionales no están diseñadas para abordar estos problemas de privacidad. No obstante, implementar estas técnicas en casos concretos puede ser complejo, especialmente cuando se trata de garantizar el cumplimiento de la privacidad  $\varepsilon$ -diferencial y ajustar los parámetros para obtener un equilibrio entre precisión y coste espacial y temporal. En flujos de datos aún más grandes, la ventaja de espacio de estas estructuras compactas frente a las tradicionales se vuelve más pronunciada, ya que su tamaño no aumenta con el número de elementos únicos, a diferencia de, por ejemplo, una tabla hash. Por lo que la escalabilidad se considera una ventaja notable de estas técnicas.

La precisión en las estimaciones devueltas por estos algoritmos no será tan alta como la que se obtiene con soluciones tradicionales, las cuales pueden usar más recursos computacionales para proporcionar resultados exactos. En conclusión, siempre es importante tener presente la finalidad de la solución que se pretende desarrollar. Quizás sea recomendable decantarse por estas técnicas de consultas aproximadas, que pese a devolver resultados con cierto margen de error, reducen sustancialmente el espacio ocupado por estructuras tradicionales y son más escalables.

### **Metodologías para ajustar $\varepsilon$ y su impacto en la utilidad y privacidad de los datos.**

En el contexto de la privacidad  $\varepsilon$ -diferencial, el parámetro  $\varepsilon$  es fundamental para balancear la privacidad y la utilidad de los datos. Una mayor privacidad en los datos se puede obtener con un valor bajo de  $\varepsilon$ . La limitación de la cantidad de información que se puede inferir sobre cualquier registro individual se debe a que la relación entre las probabilidades de los resultados en múltiples bases de datos se mantiene muy cercana. No obstante, a medida que  $\varepsilon$  disminuye, se debe agregar mayor cantidad de ruido para cumplir con los requisitos de privacidad, lo que puede deteriorar la calidad y precisión de los datos. Esto puede provocar que las respuestas sean inexactas y, se convierta en una solución inútil para muchos fines prácticos. Este balance entre privacidad y utilidad se obtiene de la Ecuación 2.5, y se ha recalado a lo largo del documento.

Sin embargo, el valor óptimo de  $\varepsilon$  en el marco de la privacidad  $\varepsilon$ -diferencial no es universalmente fijo y varía según el contexto, los requisitos de privacidad, y la utilidad esperada de los datos. Existen varias estrategias y enfoques para determinar un valor adecuado de dicho parámetro. En primer lugar, hay que tener en cuenta ciertas consideraciones contextuales, según la sensibilidad de la información con la que se trabaja y el nivel de precisión esperado en el análisis. La sensibilidad de los datos juega un papel crucial en la determinación del valor de  $\varepsilon$ . Datos altamente sensibles, como información

médica o financiera, requieren valores más bajos de  $\varepsilon$  para asegurar una alta protección de la privacidad. En contraposición, datos menos sensibles, como los emoticonos enviados por los usuarios de una red social o los productos comprados en una tienda en línea, pueden tolerar valores mayores de dicho parámetro. Por otro lado, la utilidad de la información dependerá de la precisión de las respuestas obtenidas. Aquellos casos en los que se requiera alta precisión, como en estudios médicos o análisis financieros detallados, serán necesario valores de  $\varepsilon$  mayores, siempre y cuando, la privacidad aún se considere aceptable [27].

Existen métodos empíricos que examinan cómo parámetros externos influyen en el equilibrio entre utilidad y privacidad, los cuales pueden ser muy beneficiosos para ajustar el parámetro  $\varepsilon$  en casos reales. Por ejemplo, los encargados de diseñar la solución pueden analizar una variedad de valores de  $\varepsilon$  y evaluar cómo estos influyen en la precisión de los resultados. Siguiendo este procedimiento se puede encontrar un valor de  $\varepsilon$  que cumpla con los requisitos de privacidad y mantenga la utilidad de los datos para análisis significativos.

Otro enfoque consiste en el uso de estrategias adaptativas para ajustar  $\varepsilon$ . Estas estrategias ofrecen un enfoque dinámico que responde a las variaciones en el tipo de consulta o al variar el contexto. En un supuesto sistema de consulta desarrollado, se puede utilizar un valor de  $\varepsilon$  más pequeño para consultas particularmente sensibles, mientras que consultas menos críticas pueden manejarse con un  $\varepsilon$  mayor. Este ajuste dinámico permite a los sistemas mantener un equilibrio flexible entre privacidad y utilidad. Otra estrategia similar se basa en los modelos de privacidad compuestos [17]. Estos modelos combinan diferentes mecanismos de privacidad, ajustando  $\varepsilon$  según la contribución de cada mecanismo a la protección de los datos. Este enfoque permite encontrar un equilibrio más preciso entre privacidad y utilidad, adaptando la protección a los requisitos específicos de cada situación. Sin embargo, estas últimas estrategias descritas aumentan notablemente la complejidad del diseño e implementación de la solución.

## Cumplimiento de estándares de ACM para artefactos de investigación.

La ACM o *Association for Computing Machinery* es una de las principales organizaciones internacionales en el campo de la informática, que publica una gran cantidad de revistas científicas y organiza conferencias. En una de sus principales líneas de trabajo, la ACM se enfoca en promover la calidad y la integridad en la investigación científica. En los últimos años, la ACM ha implementado un sistema de insignias [7] para mejorar la transparencia y la reproducibilidad en la investigación. Estas insignias indican el grado en que los artículos de investigación cumplen con ciertos estándares en cuanto a la revisión y disponibilidad de los artefactos asociados y la validación de los resultados. En el contexto de la investigación científica, se considera un artefacto a cualquier elemento digital que se utiliza en la investigación para generar o analizar los resultados. Todo el código fuente, librerías, datasets, scripts de automatización, etc., se consideran artefactos.

A continuación se exponen brevemente las diferentes insignias propuestas por ACM. Estos distintivos se consideran independientes y cualquiera de ellos pueden apli-

carse a un artículo dado dependiendo de los procedimientos de revisión desarrollados por la revista o conferencia. Las tres insignias principales son las siguientes:

- **Artefactos Evaluados:** Se otorga cuando los artefactos relacionados con el estudio han pasado una auditoría independiente. Hay dos niveles:
  - **Funcional:** Los artefactos están documentados, son consistentes, completos y operables.
  - **Reutilizable:** Los artefactos son de alta calidad, bien documentados y estructurados para facilitar la reutilización.
- **Artefactos Disponibles:** Se concede cuando los artefactos creados por los autores están disponibles públicamente en un repositorio accesible, permitiendo su recuperación y uso por otros investigadores.
- **Resultados Validados:** Se aplica cuando los resultados principales del artículo han sido obtenidos con éxito por un equipo diferente al de los autores. Hay dos niveles:
  - **Reproducidos:** Los resultados se obtienen usando los artefactos proporcionados por los autores.
  - **Replicados:** Los resultados se obtienen independientemente sin usar artefactos proporcionados por los autores.

Respecto al presente proyecto, todo el código fuente del algoritmo, así como los datasets implicados en el análisis, se encuentra disponibles públicamente en un repositorio de GitHub bajo el siguiente enlace: <https://github.com/Antonio-Requena/DP-Sketching-Algorithms>. Este hecho verifica el cumplimiento de la insignia de *Artefactos Disponibles*, Figura 6.1. Esto asegura que otros investigadores puedan acceder a los artefactos utilizados en la presente investigación y utilizarlos para reproducir los experimentos y verificar los resultados.



Figura 6.1: Insignia de *Artefactos Disponibles* que indica que el código está disponible públicamente en un repositorio accesible. Véase la fuente en [7].

Por otro lado, la implementación y la documentación disponible en el repositorio permiten la reproducción de los experimentos presentados en el artículo. Esto significa que otros usuarios pueden utilizar el código y los artefactos disponibles para verificar que los resultados presentados se pueden replicar con éxito. El usuario tendrá la posibilidad de clonar el repositorio y configurar el entorno al completo. Siguiendo las

instrucciones proporcionadas a lo largo del repositorio, el usuario podrá ejecutar los diferentes algoritmos. Esto posibilita la reproducción de los experimentos y la generación de resultados similares a los descritos en el documento. Por este motivo, el presente proyecto cumple con la insignia de *Resultados Reproducidos*, Figura 6.2.



Figura 6.2: Insignia de *Resultados Reproducidos* que muestra que el código permite la reproducción de los resultados presentados en el artículo. Véase la fuente en [7].

En todo caso, no se requiere ni se espera una replicación exacta o reproducción de los resultados. En su lugar, los resultados deben estar de acuerdo dentro de una tolerancia considerada aceptable para experimentos del tipo dado. No obstante, las diferencias en los resultados no deberían cambiar las principales afirmaciones hechas en el artículo. Es importante destacar que los algoritmos presentados en este proyecto se caracterizan por incorporar cierto grado de aleatoriedad derivado del uso de funciones hash y probabilidades, fundamentadas en la generación de números aleatorios. Esto puede alterar un poco los resultados de una ejecución a otra, pero los resultados que se obtendrán reflejarán, en esencia, el análisis descrito.

## 6.1. Objetivos alcanzados

A lo largo del proyecto, se ha investigado en profundidad la literatura existente sobre privacidad  $\epsilon$ -diferencial y otros conceptos y técnicas claves en el área de la protección de datos, incluyendo los fundamentos matemáticos. Se han analizado diversas técnicas de sketching para la estimación de frecuencias y cardinalidad, y su combinación con privacidad  $\epsilon$ -diferencial local. Se han analizado las propiedades de anonimato en las interacciones entre la parte cliente y servidor de dichos algoritmos. En todo momento, se ha elaborado una explicación detallada de los conceptos clave, acompañada de imágenes y ejemplos propios para facilitar la comprensión y proporcionar una base sólida de conocimiento.

Respecto a la implementación de los algoritmos seleccionados, por un lado, estos se trajeron de fuentes diferentes y se han presentado en un formato común de pseudocódigo. Esto ha permitido una uniformidad en los algoritmos dentro del contexto del proyecto. Cada algoritmo se ha documentado exhaustivamente, proporcionando una explicación clara de su funcionamiento. En base al pseudocódigo presentado, los algoritmos se han implementado exitosamente en el lenguaje de programación Python, permitiendo una sencilla ejecución por parte de usuarios externos.

Por último, se han realizado una serie de experimentos, cuyos resultados han sido presentados y comentados en el documento. Esto ha permitido evaluar las diferentes técnicas implementadas y sacar conclusiones acerca de qué algoritmos son más apropiados para según qué situaciones. La totalidad de estos experimentos es reproducible, siguiendo las directrices del repositorio<sup>1</sup>.

## 6.2. Lecciones aprendidas

Este proyecto me ha llevado a profundizar dentro del área de la privacidad de datos. He logrado aprender numerosos conceptos acerca de este campo, en especial, técnicas de enmascaramiento y modelos de privacidad. He investigado acerca de estructuras de datos compactas, diferentes a las tradicionales, que permiten realizar consultas aproximadas. Todo este trabajo de investigación, me ha permitido mejorar mis habilidades técnicas en materia de privacidad. Por supuesto, he mejorado mi capacidad de resumir información compleja y extensa, extrayendo los puntos clave de manera efectiva. De la misma forma, he tenido que ampliar y aclarar conceptos con pocos recursos bibliográficos.

La elaboración del presente documento me ha permitido perfeccionar la habilidad de presentar la información en un formato común, estructurado y siguiendo unos estándares en el campo de la investigación, logrando una explicación detallada de las técnicas por medio de algoritmos en pseudocódigo, imágenes y ejemplos. Esta capacidad es crucial en el ámbito de la divulgación, donde es esencial transmitir conocimientos técnicos de manera comprensible, y que se propuso inicialmente como objetivo a cumplir.

En último lugar, la parte práctica del proyecto me ha permitido mejorar mis habilidades de programación en Python. Además, me he familiarizado con la creación de entornos virtuales, y con la creación de un repositorio completo y correctamente documentado de GitHub<sup>2</sup>.

## 6.3. Trabajo futuro

Este proyecto me ha hecho dedicarle gran cantidad de tiempo a la investigación en materia de privacidad de datos. No obstante, la cantidad de conceptos, técnicas y algoritmos dentro de este campo es muy amplia, y este proyecto me ha permitido asentar las bases. La investigación en el área de privacidad se encuentra en auge, por lo que se me abren diferentes líneas de trabajo en las que profundizar.

Por un lado, se hace posible la investigación destinada a hacer que los algoritmos presentados sean escalables a volúmenes de datos mayores, garantizando al mismo tiempo que se mantenga un nivel adecuado de privacidad. De la misma forma sería de interés explorar o desarrollar nuevos algoritmos que combinen modelos de privacidad

---

<sup>1</sup>Repositorio de Github - <https://github.com/Antonio-Requena/DP-Sketching-Algorithms>.

<sup>2</sup>GitHub - <https://github.com/>

junto con estructuras de datos probabilísticas. En especial, algoritmos destinados a la estimación de la cardinalidad o, algoritmos que permitan realizar consultas con operadores lógicos. Por supuesto, desarrollar aplicaciones prácticas que incorporen los algoritmos descritos en sectores reales, como la salud, sería una opción muy interesante.

Un área específica de la privacidad cuya investigación es necesaria es la privacidad  $\epsilon$ -diferencial. Es posible investigar acerca de cómo aplicar la privacidad  $\epsilon$ -diferencial a nuevas tecnologías, en especial, me interesaría su combinación con la inteligencia artificial o el Internet de las cosas. También me resulta interesante lo relacionado con la aplicación de la privacidad  $\epsilon$ -diferencial a tipos de datos no estructurados como imágenes y textos, y estudiar la utilidad de estos. Otra línea de investigación podría centrarse en investigar técnicas para ajustar y optimizar los parámetros de privacidad  $\epsilon$ -diferencial, como el parámetro  $\epsilon$ .

Finalmente, una línea de trabajo futuro, que me interesa bastante, tiene relación con la divulgación. Ya sea, una divulgación más académica, de este o futuros proyectos, a través de publicaciones o conferencias. O bien, a través de medios no convencionales, como las redes sociales, sobre diversidad de temas relacionados la informática y la ciberseguridad.

# Bibliografía

---

- [1] Grupo Adaptalia. *Categorías de Datos Personales RGPD*. Jun. de 2022. URL: <https://grupoadaptalia.es/blog/categorias-de-datos-personales-rgpd/>.
- [2] Noga Alon, Yossi Matias y Miklós Szegedy. «The Space Complexity of Approximating the Frequency Moments». En: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 1996, págs. 20-29.
- [3] Guillermo Navarro Arribas. *Privacidad Diferencial: introducción*. Inf. téc. Universidad Autónoma de Barcelona, 2023.
- [4] Z. Bar-Yossef y col. «Counting distinct elements in a data stream». En: *Randomization and Approximation Techniques In Computer Science*. Springer, 2002, págs. 1-10.
- [5] Burton H. Bloom. «Space/Time Trade-offs in Hash Coding with Allowable Errors». En: *Communications of the ACM* 13.7 (1970), págs. 422-426. DOI: [10.1145/362686.362692](https://doi.org/10.1145/362686.362692).
- [6] V. Ciriani y col. «Secure Data Management in Decentralized System». En: Springer Science & Business Media, 2007, págs. 291-355.
- [7] Association for Computing Machinery. *Artifact Review and Badging - Current*. 2020. URL: <https://www.acm.org/publications/policies/artifact-review-and-badging-current> (visitado 08-08-2024).
- [8] Graham Cormode. «AMS Sketch». En: *Encyclopedia of Algorithms*. 2016, págs. 76-78. DOI: [10.1007/978-1-4939-2864-4\578](https://doi.org/10.1007/978-1-4939-2864-4_578).
- [9] Graham Cormode. «Count-Min Sketch». En: *Encyclopedia of Database Systems*. Ed. por LING LIU y M. TAMER ÖZSU. Boston, MA: Springer US, 2009, págs. 511-516. ISBN: 978-0-387-39940-9. DOI: [10.1007/978-0-387-39940-9\\_87](https://doi.org/10.1007/978-0-387-39940-9_87).
- [10] Graham Cormode. «Sketch Techniques for Approximate Query Processing». En: 2010. URL: <https://api.semanticscholar.org/CorpusID:17305954>.
- [11] Graham Cormode y Marios Hadjieleftheriou. «Finding frequent items in data streams». En: *Proc. VLDB Endow.* 1 (2008), págs. 1530-1541. URL: <https://api.semanticscholar.org/CorpusID:15022910>.
- [12] Graham Cormode y col. *Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches*. Now Foundations y Trends, 2011, págs. 201-257. DOI: [10.1561/1900000004](https://doi.org/10.1561/1900000004).
- [13] Anirban Dasgupta y col. *A Framework for Estimating Stream Expression Cardinalities*. 2016. arXiv: [1510.01455 \[cs.DS\]](https://arxiv.org/abs/1510.01455). URL: <https://arxiv.org/abs/1510.01455>.
- [14] D. Defays y M.N. Anwar. «Masking microdata using micro-aggregation». En: *Journal of Official Statistics* (1998), págs. 449-461.
- [15] Apple Differential Privacy Team. «Learning with Privacy at Scale Differential». En: 2017. URL: <https://api.semanticscholar.org/CorpusID:43986173>.

- [16] Bolin Ding, Janardhan Kulkarni y Sergey Yekhanin. *Collecting Telemetry Data Privately*. 2017. arXiv: [1712.01524 \[cs.CR\]](https://arxiv.org/abs/1712.01524). URL: <https://arxiv.org/abs/1712.01524>.
- [17] Irit Dinur y Kobbi Nissim. «Revealing Information While Preserving Privacy». En: *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. San Diego, CA, USA: ACM, 2003, págs. 202-210. DOI: [10.1145/773153.773172](https://doi.org/10.1145/773153.773172).
- [18] Víctor Drummond. *Internet, Privacidad y Datos Personales*. Editorial Reus, 2004.
- [19] C. Dwork y A. Roth. «The Algorithmic Foundations of Differential Privacy». En: *Foundations and Trends in Theoretical Computer Science* 9.3-4 (2014), págs. 211-407.
- [20] Cynthia Dwork y Moni Naor. «Calibrating noise to sensitivity in private data analysis». En: *Theory of cryptography conference* 3876 (2006), págs. 265-284.
- [21] Ú. Erlingsson, V. Pihur y A. Korolova. «RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response». En: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2014.
- [22] G. Fanti, V. Pihur y Ú. Erlingsson. «Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries». En: *Proceedings on Privacy Enhancing Technologies (PoPETS) 2016.3* (2016).
- [23] Philippe Flajolet y G. Nigel Martin. «Probabilistic Counting Algorithms for Data Base Applications». En: *J. Comput. Syst. Sci.* 31 (1985), págs. 182-209. URL: <https://api.semanticscholar.org/CorpusID:46066373>.
- [24] Philippe Flajolet y col. «HyperLogLog: The Analysis of a Near-optimal Cardinality Estimator». En: *Proceedings of the 2007 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2007, págs. 660-669. URL: <https://pubs.siam.org/doi/10.1137/1.9781614338012.61>.
- [25] María Mercedes Rodríguez García. «Semantic Perturbative Privacy-preserving Methods for Nominal Data». Tesis doct. Universitat Rovira i Virgili, 2017.
- [26] GeeksforGeeks. *Flajolet Martin algorithm*. Mar. de 2024. URL: <https://www.geeksforgeeks.org/flajolet-martin-algorithm/>.
- [27] Quan Geng y col. «Privacy and Utility Tradeoff in Approximate Differential Privacy». En: *ArXiv* (2019). URL: <https://arxiv.org/abs/1810.00877>.
- [28] Anco Hindepool y col. «Handbook on Statistical disclosure control». En: ESSNet SDC, 2010, págs. 53-102.
- [29] IBM. *¿Qué es la información de identificación personal (PII)?* Dic. de 2022. URL: [https://www.ibm.com/es-es/topics/pii#:~:text=%C2%BFQu%C3%A9%20es%20la%C2%BAPII%C3%BF,%o%20su%20n%C3%BAmero%20de%20tel%C3%A9fono%20fijo..](https://www.ibm.com/es-es/topics/pii#:~:text=%C2%BFQu%C3%A9%20es%20la%C2%BAPII%C3%BF,%o%20su%20n%C3%BAmero%20de%20tel%C3%A9fono%20fijo.)
- [30] IBM. *Tipos de variables*.  
<https://www.ibm.com/docs/es/spss-statistics/SaaS?topic=charts-variable-types>. Dic. de 2022.
- [31] Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. 5 de dic. de 2018. URL: <https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673>.

- [32] Ninghui Li, Tiancheng Li y Suresh Venkatasubramanian. «t-Closeness: Privacy Beyond k-Anonymity and l-Diversity». En: *2007 IEEE 23rd International Conference on Data Engineering*. 2007, págs. 106-115.
- [33] Ying Li y col. «PrivSketch: A Private Sketch-Based Frequency Estimation Protocol for Data Streams». En: *Database and Expert Systems Applications*. Ed. por Christine Strauss y col. Cham: Springer Nature Switzerland, 2023, págs. 147-163.
- [34] A. Machanavajjhala y col. «L-diversity: privacy beyond k-anonymity». En: *22nd International Conference on Data Engineering (ICDE'06)*. 2006, págs. 24-24. DOI: [10.1109/ICDE.2006.1](https://doi.org/10.1109/ICDE.2006.1).
- [35] Dzejla Medjedovic y Emin Tahirovic. *Algorithms and Data Structures for Massive Datasets*. Capítulos 2-5. Springer, 2023.
- [36] Adam Meyerson y Ryan Williams. «On the Complexity of Optimal K-Anonymity». En: *Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. New York, NY, USA: Association for Computing Machinery, 2004, págs. 223-228.
- [37] Microsoft Corporation. *Differential Privacy for Everyone*. Microsoft Research Technical Report. 2019. URL: <https://www.microsoft.com/en-us/research/publication/differential-privacy-for-everyone/>.
- [38] R.A. Moore. «Controlled data swapping techniques for masking public use microdata sets». En: *Statistical Research Division Report Series* (1996).
- [39] Mingen Pan. «Count-mean Sketch as an Optimized Framework for Frequency Estimation with Local Differential Privacy». En: *ArXiv* abs/2406.03785 (2024). URL: <https://api.semanticscholar.org/CorpusID:270285803>.
- [40] David Pollard. *A User's Guide to Measure Theoretic Probability*. Cambridge: Cambridge University Press, 2002.
- [41] Agencia española de Protección de Datos. *Anonimización y seudonimización (II): la privacidad diferencial*. Blog. Oct. de 2021.
- [42] Python Software Foundation. *Python Programming Language*. 1991. URL: <https://www.python.org/>.
- [43] Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo de 27 de abril de 2016 relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos). 27 de abr. de 2016. URL: <https://eur-lex.europa.eu/eli/reg/2016/679/oj>.
- [44] Lee Rhodes. *Theta Sketch Equations*. Inf. téc. 701 First Ave., Sunnyvale, CA 94089, USA: Yahoo! Inc., sep. de 2015.
- [45] P. Samarati. «Protecting respondents identities in microdata release». En: *IEEE Transactions on Knowledge and Data Engineering* 13.6 (2001), págs. 1010-1027.
- [46] P. Samarati y L. Sweeney. «Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression». En: *Technical Report SRI-CSL-98-04, Computer Science Laboratory, SRI International* (1998).

- [47] Pierangela Samarati. «Protecting Respondents' Privacy in Microdata Release». En: *Proceedings of the 1998 IEEE Symposium on Security and Privacy*. IEEE, 1998, págs. 256-270. DOI: [10.1109/SEC PRI.1998.674235](https://doi.org/10.1109/SEC PRI.1998.674235). URL: <https://ieeexplore.ieee.org/document/674235>.
- [48] Satori. *Data Generalization: The Specifics of Generalizing Data*. <https://satoricyber.com/data-masking/data-generalization/>. 2005.
- [49] Adam Smith, Shuang Song y Abhradeep Guha Thakurta. «The Flajolet-Martin Sketch Itself Preserves Differential Privacy: Private Counting with Minimal Space». En: *Advances in Neural Information Processing Systems*. Ed. por H. Larochelle y col. Vol. 33. Curran Associates, Inc., 2020, págs. 19561-19572.
- [50] Jordi Soria-Comas y Josep Domingo-Ferrer. «Big Data Privacy: Challenges to Privacy Principles and Models». En: *Data Science and Engineering* (2016), págs. 21-28.
- [51] Jordi Soria-Comas y Josep Domingo-Ferrer. «Probabilistic k-anonymity through microaggregation and data swapping». En: *2012 IEEE International Conference on Fuzzy Systems*. 2012, págs. 1-8.
- [52] Patrick Tendick. «Optimal noise addition for preserving confidentiality in multivariate data». En: *Journal of Statistical Planning and Inference* (1991), págs. 341-353.