Versión preliminar de los componentes básicos de MLEDGE

MLEDGE - Aprendizaje automático en la nube y en el borde

(Cloud and Edge Machine Learning)

Diciembre de 2023

Información sobre el entregable

Nombre del documento:

Versión preliminar de los componentes básicos de MLEDGE

Versión actual: 1.0

Proyecto: MLEDGE - Aprendizaje automático en la nube y en el borde (Cloud and Edge Machine Learning)

Paquete de trabajo: P2 - Implementación de componentes básicos de MLEDGE

Tareas: El entregable es resultado del trabajo en los diversos componentes técnicos:

- A2.1: Uso eficiente de Federated Learning (FL) en nubes híbridas y protección contra ataques, incluidos ataques de envenenamiento e inferencia (FedSecure)
- A2.2: Protección de datos sensibles o confidenciales que sean intercambiados entre diferentes dominios administrativos en la nube, incluido su borde que es potencialmente menos seguro (FedWM)
- A2.3: Equidad en términos de distribución de costos y ganancias cuando la computación en el borde se usa para entrenar de forma colaborativa modelos de aprendizaje automático (FLaaS Manager)
- A2.4: Gestión de los desafíos de portabilidad de datos en el borde de la nube (DataEdge)
- A2.5: DevOps y desarrollo continuo para servicios de aprendizaje automático que se ejecutan en borde de la nube (FLaaS)

Entregable: E2.1 - M12 - Requisitos y diseño de la arquitectura y casos de uso (preliminar)

Autores: Santiago Andrés (IMDEA), Javad Dogani (IMDEA), Devriş Işler (IMDEA), Tianyue Chu (IMDEA), Alexander Goultiaev (IMDEA), Naicheng Li (IMDEA)

Revisores: Nikolaos Laoutaris (IMDEA)

Historial de Versiones

Versión	Fecha	Resumen de modificaciones
Version 1.0	31-12-2023	Versión inicial del documento

Índice

1. Introducción

En diciembre de 2022 fue adjudicado a IMDEA Networks el proyecto “MLEDGE - Aprendizaje automático en la nube y en el borde (Cloud and Edge Machine Learning)” (REGAGE22e00052829516, en adelante el ‘Proyecto’ o MLEDGE) por parte del Ministerio de Asuntos Económicos y Transformación Digital del Gobierno de España, con fondos de la Unión Europea dentro del Plan de Recuperación, Transformación y Resiliencia (European Union - NextGenerationEU/PRTR). El proyecto tiene como objetivo habilitar un ecosistema próspero de servicios FL en el borde seguros y eficientes capaces de facilitar el uso de datos personales y B2B confidenciales para entrenar modelos de ML para consumidores mientras se protege la privacidad de los datos y de sus propietarios.

Los **objetivos generales del proyecto** se pueden resumir en los siguientes:

1. Hacer del aprendizaje federado una funcionalidad accesible y de fácil uso en el borde mediante el desarrollo de una capa de software intermedio y componentes que escondan la complejidad del procesamiento y el intercambio de datos que supone.
2. Resolver problemas técnicos asociados al aprendizaje federado en el borde de la nube.
3. Demostrar esta funcionalidad en casos de uso que reflejen problemas reales de la industria que pueden ser resueltos con estas tecnologías.
4. Explotar los resultados del proyecto involucrando a agentes externos y comunicar los hallazgos al público potencial en general.

Para alcanzar estos objetivos, se han catalogado una serie de **objetivos técnicos específicos** del proyecto que se resumen en los siguientes:

1. Diseñar un marco de desarrollo de servicios de aprendizaje federado (FLaaS) en el borde de la nube y componentes que ayuden a popularizar este tipo de servicios
2. Diseñar y desarrollar soluciones de seguridad (FedSecure) contra ataques de envenenamiento o inferencia lanzados desde servidores de borde rebeldes y/o nodos de agregación “honestos pero curiosos”.
3. Gestionar los desafíos de la portabilidad de datos en el borde de la red (DataEdge)
4. Crear un esquema de marca de agua (FedWM) para proteger contra la redistribución de los datos o metadatos que se intercambien entre servidores en el borde en el marco del FLaaS.
5. Crear una capa de lógica económica y de negocio (FLaaS Manager) que implemente una distribución justa de costes e ingresos entre las partes cuando colaboren en el entrenamiento de modelos de ML.
6. Soporte a DevOps y al desarrollo continuo de servicios de aprendizaje automático en la nube, optimizando los costes mediante su monitoreo, predicción y asignación inteligente y energéticamente eficiente de los trabajos de computación.

Finalmente, es uno de los objetivos básicos del proyecto diseñar, implementar y hacer públicos demostradores que trabajen con datos sensibles de individuos, y alimenten modelos de aprendizaje automático en diferentes campos de la industria .A tal fin, en la primera parte del proyecto se ha realizado una selección de empresas para el desarrollo de la plataforma FLaaS y el monitoreo de costes de computación, así como el diseño e implementación de casos de uso de negocio reales que se beneficien del aprendizaje distribuido en el borde de la nube.

El presente documento se corresponde con el entregable E2.1 de título “Versión preliminar de los componentes básicos de MLEDGE” y tiene como objetivo la presentación del avance de los componentes básicos de MLEDGE en los que el equipo de trabajo ha estado involucrado en el primer año del proyecto..

La estructura del documento será la siguiente. En la sección 2 resumimos la arquitectura propuesta para el proyecto y cómo esta se alinea con los objetivos del mismo. Las siguientes secciones, de la 3 a la 7, presentan los avances que se han alcanzado en los diferentes componentes. Debido a que el idioma del equipo de trabajo es el inglés, estas secciones contienen material en este lenguaje, por ejemplo, las publicaciones y los papeles de trabajo. Finalmente, la sección 8 presenta las conclusiones y siguientes pasos en el proyecto.

2. Arquitectura de MLEDGE

En esta sección, se presenta la arquitectura del proyecto MLEDGE junto con una descripción de los diferentes componentes de la misma. El proyecto busca:

i) Avanzar en el estado del arte de los componentes de acuerdo con los objetivos científicos

ii) Demostrar estos avances sobre plataformas y casos de uso comerciales de forma que se facilite la explotación posterior de los resultados del proyecto en la economía real.

El diagrama mostrado en la figura siguiente resume la arquitectura de MLEDGE y los bloques del proyecto que se presentó como propuesta del proyecto.

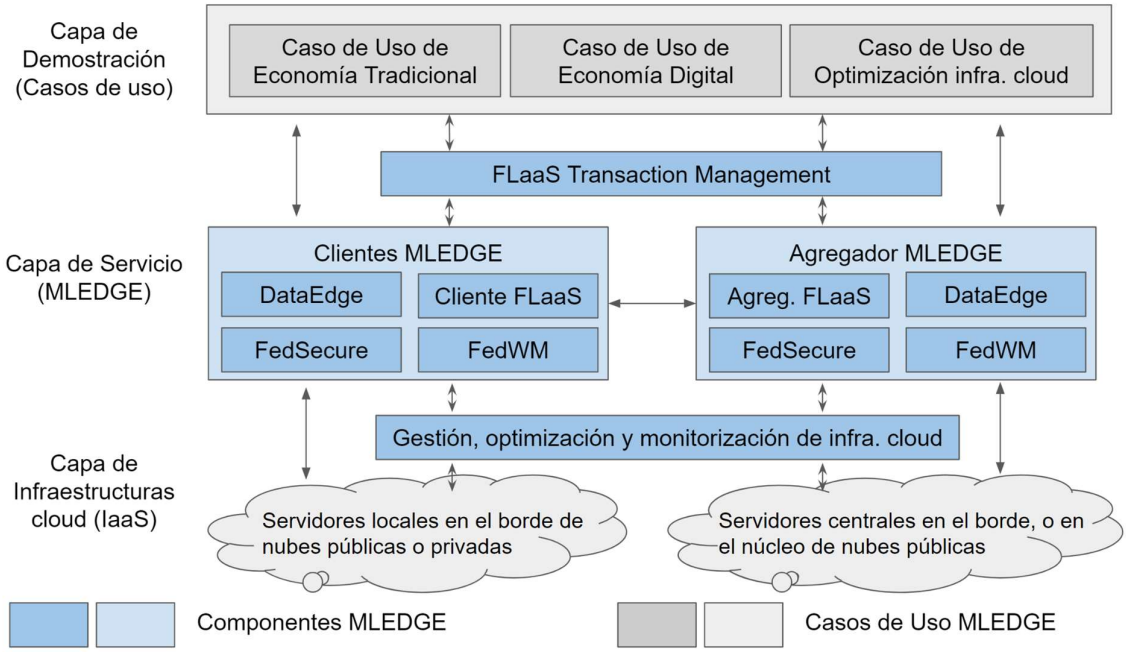


Figura 1. Diagrama de bloques de MLEDGE

La arquitectura de MLEDGE se articula en torno a tres capas:

- La **capa de infraestructuras** dispone los recursos de computación y comunicación necesarios para la ejecución del proyecto. Dicha capa de infraestructuras puede incluir recursos a diferentes niveles de la red, incluyendo clouds públicas o privadas "centralizadas" en el núcleo de la nube, nodos de computación en el borde de la nube, e incluso infraestructuras en casa o terminales de los usuarios.
- La **capa de servicios MLEDGE** busca integrar una serie de componentes que habiliten servicios FLaaS en el borde de la nube, y que puedan integrar componentes innovadores que provengan del desarrollo de los objetivos científicos del proyecto.
- La **capa de demostración** incluye casos de uso reales de empresas de la economía tradicional y digital, así como un caso de uso específico de optimización de infraestructuras cloud. Los casos de uso buscarán demostrar el uso del aprendizaje federado en el borde de la nube y de los componentes de MLEDGE en escenarios

reales y con datos reales. Para ello, durante los primeros seis meses de proyecto se ha realizado un screening de empresa y se han elaborado pliegos para realizar licitaciones de estos tres casos de uso a empresas españolas externas a IMDEA.

A continuación, se describen los componentes de la capa de servicios en los que se está trabajando durante el proyecto, su relación con los objetivos científicos, y los casos de uso que se buscará incorporar de la industria mediante estas licitaciones.

La capa de servicios de MLEDGE trabajará sobre la base de alguna de las soluciones comerciales de FLaaS que se describieron en la sección 4.2. Adicionalmente, se incluye una serie de componentes perfectamente alineados con los objetivos científicos del proyecto que buscan extender el estado del arte de algunos componentes del aprendizaje federado en el borde de la nube. Además, su integración con casos de uso específicos de la industria permitirá probar estos componentes directamente sobre casos reales y acelerar el tiempo hasta la explotación de los mismos por parte de la industria.

La siguiente tabla relaciona los componentes de la capa de servicios de MLEDGE con los objetivos científicos del proyecto:

Tabla 1. Relación entre objetivos científicos y componentes de la capa de servicios de MLEDGE

Objetivo científico	**Componente MLEDGE**
1. DevOps y desarrollo continuo para servicios de aprendizaje automático (FLaaS) que se ejecutan en borde de la nube	FLaaS
2. Uso eficiente de FL en nubes híbridas y protección contra ataques	FedSecure
3. Protección de datos sensibles o confidenciales que sean intercambiados entre dominios administrativos en la nube y el borde de la nube	FedWM
4. Equidad en términos de distribución de costos y ganancias cuando la computación en el borde se usa para entrenar de forma colaborativa modelos de ML	FLaaS Manager
5. Gestión de los desafíos de portabilidad de datos en el borde	DataEdge

En las siguientes secciones se presentan los avances de cada uno de los componentes MLEDGE, except DataEdge. Para este componente, tras el estudio del estado del arte, se decidió que se usarían formatos estándar de representación de datos e interfaces como XML, RDF, REST API, adaptados a las recomendaciones de DCAT, IDSA y GAIA-X para asegurar mecanismos estándar y controlados de intercambio de datos entre las partes que intervienen en el aprendizaje federado.

3. FLaaS

La investigación del componente de innovación de FLaaS se ha centrado en la implementación de un sistema de aprendizaje federado distribuido sobre el protocolo BitTorrent. El objetivo es evitar que exista una entidad centralizada que controle la información de todos los modelos entrenados por los diferentes clientes, a la vez que se permite el entrenamiento en paralelo de diferentes modelos propuestos por los clientes y adaptados a sus necesidades. De forma preliminar, hemos denominado a este proyecto internamente FLTorrent.

En el Anexo 1 se ofrece un resumen de los avances en este componente de innovación del proyecto a fecha diciembre de 2023 (en idioma inglés). Este componente se probará sobre la plataforma de FLaaS que incorpora al proyecto una de las empresas adjudicatarias de los lotes de trabajo de MLEDGE, será utilizado en los casos de uso del proyecto y demostrado al final del mismo.

4. FedSecure

Uno de los problemas de que adolece el aprendizaje federado es que es vulnerable a una serie de ataques realizados por parte de los clientes, el servidor o entidades externas. El módulo FedSecure tiene como objetivo trabajar para mejorar la seguridad del aprendizaje federado en el borde de la nube. Este trabajo se realizará en dos direcciones: 1) el desarrollo de modelos de reputación de los clientes para detectar ataques de envenenamiento, 2) la mejora de los criterios de agregación del modelo global.

El componente FedSecure ha comenzando trabajando por este segundo componente. Como resultado se ha publicado el siguiente artículo en NDSS, conferencia tier-1 en seguridad de sistemas distribuidos:

> T Chu, A Garcia-Recuero, C Iordanou, G Smaragdakis, N Laoutaris. Securing Federated Sensitive Topic Classification against Poisoning Attacks. 30th Annual Network and Distributed System Security Symposium, {NDSS} 2023
>
> Enlace a la presentación del paper en NDSS

El paper presenta una solución basada en aprendizaje federado para construir un clasificador distribuido capaz de detectar URLs contenido sensible, es decir, contenido relacionado con categorías como como la salud, las creencias políticas, la orientación sexual, etc. Aunque las limitaciones de los anteriores clasificadores offline/centralizados, sigue siendo centralizados, sigue siendo vulnerable a los ataques de envenenamiento de maliciosos que pueden intentar reducir la precisión de los usuarios benignos difundiendo actualizaciones defectuosas del modelo. Para evitarlo desarrollamos un esquema de agregación robusto basado en la lógica subjetiva y la detección de ataques basados en residuos. Empleando una combinación de de análisis teórico, simulación basada en trazas y validación experimental validación experimental con un prototipo y usuarios reales, demostramos que nuestro clasificador puede detectar contenidos sensibles con gran precisión, aprender nuevas etiquetas con rapidez, y seguir siendo robusto ante ataques de envenenamiento envenenamiento por parte de usuarios maliciosos, así como de entradas imperfectas de usuarios no maliciosos. no maliciosos.

El Anexo 2 del entregable incluye el paper (en inglés).

5. FedWM

El objetivo el componente es crear un esquema de marca de agua o similar para proteger de la propiedad de los datos cuando se requiera la redistribución de los datos o metadatos que se necesiten intercambiar entre servidores en el borde en el marco del FLaaS. Como primer avance, se una nueva técnica para modular la frecuencia de aparición de unos pocos tokens en un conjunto de datos para codificar una marca de agua invisible que puede utilizarse para proteger derechos de propiedad sobre los datos. Desarrollamos algoritmos heurísticos algoritmos heurísticos óptimos y rápidos para crear y verificar esas marcas de agua.

La técnica se presentará en una conferencia internacional Tier-1 en materia de ingeniería de datos:

> D. Isler, E. Cabana, A. Garcia-Recuero, G. Koutrika, N. Laoutaris, “FreqyWM: Frequency Watermarking for the New Data Economy,” Aceptado para publicación en IEEE International Conference of Data Engineering 2024.

En el artículo también demostramos la robustez de nuestra técnica contra contra varios ataques y derivamos límites analíticos para la probabilidad de "detectar" erróneamente una marca de agua en un conjunto de datos que no la contiene. Nuestra técnica es aplicable tanto a datos unidimensionales a conjuntos de datos unidimensionales y multidimensionales. tipo de token, permite un control fino de la distorsión introducida y se puede utilizar en una variedad de casos de uso que implican la compra y en los mercados de datos actuales.

El anexo 3 del entregable incluye el paper (en inglés)

6. FLaaS Manager

De alguna manera, el aprendizaje federado asume que los diferentes nodos que participan en el entrenamiento del modelo de aprendizaje distribuido tienen incentivo suficiente en mejorar el modelo para participar activamente en el proceso. Esto ha sido así en los primeros modelos de aprendizaje federado, como los empleados por el teclado de Google. Sin embargo, según se desarrolle la tecnología y aumenten los casos de uso, puede haber ocasiones en que la entidad que dispone los datos para entrenar los modelos no necesariamente es la que está interesada en el desarrollo de estos modelos.

Por el contrario, los mercados de datos buscan diseñar entidades que sean capaces de mediar entre los proveedores y los consumidores de datos sin que medie ningún interés de los primeros en el uso que los segundos hacen de los mismos. El diseño de estos mercados de datos distribuidos tiene una serie de desafíos técnicos, algunos de los cuales son objeto de investigación en el proyecto. En este sentido, se está trabajando en dos frentes:

1. Facilitar el establecimiento de precios de los datos, para dinamizar y reducir la incertidumbre en las transacciones, facilitando la toma de decisiones de precios por parte de la parte vendedora, y proporcionando información para la toma de decisiones de compra. En esta dirección se ha conseguido una primera publicación del primer modelo de predicción de precios basado en información de mercado y que se presentó en 2023 en una conferencia internacional Tier-1 en materia de ingeniería de datos en Anaheim, California::

   Santiago Andrés Azcoitia, Costas Iordanou, and Nikolaos Laoutaris. Understanding the Price of Data in Commercial Data Marketplaces. April 2023. 39th IEEE International Conference on Data Engineering (ICDE 2023)

   Enlace a la presentación (Talk)

   Enlace a las transparencias (Slides)

   Enlace al conjunto de datos compartido con la comunidad (Dataset)

   En la actualidad, se trabaja en la federación de este modelo de precios, de forma que la información de oferta y transacciones resida en los proveedores y mercados de datos, pero puedan compartir modelos en la nube para establecer los precios en base a su conocimiento conjunto.

2. Definir una nueva plataforma de mercado de datos.

   Se han propuesto nuevos mecanismos de mercado de datos (DM) para entrenar de forma colaborativa modelos compartidos por compradores potenciales mediante arquitecturas distribuidas que utilizan datos bajo el control de la plataforma. tivamente modelos compartidos por compradores potenciales mediante arquitecturas distribuidas que utilizan datos bajo el control de la plataforma. En la mayoría de los casos, los DM exigen a los compradores que compartan información y modelos confidenciales, lo que compromete su escalabilidad y la privacidad y propiedad intelectual de los compradores, y cargan a la plataforma con el coste del procesamiento para seleccionar o evaluar los datos, que no es insignificante. Además, cargan a la plataforma con el coste del procesamiento para seleccionar o evaluar los datos, que, según demostramos, dista mucho de ser insignificante si nos basamos en

los costes reales de las nubes públicas y los DM comerciales, lo que pone en peligro su viabilidad.

Se ha definido una novedosa arquitectura de mercado de datos que permite a los compradores probar datos en su modelo para seleccionar y comprar los activos que mejor se adapten a sus necesidades, y propone cobrar a los mismos por los costes de procesamiento relacionados con las transacciones de datos. A diferencia de anteriores diseños, nuestra propuesta obliga a los compradores a preocuparse por la cantidad de procesamiento solicitado a la plataforma. Mostramos técnicas para que los compradores optimicen el coste de los procesos de selección y compra de datos, a saber, novedosas estrategias de compra inteligente para reducir el número de solicitudes de evaluación, y el uso de "modelos de valoración de marionetas" (*puppet valuation models* o PVM) y "funciones de valoración" (*valuation functions* o VF) para reducir su complejidad. Los PVM y las VF también sortean el problema de los compradores que comparten propiedad intelectual sensible. Creemos que nuestro mercado de datos está listo para ser implementado utilizando la funcionalidad sandbox ya existente de entidades comerciales, y estamos trabajando en demostrar su viabilidad utilizando casos de uso de clasificación de imágenes y predicción en base a datos de movilidad.

En el anexo 5 presentamos el avance en este componente, que preliminarmente denominamos Try-Before-You-Buy.

3. En un entorno de aprendizaje distribuido descentralizado. cada cliente debe seleccionar los clientes en los que confía para entrenar su modelo y que mejor sirven a su propósito particular. En este contexto, la valoración de los datos o de los gradientes aportados por esos otros clientes al modelo que está entrenando cada nodo participante es fundamental para resolver este problema. Este campo de trabajo todavía no se ha trabajado en el proyecto y se planea hacerlo en el año entrante.

7. Conclusión

En el presente documento se ha presentado el avance de los componentes del proyecto MLEDGE y una primera versión preliminar de los mismos. Uno de los grandes objetivos del proyecto y del PRTR es la explotación de los avances científicos generados por parte de la industria. En este sentido, se ha introducido el contenido de tres casos de uso que han sido objeto de sendas licitaciones en el marco del proyecto para contratar agentes de la industria que se encarguen de demostrar la viabilidad del aprendizaje federado en el borde de la nube, y que sirvan a la vez como prueba de concepto para los diferentes componentes científicos que se desarrollen durante el proyecto.

Los siguientes pasos en el proyecto son fundamentalmente tres:

- Incorporar al proyecto a las empresas que se encarguen de elaborar los casos de uso del proyecto
- Continuar con el trabajo de investigación y desarrollo de los componentes
- Estudiar la integración de estos componentes técnicos de MLEDGE con la plataforma FLaaS que se emplee y en los casos de uso en que puedan ser útiles.

Durante este proceso y según se disponga de mayor visibilidad respecto a los planes de las diferentes empresas adjudicatarias, se utilizará la información del presente entregable para alinearla con los planes de las empresas adjudicatarias. Esto con el objetivo de incorporar más detalles sobre los casos de uso, sus requisitos y el diseño de la plataforma sobre la que se desarrollarán estos en los documentos de análisis de requisitos y diseño de los casos de uso. Así mismo, se informará en estos entregables sobre qué módulos se van a probar en cada caso de uso y con qué demostradores prácticos se probará su funcionamiento.

Anexo 1: FLTorrent progress report

Summary

The FLTorrent project has made substantial progress in the realm of decentralized federated learning, emphasizing a delicate balance between security, performance, and bandwidth optimization. Decentralized Federated Learning (FL) is a variant of FL, where the central server is eliminated and clients can send local gradients to others by peer-to-peer communication. Despite their pioneering contributions to decentralized FL, most of these works focus on improving the convergence speed of the model. Decentralized FL still has privacy issues that cannot be ignored since the clients can investigate their neighbors' gradients directly. FLTorrent, with its foundation in BitTorrent, introduces a novel dimension to the distributed learning architecture. By leveraging the inherent efficiency and scalability of BitTorrent's peer-to-peer communication model, FLTorrent enhances its ability to distribute and synchronize model updates across a decentralized network of peers. This not only fosters collaboration but also facilitates the seamless exchange of information, contributing to the democratization of machine learning processes.

This report encapsulates the detailed efforts to design, implement, and evaluate the FLTorrent system with a particular focus on optimizing loss while adhering to constraints on upload and download links' bandwidth. Notably, our efforts also include incorporating insights from FLtorrent, a groundbreaking approach that leverages BitTorrent for the efficient communication of model chunks.

Objectives

The overarching objectives of the FLTorrent project have been intricately crafted to address the multifaceted challenges associated with decentralized federated learning:

- **Decentralized FL Model Development**: Develop a collaborative and secure decentralized federated learning model.
- **Privacy-Preserving Techniques**: Implement advanced techniques to ensure privacy, focusing on anonymizing data chunks during the learning process.
- **Bandwidth-Constrained Performance Optimization**: Optimize model loss while considering constraints on upload and download links' bandwidth.

Methodologies

Decentralized Federated Learning Model

The FLTorrent decentralized FL model utilizes a collaborative learning approach, distributing the training process across a network of peers. This ensures a robust model while promoting privacy, security, and efficient bandwidth utilization.

Privacy-Preserving Techniques

Our commitment to privacy is underscored by the implementation of multiple advanced techniques, including the anonymization of data chunks. By employing a combination of privacy-preserving methods, the FLTorrent system fortifies itself against potential security

threats and breaches. This approach not only safeguards sensitive information during data exchange but also ensures the integrity of the decentralized federated learning process.

Bandwidth-Constrained Performance Optimization Heuristics

In the pursuit of optimal performance, FLTorrent incorporates performance optimization heuristics that explicitly consider constraints on upload and download links' bandwidth. This strategic approach minimizes loss while ensuring efficient bandwidth utilization. The system employs adaptive chunk distribution heuristics, considering constraints on upload and download links' bandwidth. This approach optimizes the learning process by adapting to the unique bandwidth capabilities of individual peers, minimizing loss, and enhancing overall performance. Importantly, we draw inspiration from FLTorrent, a paradigm that utilizes BitTorrent for the efficient communication of model chunks across distributed peers.

Evaluation

The current phase of the FLTorrent project revolves around a meticulous evaluation process, focusing on:

1. **Security Evaluation**: Ensuring the effectiveness of privacy-preserving techniques against potential privacy breaches and attacks.
2. **Bandwidth-Constrained Performance Evaluation**: Assessing the impact of performance optimization heuristics with a focus on minimizing loss while adhering to constraints on upload and download links' bandwidth.

Next Steps

As the project transitions into the evaluation phase, the following key steps will be undertaken:

Thorough Analysis and Validation: Conduct a detailed analysis of results obtained from real-world FL projects, with a specific emphasis on bandwidth-constrained performance.

Stakeholder Feedback and Refinement: Engage with stakeholders to gather feedback, facilitating iterative refinement of the FLTorrent model with a focus on bandwidth optimization.

Parameter Fine-Tuning: Fine-tune privacy and performance parameters based on evaluation findings, ensuring the effective optimization of loss within bandwidth constraints.

Conclusion

The FLTorrent project stands at the forefront of decentralized federated learning, emphasizing not only security and privacy but also performance optimization within the constraints of upload and download links' bandwidth. It extends beyond conventional methodologies, offering a solution that is not only secure and privacy-conscious but also highly performant in bandwidth utilization. The integration of privacy-preserving techniques, which encompass a sophisticated blend of multiple advanced methods, ensures that sensitive information remains confidential during the decentralized learning process. Moreover, FLTorrent 's commitment to performance optimization is underscored by the deployment of bandwidth-constrained heuristics. These heuristics, inspired by the dynamic distribution principles of FLTorrent, adaptively allocate model chunks based on the bandwidth constraints of individual peers. This approach not only

minimizes loss in the learning process but also optimizes the use of available bandwidth resources, contributing to the overall efficiency of the federated learning model.

Drawing inspiration from Bit Torrent network and incorporating its principles, it emerges not just as a solution but as a transformative force in the realm of decentralized federated learning. This amalgamation of cutting-edge technologies, privacy-preserving methodologies, and innovative bandwidth utilization strategies positions FLTorrent as a beacon of innovation and a pioneering solution in achieving the delicate equilibrium between security, performance, and efficient bandwidth utilization in the ever-evolving landscape of machine learning. The successful integration of privacy-preserving techniques and bandwidth-constrained performance optimization heuristics positions FLTorrent as a pioneering solution in achieving a delicate equilibrium between security, performance, and efficient bandwidth utilization.

Anexo 2: Securing Federated Sensitive Topic Classification against Poisoning Attacks

Securing Federated Sensitive Topic Classification against Poisoning Attacks

Tianyue Chu
IMDEA Networks Institute
Universidad Carlos III de Madrid

Alvaro Garcia-Recuero
IMDEA Networks Institute

Costas Iordanou
Cyprus University of Technology

Georgios Smaragdakis
TU Delft

Nikolaos Laoutaris
IMDEA Networks Institute

***Abstract*—We present a Federated Learning (FL) based solution for building a distributed classifier capable of detecting URLs containing sensitive content, i.e., content related to categories such as health, political beliefs, sexual orientation, etc. Although such a classifier addresses the limitations of previous offline/centralised classifiers, it is still vulnerable to poisoning attacks from malicious users that may attempt to reduce the accuracy for benign users by disseminating faulty model updates. To guard against this, we develop a robust aggregation scheme based on subjective logic and residual-based attack detection. Employing a combination of theoretical analysis, trace-driven simulation, as well as experimental validation with a prototype and real users, we show that our classifier can detect sensitive content with high accuracy, learn new labels fast, and remain robust in view of poisoning attacks from malicious users, as well as imperfect input from non-malicious ones.**

I. Introduction

Most people are not aware that tracking services are present even on sensitive web domains. Being tracked on a cancer discussion forum, a dating site, or a news site with non-mainstream political affinity can be considered an "elephant in the room" when it comes to the anxieties that many people have about their online privacy. The General Data Protection Regulation (GDPR) [33] puts specific restrictions on the collection and processing of sensitive personal data *"revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership, also genetic data, biometric data for the purpose of uniquely identifying a natural person, data concerning health or data concerning a natural persons sex life or sexual orientation"*. So do other public bodies around the world, *e.g.* in California (California Consumer Privacy Act (CCPA) [34]), Canada [35], Israel [36], Japan [37], and Australia [38].

In a recent paper, Matic et al. [4] showed how to train a classifier for detecting whether the content of a URL relates to any of the above-mentioned sensitive categories. The classifier was trained using 156 thousand sensitive URLs obtained from the Curlie [32] crowdsourced web taxonomy project. Despite the demonstrated high accuracy, this method has limitations that stem from being *centralised* and *tied to a fixed training set*. The first limitation means that the method cannot be used "as is" to drive a privacy-preserving distributed classification system. The second limitation implies that it is not straightforward to cover new labels related to yet unseen sensitive content. For example, in their work the Health category could be classified with accuracy greater than 90%. However, the training labels obtained from Curlie in 2020 did not include any labels related to the COVID-19 pandemic. Therefore, as will be shown later, this classifier classifies COVID-19 related sites with only 53.13% accuracy.

Federated Learning (FL) [5], [13] offers a natural solution to the above two mentioned limitations, namely, centralized training and training for a fixed training set. FL allows different clients to train their classification models locally without revealing new or existing sensitive URLs that they label, while collaborating by sharing model updates that can be combined to build a superior global classification model. FL has proved its value in a slew of real-world applications, ranging from mobile computing [46]–[48] to health and medical applications [49]–[51]. However, due to its very nature, FL is vulnerable to so-called *poisoning attacks* [12], [26] mounted by malicious clients that may intentionally train their local models with faulty labels or backdoor patterns, and then disseminate the resulting updates with the intention of reducing the classification accuracy for other benign clients. State-of-the-art approaches for defending against such attacks depend on robust aggregation [8], [15], [16], [20], [27], [60] which, as we will demonstrate later, are slow to converge, thereby making them impractical for the sensitive-content classification problem that we tackle in this paper.

Our Contributions: In this paper, we employ FL for sensitive content classification. We show how to develop a robust FL method for classifying arbitrary URLs that may contain GDPR sensitive content. Such a FL-based solution allows building a distributed classifier that can be offered to end-users in the form of a web browser extension in order to: (i) warn them before and while they navigate into such websites, especially when they are populated with trackers, and (ii) allow them to contribute new labels, e.g., health-related websites about COVID-19, and thus keeping the classifier always up-to-date. To the best of our knowledge this method represents the first use of FL for such task.

Network and Distributed System Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.23112
www.ndss-symposium.org

Our second major contribution is the development of a reputation score for protecting our FL-based solution from poisoning attacks [12], [26]. Our approach is based on a novel combination of subjective logic [3] with residual-based attack detection. Our third contribution is the development of an extensive theoretical and experimental performance evaluation framework for studying the accuracy, convergence, and resilience to attacks of our proposed mechanism. Our final contribution is the implementation of our methods in a prototype system called *EITR* (standing for "Elephant In the Room" of privacy) and our preliminary experimental validation with real users tasked to provide fresh labels for the accurate classification of COVID-19 related URLs.

Our findings: Using a combination of theoretical analysis, simulation, and experimentation with real users, we:

- Demonstrate experimentally that our FL-based classifier achieves comparable accuracy with the centralised one presented in [4].
- Prove analytically that under data poisoning attacks, our reputation-based robust aggregation built around subjective logic, converges to a near-optimal solution of the corresponding Byzantine fault tolerance problem under standard assumptions. The resulting performance gap is determined by the percentage of malicious users.
- Evaluate experimentally our solution against state-of-the-art algorithms such as Federated Averaging [5], Coordinate-wise median [20], Trimmed-mean [20], FoolsGold [8], [15], Residual-based re-weighting [16] and FLTrust [60], and show that our algorithm is robust under Byzantine attacks by using different real-world datasets. We demonstrate that our solution outperforms these popular solutions in terms of convergence speed by a factor ranging from $1.6\times$ to $2.4\times$ while achieving the same or better accuracy. Furthermore, our method yields the most consistent and lowest Attack Success Rate (ASR), with at least 72.3% average improvement against all other methods.
- Validate using our *EITR* browser extension that our FL-based solution can quickly learn to classify health-related sites about COVID-19, even in view of noisy/inconsistent input provided by real users.

The remainder of the article is structured as follows: Section II introduces the background for our topic. Section III presents our reputations scheme for FL-based sensitive content classification, as well as its theoretical analysis and guarantees. Section IV covers our extensive performance evaluation against the state-of-the-art and Section V some preliminary results from our *EITR* browser extension. Section VI concludes the paper and points to on-going and future work including the generalization of our method to other topics.

II. Background

A. A Centralised Offline Classifier for Sensitive Content

Matic et al. [4] have shown how to develop a text classifier able to detect URLs that contain sensitive content. This classifier is centralised and was developed in order to conduct a one-off offline study aimed at estimating the percentage of the web that includes such content. Despite achieving an accuracy of at least 88%, utilising a high-quality training set meticulously collected by filtering the Curlie web-taxonomy project [32], this classifier cannot be used "as is" to protect real users visiting sensitive URLs populated by tracking services.

B. Challenges in Developing a Practical Classifier for Users

From offline to online: The classifier in [4] was trained using a dataset of 156 thousand sensitive URLs. Despite being the largest dataset of its type in recent literature, this dataset is static and thus represents sensitive topics up to the time of its collection. This does not mean, of course, that a new classifier trained with this data would never be able to accurately classify new URLs pertaining to those sensitive categories. This owes to the fact that categories such as Health, involve content and terms that do not change radically with time. Of course, new types of sensitive content may appear that, for whatever reason, may not be so accurately classified using features extracted from past content of the same sensitive category. Content pertaining to the recent COVID-19 pandemic is such an example. Although the Health category had 74,764 URLs in the training set of [4] which lead to a classification accuracy of 88% for Health, as we will see later in Figure 14 middle of Section V-C, the classifier of [4] classifies accurately as Health only 53.13% of the COVID-19 URLs with which we tested it. This should not come as a surprise since the dataset of [4] corresponds to content generated before the first months of 2020, during which COVID-19 was not yet a popular topic. Therefore, we need to find a way to update an existing classifier so that it remains accurate as new sensitive content appears.

From centralised to distributed: A natural way to keep a classifier up-to-date is to ask end-users to label new sensitive URLs as they encounter them. End-users can report back to a centralised server such URLs which can then be used to retrain the classification model. This, however, entails obvious privacy challenges of "Catch-22" nature, since to protect users by warning them about the presence of trackers on sensitive URLs, they would first be required to report to a potentially untrusted centralised server that they visit such URLs. Even by employing some methods for data scarcity, e.g., semi-supervised learning, the manual labelling from users remains sensitive and may be harmed by the untrusted server. Federated Learning, as already mentioned, is a promising solution for avoiding the above Catch22 by conducting a distributed, albeit, privacy-preserving, model training. In an FL approach to our problem, users would label new URLs locally, e.g., a COVID-19 URL as Health, retrain the classifier model locally, and then send model updates, not labelled data, to a centralised server that collects such updates from all users, compiles and redistributes the new version of the model back to them. In Section III we show how to develop a distributed version of the sensitive topic classifier of [4] using FL. The trade-off of using FL, is that the distributed learning group becomes vulnerable to attacks, such as "label-flipping" poisoning attacks discussed in Section IV. This paper develops a reputation scheme for mitigating such attacks. Other types of attacks and measures for preserving the privacy of users that participate in a FL-based classification system for sensitive content are discussed in Section VI.

C. Related Work

Privacy preserving crowdsourcing: Similar challenges to the ones discussed in the previous paragraph have been faced in

services like the *Price $heriff* [54] and *eyeWnder* [55] that have used crowdsourcing to detect online price discrimination and targeted advertising, respectively. Secure Multi-Party Computation (SMPC) techniques such as private k-means [56] are used to allow end-users to send data in a centralised server in a privacy-preserving manner. The centralised computation performed by *Price $heriff* and *eyeWnder* is not of ML nature, thus leaving data anonymisation as the main challenge, for which SMPC is a good fit. Classifying content as sensitive or not is a more complex ML-based algorithm for which FL is a more natural solution than SMPC.

General works on FL: FL [5], [13] is a compelling technique for training large-scale distributed machine learning models while maintaining security and privacy. The motivation for FL is that local training data is always kept by the clients and the server has no access to the data. Due to this benefit that alleviates privacy concerns, several corporations have utilised FL in real world services. In mobile devices, FL is used to predict keyboard input [46], human mobility [47] and behaviour for the Internet of Things [48]. FL is also applied in healthcare to predict diseases [49], [50], detect patient similarity [51] while overcoming any privacy constrains. For the classification, FL is not only implemented for image classification [52] but also text classification [53].

Resilience to poisoning attacks: Owing to its nature [12], [26], FL is vulnerable to poisoning attacks, such as label flipping [16] and backdoor attacks [12]. Therefore, several defence methods have been developed [8], [15], [16], [20]. While these state-of-the-art approaches perform excellently in some scenarios, they are not without limitations. First, they are unsuitable for our sensitive content classification, which necessitates that a classifier responds very fast to "fresh" sensitive information appearing on the Internet. In existing methods, the primary objective is to achieve a high classification accuracy. This is achieved via statistical analysis of client-supplied model updates and discarding of questionable outliers before the aggregation stage. However, since the server distrusts everyone by default, even if an honest client discovers some fresh sensitive labels, its corresponding updates may be discarded or assigned low weights, up until more clients start discovering these labels. This leads to a slower learning rate for new labels.

Second, recent studies [12], [26] have shown that existing Byzantine-robust FL methods are still vulnerable to local model poisoning since they are forgetful by not tracking information from previous aggregation rounds. Thus, an attacker can efficiently mount an attack by spreading it across time [31]. For example, [22] recently showed that even after infinite training epochs, any aggregation which is neglectful of the past cannot converge to an efficient solution.

The preceding studies demonstrate the importance of incorporating clients' previous long-term performance in evaluating their trustworthiness. Few recent studies have considered this approach [22], [60]. In [22], the authors propose leveraging historical information for optimisation, but not for assessing trustworthiness. In [60], a trust score is assigned to each client model update according to the cosine similarity between the client's and server's model updates, which is trained on the server's root dataset (details in Section IV-A3). However, it is impractical for a server to obtain additional data, such as a root dataset, in order to train a server-side model. In addition, because the server collects root data only once and does not update it throughout the training process, when new types of content emerge over time, the root data may become stale thereby harming the classifier's performance. Other recent studies employ spectral analysis [63], differential privacy [65], and deep model inspection [66] to guard against poisoning attacks, but, again, they do not use historical information to assess the reliability of clients. To measure client trustworthiness without collecting additional data at the server, in the next sections we show how to design a robust aggregation method to generate reputation automatically based on the historical behaviours of clients, which is a more realistic approach for a real FL-based decentralised system implemented as a browser extension for clients.

TABLE I: Notation

Abbreviation	Description
M	the total number of clients
N	the number of parameters of global model
Q	the number of samples of each client
T	the total number of iterations
$w_{i,n}^t$	the n-th parameter from client i in t iteration
$x_{i,n}^t$	the ranking of $w_{i,n}^t$ in w_n^t
A_n, B_n	the slope and intercept of repeated median linear regression
$e_{i,n}^t$	the normalised residual of the n-th parameter from client i in t iteration

III. A Robust FL Method for Classifying Sensitive Content on the Web

In this section, we first show how to build an FL-based classifier for sensitive content. Then we design a reputation score for protecting against poisoning attacks. We analyse theoretically the combined FL/reputation-based solution and establish convergence and accuracy guarantees under common operating assumptions.

A. FL Framework for Classifying Sensitive Content

Table I presents the notation that we use in the remainder of the paper. In FL, clients provide the server updated parameters from their local model, which the server aggregates to build the global model M.

Suppose we have M clients participating in our classification training task and the dataset $\mathcal{D} = \bigcup_{i=1}^{M} \mathcal{D}_i$, where $\mathcal{D}_i \sim \mathcal{X}_i(\mu_i, \sigma_i^2)$ denotes the local data of client i from non-independent and non-identically (Non-IID) distribution $\mathcal{X}_i$ with the mean μ_i and standard deviation σ_i. In our task, the clients' data is the textual content of URLs stripped of HTML tags. The objective function of FL, $\mathcal{L} : \mathbb{R}^d \rightarrow \mathbb{R}$ which is the negative log likelihood loss in our task, can be described as

$$\mathcal{L}(w) = \mathbb{E}_{D\sim\mathcal{X}}\left[l(w;\mathcal{D})\right]$$

where $l(w;\mathcal{D})$ is the cost function of parameter $w \in \mathcal{W} \subseteq \mathbb{R}^d$. Here we assume $\mathcal{W}$ is a compact convex domain with diameter d. Therefore, the task becomes

$$w^* = \arg\min_{w\in\mathcal{W}} \mathcal{L}(w)$$

To find the optimal w^*, we employ Stochastic Gradient Descent (SGD) to optimise the objective function.

During the broadcast phase, the server broadcasts the classification task and training instructions to clients. Then, the clients apply the following standard pre-processing steps

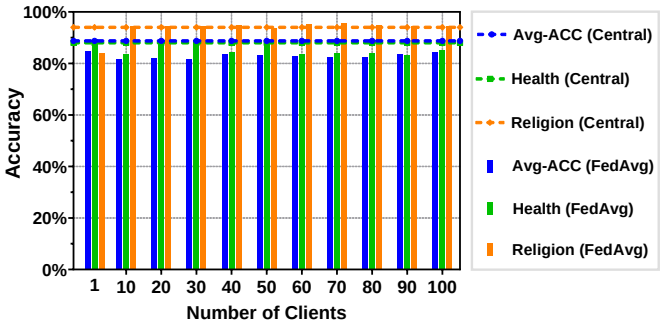


Fig. 1: Accuracy of FL classifiers and centralised classifiers in Health, Religion and all category.

on the webpage content, that is, transformation of all letters in lowercase and the removal of stop words. Next, the clients extract the top one thousand features utilising the Term Frequency-Inverse Document Frequency (TF-IDF) [58] as in [4]. At iteration t, the client i receives the current global model M_{global} and then following the training instructions from server, trains the local model on its training data $\mathcal{D}_i$ and optimises $w_i^t = \arg\min_w \mathcal{L}_i(w_i^t)$ by using SGD:

$$w_i^t \leftarrow w_i^{t-1} - r\frac{\partial \mathcal{L}_i(w_i^{t-1})}{\partial w}$$

where $\mathcal{L}_i(w) := \mathbb{E}_{\mathcal{D}_i \sim \mathcal{X}}[l(w;\mathcal{D}_i)] = \frac{1}{Q_i}\sum_{j=1}^{Q_i} l(w;\mathcal{D}_i^j)$, $\mathcal{D}_i^j$ and Q_i means the j-th sample and the number of samples of the client i respectively, and r is the learning rate.

In every iteration, after finishing the training process the clients send back their local updates to the server. Then, the server computes a new global model update by combining the local model updates via an aggregation method AGG as follows:

$$w^t = \texttt{AGG}\left(\{w_i^t\}_{i=1}^M\right)$$

Here we utilise the basic aggregation method (FedAvg) [5], which uses the fraction of each client's training sample size in total training samples as the average weights:

$$w^t = \sum_{i=1}^{M} \frac{Q_i}{Q} w_i^t$$

We introduce other robust aggregation methods in the next subsection. Subsequently, the server uses the global model update to renew the global model M_{global}.

Using the above FL-based framework we first evaluate how the number of users in the system affects the average accuracy of the classifier. The results for the sensitive categories, Health and Religion, as well as the overall average accuracy (Avg-ACC) are depicted in Figure 1. A first observation is that when a fixed size dataset is divided into multiple segments and distributed to more clients, the model's accuracy decreases since each client has less data for training. Compared to the centralised classifier, using the same data, the accuracy of the FL classifier is slightly lower, which is expected when the training is distributed to a larger number of clients. Overall, we observe that the average accuracy difference between the FL and the centralised classifier is 5.76%, and this remains steady as the number of clients grows. In addition, Looking at the different sensitive categories (Health and Religion), we see the FL-based classifier achieves an accuracy very close to

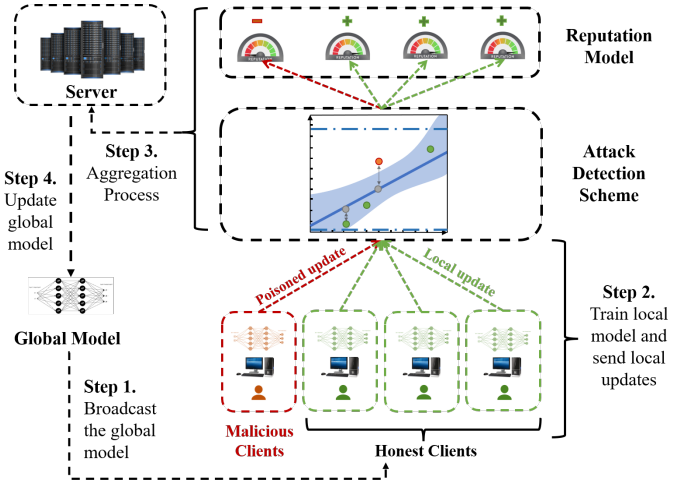


Fig. 2: Overview of reputation-based aggregation algorithm.

the corresponding one of the centralised classifier for these categories (on average 0.8533 vs. 0.88 and 0.9366 vs. 0.94, respectively).

B. A Reputation score for Thwarting Poisoning Attacks

Figure 2 shows an overview of our reputation-based aggregation algorithm consisting of three components: the *attack detection scheme*, the *reputation model*, and the *aggregation module*. The attack detection scheme re-scales and rectifies damaging updates received from clients. Then, the reputation model calculates each client's reputation based on their past detection results. Finally, the aggregation module computes the global model by averaging the updates of the clients using their reputation scores as weights. We detail each component in the following subsections.

1) Attack Detection Scheme: Our attack detection scheme aims to reduce the impact of suspicious updates by identifying them and applying a rescaling algorithm. At every iteration, when model updates from clients arrive at the server, we apply Algorithm 1 there to rescale the range of values for those parameters in the updates.

This restriction on the value range aims not only to minimise the impact of abnormal updates from attackers but also to limit the slope for the repeat median regression. Considering the n-th parameter in round t from all the participants, we calculate the standard deviation $\sigma(w_{i,n}^t)$ of this series. Then we sort them in ascending order and determine the range by subtracting the lowest value from the highest one. If the result is above the threshold ϖ, we rescale the highest and lowest value by deducting and adding its standard deviation respectively to further bound their range.

Then, a robust regression [14] is carried out to identify outliers among the updates in the current round. Outlier detection is a well-established topic in statistics. Robust regression methods often handle outliers by using the median estimators. Median-based aggregation methods have a rich and longstanding history in the area of robust statistics [21]. However, the methods developed by the traditional robust statistics can only withstand a small fraction of Byzantine clients, resulting in a low "breakdown point" [59]. Different from many other variations of the univariate median, the repeated median [14]

Algorithm 1: Rescale(w)

Input : $\{w_{i,n}^t\} \leftarrow$ Local Model parameters in round t

Output: $\{w_{i,n}^t\}$ with range of value less than ϖ

for $n \leftarrow 1$ **to** N **do**
  // Determine the maximum range
  $Rm = \max w_{i,n}^t - \min w_{i,n}^t = w_{i,n}^{t,(Max)} - w_{i,n}^{t,(Min)}$
  while $Rm > \varpi$ **do**
    // Rescale range based on standard deviation.
    $w_{i,n}^{t,(Max)} := w_{i,n}^{t,(Max)} - \sigma(w_{i,n}^t)$;
    $w_{i,n}^{t,(Min)} := w_{i,n}^{t,(Min)} + \sigma(w_{i,n}^t)$;
    // Updated Rm.
    $Rm = \max w_{i,n}^t - \min w_{i,n}^t = w_{i,n}^{t,(Max)} - w_{i,n}^{t,(Min)}$
  end while
end for

is impervious to atypical points even when their percentage is nearly 50%. The repeated median is defined as a modified U-statistic and the concept behind it is to utilise a succession of partial medians for computing approximation $\hat{\tau}$ of the parameter τ: For $k \in \mathbb{N}$, the value of parameter $\tau(z_1, \cdots, z_k)$ is determines by subset of k data points $z_1, \cdots, z_k$.

$$\hat{\tau} = \underset{z_1}{\text{median}} \left\{ \underset{z_2 \notin \{z_1\}}{\text{median}} \left\{ \underset{z_k \notin \{z_1, \cdots, z_{k-1}\}}{\text{median}} \tau(z_1, \cdots, z_k) \right\} \right\} \quad (1)$$

In our case, the intercept $\hat{A}$ and slop $\hat{B}$ are estimated by repeated median as bellow:

$$\hat{B}_n = \underset{i}{\text{median}} \left\{ \underset{i \neq j}{\text{median}} \{B_n(i,j)\} \right\} \quad (2)$$

$$\hat{A}_n = \underset{i}{\text{median}} \left\{ w_{i,n} - \hat{B}_n x_{i,n} \right\} \quad (3)$$

where $B_n(i,j) = \frac{w_{j,n} - w_{i,n}}{x_{j,n} - x_{i,n}}$, $x_{i,n}$ represents the index of $w_{i,n}$ in w_n which is sorted in ascending order.

Next, we employ the IRLS scheme [10] to generate each parameter's confidence score $s_{i,n}^t$ based on the normalised residual from repeated median regression, which is also utilised in a residual-based aggregation method [16]:

$$s_{i,n}^t = \frac{\sqrt{1 - diag(H_n^t)}}{e_{i,n}^t} \Psi \left(\frac{e_{i,n}^t}{\sqrt{1 - diag(H_n^t)}} \right) \quad (4)$$

where confidence interval $\Psi(x)$:

$$\Psi(x) = \max\{-\lambda\sqrt{2/M}, \min(\lambda\sqrt{2/M}, x)\}$$

and the hat matrix H_n^t:

$$H_n^t = x_n^t ({x_n^t}^T x_n^t)^{-1} {x_n^t}^T$$

with $e_{i,n}^t = \frac{25(M-1)\left(w_{i,n}^t - \hat{B}_n x_{i,n}^t - \hat{A}_n\right)}{37(M+4)\underset{i}{\text{median}}\left(|w_{i,n}^t - \hat{B}_n x_{i,n}^t - \hat{A}_n|\right)}$.

The distance between the point and the robust line is described by the confidence score derived from the normalised residual, which can be used to evaluate if the point is anomalous. Following the computation of the parameter's confidence score, and in light of the fact that some attackers want to generate updates with abnormal magnitudes in order to boost the damage, a useful protection is to identify low confidence values based on a threshold δ. Once the server recognises an update $\mathbf{w}_{i,n}^t$ of the client i with confidence values less than δ, rather than altering this update to the repeat median estimation, our technique replaces it with the median of $\mathbf{w}_n^t$, as follows:

$$\mathbf{w}_{i,n}^t = \begin{cases} \mathbf{w}_{i,n}^t & \text{if } s_{i,n}^t > \delta \\ \underset{i}{\text{median}} \left\{ \mathbf{w}_{i,n}^t \right\} & \text{if } s_{i,n}^t \leq \delta \end{cases} \quad (5)$$

With the above, not only we bound the range of updates, but also improve the aggregation by introducing a robustness estimator.

2) Reputation Model: During the aggregation phase in FL, we use a subjective logic model to produce client reputation scores. The subjective logic model is a subset of probabilistic logic that depicts probability values of belief and disbelief as degrees of uncertainty [3]. In the subjective logic model, reputation score R_i^t for client i in t iteration correlates to a subjective belief in the dependability of the client's behaviour [39], as measured by the belief metric opinion τ_i^t [9]. An opinion is comprised of three elements: belief b_i^t, disbelief d_i^t and uncertainty u_i^t, with restrictions that $b_i^t + d_i^t + u_i^t = 1$ and $b_i^t, d_i^t, u_i^t \in [0, 1]$. The reputation score may be calculated as the expected value of an opinion $E(\tau_i^t)$ which can be regarded as the degree of trustworthiness in client i. As a result, the value of the client's reputation is defined as follows:

$$R_i^t = E(\tau_i^t) = b_i^t + a u_i^t \quad (6)$$

where $a \in [0, 1]$ denotes the prior probability in the absence of belief, which reflects the fraction of uncertainty that may be converted to belief. On the other side, distinct observations determined by the rectification phase in our Algorithm 2 are used to count belief, disbelief, and uncertainty opinions. The positive observation denoted by P_i^t indicates that the update $\mathbf{w}_{i,n}^t$ is accepted ($s_{i,n}^t > \delta$), whereas a negative observation denoted by N_i^t indicates that the update is rejected ($s_{i,n}^t \leq \delta$). As a consequence, the positive observations boost the client's reputation, and vice versa. To penalise the negative observations from the unreliable updates, a higher weight η is assigned to negative observations than the weight κ to positive observations with constrain $\eta + \kappa = 1$. Therefore, in Beta distribution below:

$$Beta(p|\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha - 1}(1 - x)^{\beta - 1} \quad (7)$$

with the constraints $0 \leq x \leq 1$, parameters $\alpha > 0, \beta > 0$, and $x \neq 0$ if $\alpha < 1$ and $x \neq 1$ if $\beta < 1$. The parameters α and β that represent positive and negative observations respectively, can be expressed as below

$$\begin{cases} \alpha = \kappa P_i^t + W a \\ \beta = \eta N_i^t + W(1 - a) \end{cases} \quad (8)$$

where W is the non-information prior weight and the default value is 2 [3].

As a consequence, the expected value of Beta distribution, which also stands for reputation value, can be calculated as follows:

$$E(Beta(p|\alpha, \beta)) = \frac{\alpha}{\alpha + \beta} = \frac{\kappa P_i^t + W a}{\kappa P_i^t + \eta N_i^t + W} = R_i^t \quad (9)$$

Based on (6) and (9), we can derive

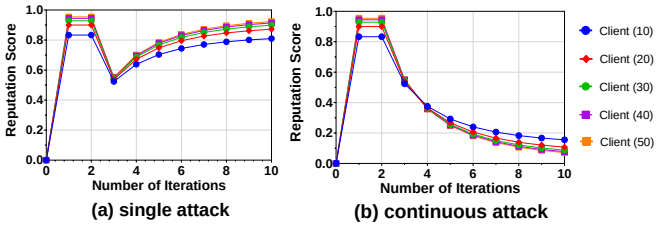


Fig. 3: The decay of reputation score in Client (X) with X model parameters when they (a) attack once at 3rd iteration and (b) attack continuously at and after the 3rd iteration.

$$\begin{cases} b_i^t = \frac{\kappa P_i^t}{\kappa P_i^t + \eta N_i^t + W} \\ d_i^t = \frac{\eta N_i^t}{\kappa P_i^t + \eta N_i^t + W} \\ u_i^t = \frac{W}{\kappa P_i^t + \eta N_i^t + W} \end{cases} \tag{10}$$

In addition, in order to take the client's historical reputation values in previous rounds into consideration, a time decay mechanism is included to lower the relevance of past performances without disregarding their influence. In other words, the reputation value from the most recent iteration contributes the most to the reputation model. We use exponential time decay in our model, as shown below:

$$\theta_{j,t} = \exp(-c(t-j)) \tag{11}$$

where $\exists c > 0$, $j \in [\tilde{s}, t]$, $\tilde{s} = \max(t-s, 0)$. We include a sliding window with a window length s that allows us to get a reputation for a certain time interval rather than the entire training procedure. We remove expired tuples with timestamps outside the window period during computation since they cannot provide meaningful information for the reputation. Hence, the final reputation score $\tilde{R}_i^t$ can be expressed as:

$$\tilde{R}_i^t = \frac{\sum_{j=\tilde{s}}^{t} \theta_{j,t} R_i^j}{\sum_{j=\tilde{s}}^{t} \theta_{j,t}} \tag{12}$$

To demonstrate how the reputation model evolves, we consider four scenarios where each client: (i) only attacks once at the same iteration, (ii) attacks continuously after launching an attack at the same iteration, (iii) only attacks once at different iteration, (iv) attacks continuously after launching an attack at different iteration. Here, clients conduct attacks as described in Section IV-A2 by utilising polluted data while training the local model, whereas the server uses our attack detection mechanism to identify these attacks.

Figure 3 displays the first two scenarios (i)-Figure 3a and (ii)-Figure 3b, respectively with Client X, who has X parameters in their local models, under single and continuous attack. In Figure 3a, all of the clients only attack once at the third iteration. When they start attacking, their reputation score plummets dramatically. In both scenarios, we observe the client who has more parameters has a larger relative decline in reputation score. This is also compatible with Corollary 1 in the Section III-C, that is, increasing the number of parameters N in the global model results in a lower error rate.

Figure 4 shows the last two scenarios (iii) and (iv) respectively with clients, who have 20 parameters in their local models, under single and continuous attack. In Figure 4a,

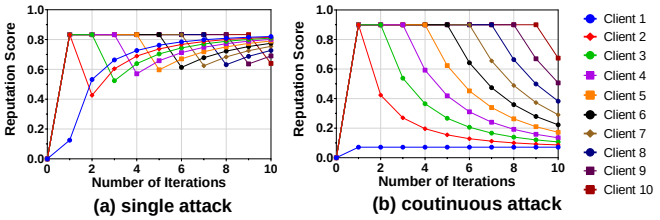


Fig. 4: The decay of reputation score in Client X with same model parameters when they (a) attack once at X iteration and (b) attack continuously after starting to attack at X iteration.

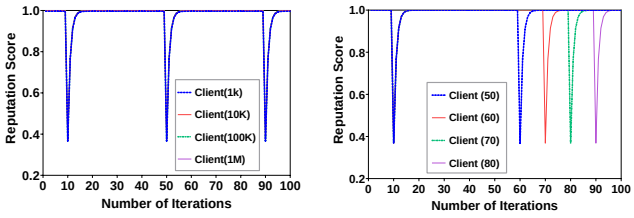


Fig. 5: The decay of reputation score in (left) Client with X model parameters when they attack at 10, 50 and 90 iteration; (right) Client X with 1 million parameters when they attack at 10 and $10 + X$ iteration.

Client X only launches an attack at X iteration. We observe that only one attack would lead to at least a 25.11% relative decrease in reputation score. In Figure 4b, Client X launches an attack at X iteration and keeps attacking in the following iterations. We observe in the end that 80% of their reputation scores are below 0.5, which is approximately half of the reputation score of honest clients, implying that the damage that they can inflict throughout the aggregation process is considerably decreased.

In addition, we consider a scenario in which an attacker spreads out the poisoning over a longer time duration, while using a higher number of model parameters. Figure 5 (left) depicts an attack over 40 iterations under different parameter sizes. Figure 5 (right) depicts an attack with 1 million parameters repeating every 50 to 80 iterations. These figures show that even if attackers spread our poisoning over multiple iterations and then try to recover their reputation score by acting benignly, our detection scheme can still identify them. This is because our attack detection and reputation schemes work in sequence. The attack detection scheme detects malicious updates without considering any reputation scores and rectifies them to mitigate damage. Then, the reputation scheme modifies the reputation scores based on the detection results. Also, attackers that employ a higher number of model parameters suffer a slightly higher reduction of reputation, which is consistent with Corollary 1.

3) Aggregation Algorithm: Algorithm 2 explains our aggregation method based on the attack detection scheme and subjective logic reputation model. First, the server sends all clients the pre-trained global model with initial parameters. Then, using their own data samples, clients train the global model locally and send the trained parameters back to the server. At this point, the server executes the attack detection scheme. In round t, if the n-th update parameter $\mathbf{w}_{i,n}^t$ from

Algorithm 2: Aggregation Algorithm

Server :

Input : $w^0 \leftarrow$ Pretrained Model
κ,η,a,W,c,$s \leftarrow$ Reputation parameters

Output: Global model M_{global} with w^T

for *Iteration* $t \leftarrow 1$ **to** T **do**
  // Broadcast global model to clients
  send(w^{t-1});
  // Wait until all updates arrive
  receive(w^t);
  // Rescale parameters by Algorithm 1
  $\bar{w}^t \leftarrow$ `Rescale`(w^t);
  for $n \leftarrow 1$ **to** N **do**
    for $i \leftarrow 1$ **to** M **do**
      // Compute parameter confidence
      $s^t_{i,n} = Eq\ 4(\bar{w}^t_{i,n})$;
      // Rectify abnormal parameters
      $w^t_{i,n} := Eq\ 5\ (s^t_{i,n}, \delta)$;
      record (P^t_i, N^t_i);
    end for
  end for
  for $i \leftarrow 1$ **to** M **do**
    // Calculate reputation score
    $\tilde{R}^t_i = Eq\ 12(P^t_i,\ N^t_i,\ \kappa,\ \eta,\ a,\ W,\ c,\ s)$;
  end for
  // Normalisation
  $\bar{R}^t \leftarrow$ `Norm`$(\tilde{R}^t)$;
  for $n \leftarrow 1$ **to** N **do**
    // Update the parameters
    $w^t_n := \sum_{i=1}^{M} \frac{\bar{R}^t_i}{\sum_{i=1}^{M} \bar{R}^t_i} w^t_{i,n}$;
  end for
  // Obtain parameters for global model
  $w^t := [w^t_1, \cdots, w^t_n]$;
end for

Client :

for *Client* $i \leftarrow 1$ **to** M **do in parallel**
  receive(w^{t-1});
  // Train local model
  $w^t_i \leftarrow w^{t-1}_i - r\frac{\partial \ell_i(w^{t-1}_i)}{\partial w}$;
  send(w^t_i);
end forpar

the client i has been rectified by the attack detection scheme in Section III-B1 to the median value, the server regards it as a negative observation, whereas no rectification represents a positive observation. Then, the server punishes the negative observation by reducing the corresponding client's reputation. Both types of observations are accumulated through all the N parameters of client i to obtain the reputation value $\tilde{R}^t_i$ in t round for client i so as to all the other clients. The server would conduct Min-Max normalisation to obtain $\bar{R}^t$ after receiving the reputation values $\tilde{R}^t$ of all the clients in t round.

After the server gets correction updates and the normalised reputation of each client, it aggregates the updates using average weighted reputation as the weights to get our global model updates for the current iteration. In this way, even over many training rounds, the attackers are still incapable of shifting parameters notably from the target direction and this ensures the quality of the resulting global model as will be demonstrated experimentally and analytically next.

C. Theoretical Guarantees

We prove the convergence of our reputation-based aggregation method. Our major results are Theorem 1 and Corollary 1, which state that convergence is guaranteed in bounded time. Regarding the performance of our algorithm in terms of metric average accuracy and convergence, we show that it is consistent with our theoretical analysis. We start by stating our assumptions, which are standard and common for such types of results, and per recent works such as [7], [20].

Assumption 1 (Smoothness)**.** *The loss functions are L-smooth, which means they are continuously differentiable and their gradients are Lipschitz-continuous with Lipschitz constant $L > 0$, whereas:*

$$\forall i \in N,\ \forall \mathbf{w}_1, \mathbf{w}_2 \in \mathbf{R}^d$$
$$\|\nabla \mathcal{L}(\mathbf{w}_1)) - \nabla \mathcal{L}(\mathbf{w}_2))\|_2 \leq L \|\mathbf{w}_1 - \mathbf{w}_2\|_2$$
$$\|\nabla \ell(\mathbf{w}_1; \mathcal{D}) - \nabla \ell(\mathbf{w}_2; \mathcal{D})\|_2 \leq L \|\mathbf{w}_1 - \mathbf{w}_2\|_2$$

Assumption 2 (Bounded Gradient)**.** *The expected square norm of gradients $\sqsupseteq$ is bounded:*

$$\forall \mathbf{w} \in \mathbf{R}^d, \exists \mathcal{G}_{\mathbf{w}} < \infty, \mathbb{E}\|\nabla \ell(\mathbf{w}; \mathcal{D})\|_2^2 \leq \mathcal{G}_{\mathbf{w}}$$

Assumption 3 (Bounded Variance)**.** *The variance of gradients $\mathbf{w}$ is bounded:*

$$\forall \mathbf{w} \in \mathbf{R}^d, \exists \mathcal{V}_{\mathbf{w}} < \infty, \mathbb{E}\|\nabla \ell(\mathbf{w}; \mathcal{D}) - \mathbb{E}(\nabla \ell(\mathbf{w}; \mathcal{D})\|_2^2 \leq \mathcal{V}_{\mathbf{w}}$$

Assumption 4 (Convexity)**.** *The loss function $\mathcal{L}(\sqsupseteq)$ are μ-strongly convex:*

$$\exists \mu > 0, \forall \mathbf{w}_1, \mathbf{w}_2 \in \mathbf{R}^d, \nabla \mathcal{L}(\mathbf{w}^*) = 0$$
$$\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_2) \geq \langle \nabla \mathcal{L}(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2 \rangle + \frac{\mu}{2}\|\mathbf{w}_1 - \mathbf{w}_2\|_2^2$$

Suppose the percentage of attackers in the whole clients is p, and all the clients in the system participant every training iteration. r is the learning rate($r < \frac{1}{L}$) and $\hat{Q} = \max\{Q_i\}_{i=1}^M$. $\forall \mathbf{w} \in \mathcal{W}$, we denote

$$\mathbf{m}_i(\mathbf{w}^t) = \begin{cases} * & \text{if } i \in \text{malicious clients} \\ \nabla l_i(\mathbf{w}^t; \mathcal{D}) & \text{if } i \in \text{honest clients} \end{cases}$$

where $*$ stands for an arbitrary value from the malicious clients.

$$\mathbf{m}(\mathbf{w}^t) = \sum_{i=1}^{M} \bar{R}_i \mathbf{m}_i(\mathbf{w}^t)$$
$$s.t.\ \bar{R}_i = \frac{\tilde{R}^t_i}{\sum_{i=1}^{M} \tilde{R}^t_i},\ \sum_{i=1}^{M} \bar{R}_i = 1,\ \bar{R}_i \in (0, 1)$$

Consider the assumptions above and lemmas presented in Appendix A, we have

Theorem 1. *Under Assumptions 1, 2, 3 and 4, $\exists \epsilon > 0$ that:*

$$\sqrt{\frac{d \log(1 + \hat{Q} M L D_\epsilon)}{M(1-p)}} + C\frac{\mathcal{G}_{\mathbf{w}}}{\sqrt{\hat{Q}}} + p \leq \frac{1}{2} - \epsilon \tag{13}$$

After t rounds, Algorithm 2 converges with probability at least $1 - \xi \in \left[1 - \frac{4d}{(1+\hat{Q}MLv)^d}, 1\right)$ as

$$\left\|\mathbf{w}^t - \mathbf{w}^*\right\|_2 \leq (1 - Lr)^t \left\|\mathbf{w}^0 - \mathbf{w}^*\right\|_2 + \frac{\sqrt{N}}{L}\Delta_1 + \frac{1}{L}\Delta_2 \tag{14}$$

where

$$\Delta_1 = \frac{M\left(\varpi(M-1) + \frac{2E}{\sqrt{M\delta}}\right)}{\frac{Wa(M-1)(\kappa N+W)}{(\eta N+W)(\kappa N+Wa)} + 1}$$

$$\Delta_2 = 2\sqrt{2}\frac{1}{M\hat{Q}} + \sqrt{\frac{2}{\hat{Q}}} D_\epsilon V_w \left(\sqrt{\frac{d\log(1+\hat{Q}ML\upsilon)}{M(1-p)}} + C\frac{\mathcal{G}_w}{\sqrt{\hat{Q}}} + p\right)$$

$$D_\epsilon := \sqrt{2\pi}\exp\left(\frac{1}{2}\left(\Phi(1-\epsilon)\right)^2\right)$$

with $\Phi(\cdot)$ being the cumulative distribution function of Wald distribution.

Corollary 1. *Continuing with Theorem 1, when the iterations satisfy $t \geq \frac{1}{Lr}\log\left(\frac{L}{\sqrt{N}\Delta_1+\Delta_2}\left\|\mathbf{w}^0 - \mathbf{w}^*\right\|_2\right)$, $\exists \xi \in \left(0, \frac{4d}{(1+\hat{Q}ML\upsilon)^d}\right]$, we have:*

$$\mathbb{P}\left(\left\|\mathbf{w}^t - \mathbf{w}^*\right\|_2 \leq \frac{2\sqrt{N}}{L}\Delta_1 + \frac{2}{L}\Delta_2\right) \geq 1 - \xi$$

Remark 1. *Due to*

$$\Delta_1 := \mathcal{O}\left(\frac{\varpi}{a\kappa WN} + \frac{1}{\kappa N} + \frac{1}{\sqrt{MN\delta}}\right)$$

and

$$\Delta_2 := \mathcal{O}\left(\frac{1}{\hat{Q}} + \frac{p}{\sqrt{\hat{Q}}} + \frac{1}{\sqrt{\hat{Q}M}}\right)$$

Based on Corollary 1, we achieve an error rate:

$$\mathcal{O}\left(\frac{\varpi}{a\kappa W\sqrt{N}} + \frac{1}{\kappa\sqrt{N}} + \frac{1}{\sqrt{M\delta}} + \frac{1}{\hat{Q}} + \frac{p}{\sqrt{\hat{Q}}} + \frac{1}{\sqrt{\hat{Q}M}}\right)$$

we observe the experimental results in Figure 3 and 11 of Sections III and IV respectively, when varying the parameters of N, p, a and κ, results are consistent with this error rate.

Remark 2. *Derived from the Corollary 1 and Remark 1, there is a trade-off problem between convergence speed and error rate according to the level of reward κ and punishment η from the reputation model. This trade-off problem is mainly based on the fact that if the model penalises the bad behaviours of clients heavily, it would decrease their reputation dramatically so the model would take a longer time to converge. On the other hand, mitigating the punishment to increase the reward, would lead to an increase in the error rate.*

IV. Performance Evaluation

The objectives of our experimental evaluation are the following: (a) evaluate the performance of our aggregation method against other state-of-art robust aggregation methods, (b) benchmark it in three different scenarios, namely, no attack, label flipping attack, and backdoor attack, (c) do so using a text based real-world dataset of sensitive categories from [4] to which we will henceforth refer to as SURL, and finally (d) show that our experimental result are consistent with our previous theoretical analysis.

A. Experimental Setup

1) Datasets: The SURL dataset comes from a crowdsourcing taxonomy in the Curlie project [32], containing six categories of URLs: five sensitive categories (Health, Politics, Religion, Sexual Orientation, Ethnicity) and one for non-sensitive URLs, with a total of 442,190 webpages. The number of URLs in sensitive and non-sensitive categories are equally balanced. Each sample contains content, metadata and a class label of the webpage. For the SURL text classification task, we train a neural network with three fully connected layers and a final softmax output layer, same as in the evaluated methods [16], [20]. Furthermore, in order to fulfil the fundamental setting of an heterogeneous and unbalanced dataset for FL, we sample u_k from a Dirichlet distribution [18] with the concentration parameter $\iota = 0.9$ as in [12], which controls the imbalance level of the dataset, then assigns a $u_{k,i}$ fraction of samples in class k to client i, with the intention of generating non-IID and unbalanced data partitions. As a sanity check, we also tested our reputation scheme on a different classification task involving images and got consistent results as those we got for sensitive content (see Appendix C).

2) Threat Model: We consider the following threat model.
Attack capability: In the FL setting, the malicious clients have complete control over their local training data, training process and training hyper-parameters, e.g., the learning rate, iterations and batch size. They can pollute the training data as well as the parameters of the trained model before submitting it to the server but cannot impact the training process of other clients. We follow the common practice in the computer security field of overrating the attacker's capability rather than underrating it, so we limit our analysis to worst-case scenarios. There, an attacker has perfect knowledge about the learning algorithm, the loss function, the training data and is able to inspect the global model parameters. However, attackers would still have to train with the model published by the server, thus complying with the prescribed training scheme by FL to their local data. Furthermore, the percentage of byzantine clients p is an important factor that determines the level of success for the attack. We assume that the number of attackers is less than the number of honest clients, which is a common setting in similar methods [16], [20] to the ones we evaluate and compare our method with.
Attack strategy: We focus on two common attack strategies for sensitive context classification, namely, (i) label flipping attack [24] and (ii) backdoor attack [12]. Comparing to other attacks, for example model poisoning attack [29], [63], these two data poisoning attacks are more likely to be carried out by real users in the real world via our browser extension described in Section V, since polluting data is easier than manipulating model updates using the browser extension. Note that privacy attacks including membership inference attack [65] and property inference attack [64], are out of the scope of this paper, but form part of our ongoing and future work.

In a label flipping attack, the attacker flips the labels of training samples to a targeted label and trains the model accordingly. In our case, the attacker changes the label of "Health" to "Non-sensitive". In a backdoor attack, attackers inject a designed pattern into their local data and train these manipulated data with clean data, in order to develop a local model that learns to recognise such pattern. We realise

backdoor attacks inserting the top 10 frequent words with their frequencies for the "Health" category. Therein the backdoor targets are the labels "non-sensitive". A successful backdoor attack would acquire a global model that predicts the backdoor target label for data along with specific patterns.

For both attacks, instead of a single-shot attack where an attacker only attacks in one round during the training, we enhance the attacker by a repeated attack schedule in which an attacker submits the malicious updates in every round of the training process. Also, we evaluate a looping attack where the attackers spreads out poisoning every 30 epochs for the label flipping attack based on Figure 5. It is important to note that even if the attackers have full knowledge of our method, they would still be unable to mount smarter attacks that would try to maximise the damage caused while minimising their reputation drop. This is because the attackers are unaware and cannot compute their reputation score since the latter is computed at the server and requires input from all clients. Moreover, we allow extra training epochs for an attacker, namely, being able to train the local models with 5 more epochs as in [12].

3) Evaluated Aggregation Methods: We compare the performance of our aggregation method against the existing state-of-the-art in the area FedAvg [5], as well as against popular robust aggregation methods such as Coordinate-wise median [20], Trimmed-mean [20], FoolsGold [8], [15], Residual-based re-weighting [16], and FLTrust [60].

FedAvg is a FL aggregation method that demonstrates impressive empirical performance in non-adversarial settings [5]. Nevertheless, even a single adversarial client could control the global model in FedAvg easily [27]. This method averages local model updates of clients as a global model update weighted by the fraction of training samples size of each client compared to total training samples size. We use it as baseline evaluation to assess the performance of our method.
Median is using coordinate-wise median for aggregation. After receiving the updates in round t, the global update is set equal to the coordinate-wise median of the updates, where the median is the 1-dimensional median.
Trimmed-mean is another coordinate-wise mean aggregation technique that requires prior knowledge of the attacker fraction β, which should be less than half of the number of model parameters. For each model parameter, the server eliminates the highest and lowest β values from the updates before computing the aggregated mean with remaining values.
FoolsGold presents a strong defence against attacks in FL, based on a similarity metric. Such approach identifies attackers based on the similarity of the client updates and decreases the aggregate weights of participating parties that provide indistinguishable gradient updates frequently while keeping the weights of parties that offer distinct gradient updates. It is an effective defence for sybil attacks but it requires more iterations to converge to an acceptable accuracy.
Residual-based re-weighting weights each local model by accumulating the outcome of its residual-based parameter confidence multiplying the standard deviation of parameter based on the robust regression through all the parameters of this local model. In our reputation-based aggregation method, we implement the same re-weighting scheme IRLS [10] as residual-based aggregation, but choose the collection of reputation as the weights of clients' local models.
FLTrust establishes trust in the system by bootstrapping it via the server, instead of depending entirely on updates from clients, like the other methods do. The server obtains an initial server model trained on clean root data. Then, depending on the cosine similarity of the server model and each local model, it assigns a trust score to each client in each iteration.

4) Performance Metrics: We use the average accuracy (Avg-ACC) of the global model to evaluate the result of the aggregation defence for the poisoning attack in which attackers aim to mislead the global model during the testing phase. The accuracy is the percentage of testing examples with the correct predictions by the global model in the whole testing dataset, which is defined as:

$$\text{Avg-ACC} = \frac{\text{\# correct predictions}}{\text{\# testing samples}}$$

In addition, there is existence of targeted attacks that aim to attack a specific label while keeping the accuracy of classification on other labels unaltered. Therefore, instead of Avg-ACC, we choose the attack success rate (ASR) to measure how many of the samples that are attacked, are classified as the target label chosen by a malicious client, namely:

$$\text{ASR} = \frac{\text{\# successfully attacked samples}}{\text{\# attacked samples}}$$

A robust federated aggregation method would obtain higher Avg-ACC as well as a lower ASR under poisoning attacks. An ideal aggregation method can achieve 100% Avg-ACC and has the ASR as low as the fraction of attacked samples from the target label.

5) Evaluation Setup: For the malicious attack, we assume that 30% of the clients are malicious as in [27], which is also a common byzantine consensus threshold for resistance to failures in a typical distributed system [6]. For the server-side setting, in order to evaluate the reliability of the local model updates sent by the client to the server, we assume that the server has the ability to look into and verify the critical properties of the updates from the clients before aggregating.

Also, we only consider FL to be executed in a synchronous manner, as most existing FL defences require [7], [20], [27], [29]. For all the above aggregation methods under attack, we perform 100 iterations using the SURL dataset with a batch size of 64 and 10 clients. Furthermore, we evaluate our method for increasing numbers of clients. These settings are inline with existing state-of-the-art methods for security in FL [12], [16], [20] More details related to the training setting are presented in Appendix B.

B. Convergence and Accuracy

In Figure 6 (left), we analyse the performance of our method in the no attack scenario and compare the convergence and accuracy of our method with others during training. We show the training loss (left axis) and average accuracy (right axis) during 100 training iterations for 7 methods.

Our aggregation starts with the lowest training loss and maintains it throughout the training process. It only takes 24 iterations to achieve 82% accuracy and then converge to 82.13%, which represents a 2.7× faster converge rate than FedAvg. In comparison, Residual-based and Trimmed-mean

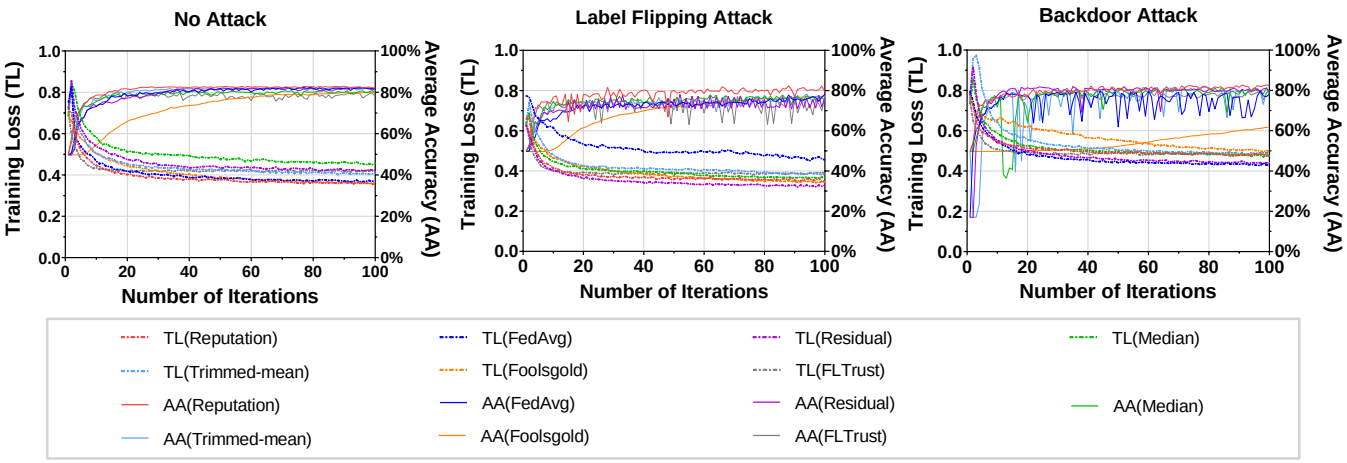


Fig. 6: Training Loss (TL) and Average Accuracy (AA) for 100 epochs of Reputation-based, FedAvg, Residual-based, Median, Trimmed-mean, FoolsGold and FLTrust methods in SURL Dataset under no attack (left) scenario, under label flipping attack (middle) and backdoor attack (right) scenarios with 30% malicious clients.

have almost identical training loss and take 52 and 47 iterations to reach 82% accuracy and practically converge to 81.79% and 80.76% respectively, which is 2.2× and 2× slower than our reputation method. Median reaches 81% at 83 rounds and after that converges to 79.94%, which amounts to a 3.6× slower converge rate than our method. Especially, Foolsgold and FLTrust are slow to converge and do not converge within 100 iterations, so our convergence rate is at least 4.2× better than FoolsGold and FLTrust. This demonstrates that our reputation model benefits from convergence speed and accuracy performance. This is because our reputation scheme assigns higher weight to more reliable clients when there is no ongoing attack, which generates more consistent updates thereby accelerating the convergence.

C. Resilience to Attacks

We begin by analysing the performance with a static percentage (30%) of attackers, and then move on to the performance with a varying percentage of attackers under label flipping and backdoor attacks.

1) Label Flipping Attack: **Static percentage of attackers:** Figure 6 (middle) shows the convergence of mentioned methods under label flipping attack. Our method converges 1.8× to 2× faster than all competing state-of-the-art methods under attack, enlarging its performance benefits compared to the no attack scenario. In addition, our method outperforms competing methods by at least 1.4% in terms of accuracy.

Varying the percentage of attackers: Here we analyse the impact on our aggregation method as the proportion of attackers increases. Figure 7 (left) shows the change of performance metrics for varying percentage of attackers for seven evaluated methods When the percentage of attackers p ranges from 10% to 50%, our method is resistant against label flipping attacks with a small loss in accuracy and a consistent attack success rate of all the methods. As p approaches 50%, FedAvg, Residual-based, Median and FLTrust defences become ineffective in mitigating the attack, and correspondingly their Avg-ACC decreases linearly. Moreover, under label flipping attack during the whole process, our reputation-based method has the highest accuracy outperforming other methods by 1% to 23.1%. At the same time it has the lowest ASR. The average ASR of other methods are at least 82.8% higher than ours.

2) Backdoor Attack: **Static percentage of attackers:** Figure 6 (right) shows the convergence of mentioned methods under backdoor attack. Same as in no attack and label flipping attack scenario, our method converges 1.6× to 2.4× faster than all competing state-of-the-art methods. In addition, our method outperforms competing methods by 3.5% to 33.6% in terms of classification accuracy.

Varied percentage of attackers: We examine the scenario in which the percentage of attackers increases. Figure 7 (right) shows the performance for the seven evaluated methods under backdoor attack when varying the percentage of attackers p from 10% to 50%. Figure 7 (right) demonstrates that under backdoor attack, our reputation-based method has a consistent accuracy throughout the process with the lowest attack success rate, whereas the average ASR of other methods is at least 72.3% higher than ours. As p changes, the ASR of the Residual-based, Median, and Foolsgold methods increase linearly. Although FLTrust has a stable ASR, it increases by a factor of 1.39 when p reaches 50%.

First, we evaluate a varying compromise rate for the label flipping and backdoor attacks using our reputation-based method. For the label-flipping attack, we vary the percentage of the flipped label poisoned by attackers from 10% to 90%. Also, for the backdoor attack, we vary the number of top frequent words inserted as the trigger pattern, from 5 to 25. The remaining settings are the same as in previous experiments. Figure 8 plots the ACC and ASR when varying the compromise rate for both attacks. Figure 8 (left) shows that when the percentage of the poisoned sample is increased, it leads to the decrease of the accuracy of the model and to a slight increase of ASR. Figure 8 (right) shows that when we increase the number of frequent words from 5 to 20, the ASR remains unaffected. When the frequent words exceed 25, the attack becomes less stealthy and thus can be more easily detected resulting in a lower ASR.

We also evaluate the performance of our method in terms of the number of participating clients. With a 30% compromise rate, we expand the number of clients from 10 to 200. Our

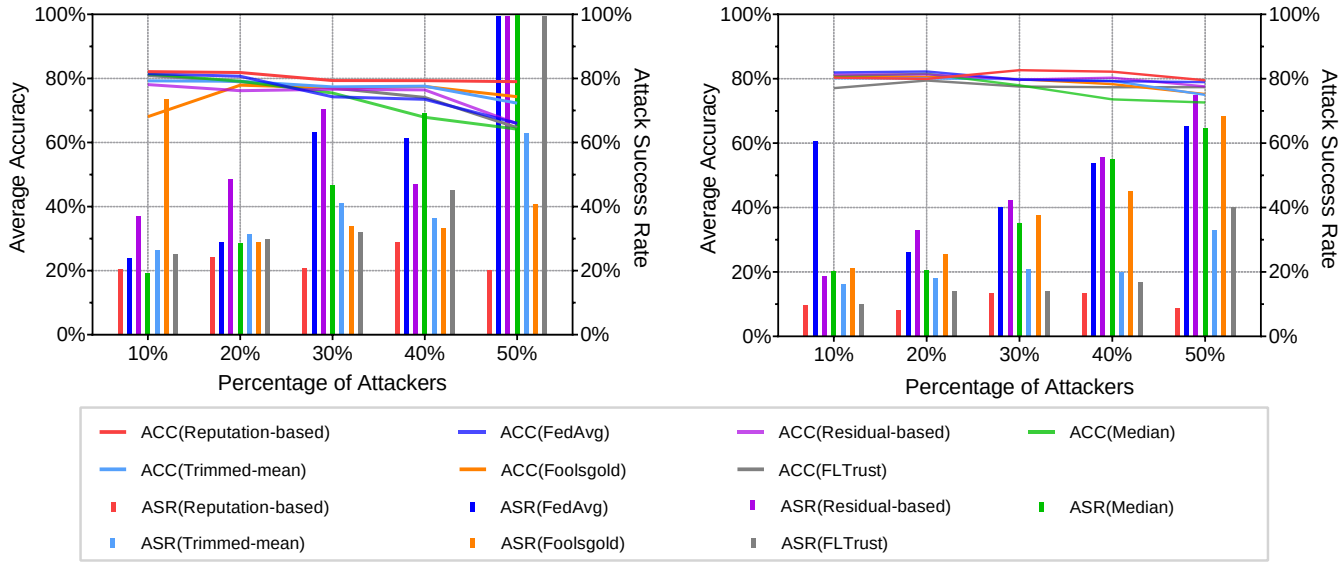


Fig. 7: Average accuracy (ACC) and attack success rate (ASR) for varying percentage of attackers from 10% to 50% under label flipping (left) and backdoor (right) attack for Reputation-based, FedAvg, Residual-based, Median, Trimmed-mean, FoolsGold and FLTrust methods in SURL Dataset.

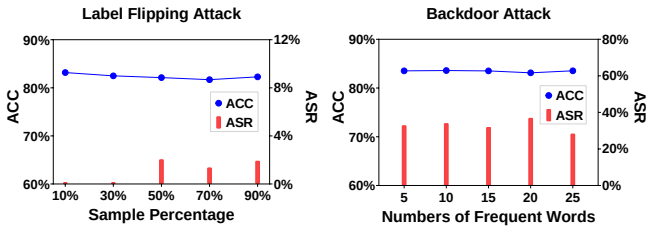


Fig. 8: ACC and ASR as we vary the percentage of flipped label from 10% to 90% (left) for label flipping attack, and the number of the frequent words as the trigger pattern from 5 to 25 for backdoor attack(right).

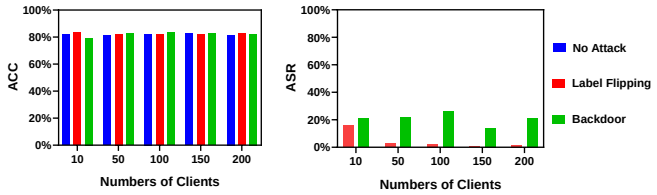


Fig. 9: Average accuracy (ACC) and attack success rate (ASR) for varying the number of clients from 10 to 200 under label flipping and backdoor attack with 30% malicious clients.

method performs consistently for a larger number of clients, as seen by the stable ACC and ASR as the number of clients grows in Figure 9.

3) Analysis of Attacks: Finally, instead of repeating the attack at every epoch, the attacker stretches poisoning across 30 epochs in our study of the looping attack. The performance of the looping attack is seen in Figure 10. As expected, the looping attack is not as effective as the repeated attack that we previously assessed. All the methods manage to defend it with low ASR, and our method still has the greatest accuracy.

4) Evaluation Results: In the no attack scenario, we observe (i) Our method converges $2\times$ to $4.2\times$ faster than all competing state-of-the-art methods. (ii) Our method is at least as good or outperforms competing methods in terms of classification accuracy. The above validates that our reputation scheme is helpful even in the no attack scenario. This is due to the fact that in our algorithm we give higher weights to the clients with high-quality updates, as illustrated in Figure 12, causing the model to converge rapidly and retain consistent accuracy. In addition, even under the two different attacks, our method:

- converges $1.6\times$ to $2.4\times$ faster than all competing state-of-the-art methods.
- provides the same or better accuracy than competing methods.
- yields the lowest ASR compared to all other methods, with the average ASR of them being at least 72.3% higher than ours.

We obtained comparable findings for the evaluation of the aforementioned methods on 100 clients, as presented in Appendix D. Furthermore, the result is consistent with the theoretical analysis: as p increases, so does the error rate.

D. Stability of Hyper-parameters

We employ four hyper-parameters in our reputation model: rewarding weight κ, prior probability a, time decay parameter c and window length s. As shown in Remark 1, c and s do not affect the performance of our model, we only consider hyper-parameters κ and a, where κ controls the reward weight to positive observations and a controls the fraction of uncertainty converted to belief. To demonstrate the impact of these two hyper-parameters of our reputation model, we grid search κ in $[0.1, 0.2, 0.3, 0.4]$ and a in $[0.1, 0.3, 0.5, 0.7, 0.9]$. The setup is the same as on SURL dataset under label flipping attack. The ultimate accuracy of stability of reputation-based aggregation

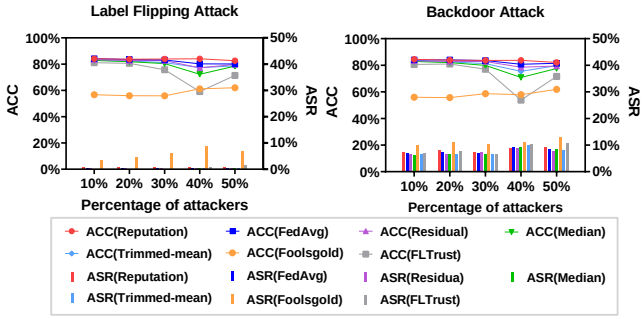
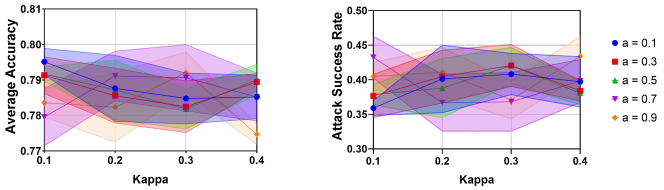


Fig. 10: Average accuracy (ACC) and attack success rate (ASR) for varying percentage of attackers from 10% to 50% under label flipping (left) and backdoor (right) attack with a looping attack in which an attacker attacks every 30 epochs.



(a) Accuracy (b) Attack Success Rate

Fig. 11: Average accuracy and attack success rate as we vary rewarding weight κ and prior probability a.

are shown in Figure 11. Note that these results are tested for the label flipping attack and they hold according to theory also for backdoor.

The result in Figure 11 demonstrates that our approach is very stable and efficient in terms of hyper-parameter selection, and it achieves a high degree of precision. Furthermore, the result is compatible with the theoretical analysis in Section III-C.

E. Comparison against a residual-based method

To demonstrate how our method improves the residual-base method by assigning the aggregation weights based on reputation, we consider a scenario with 10 clients in the FL system, 8 of which are malicious. The training lasts 10 communication rounds during which attackers carry out the backdoor attack. The remaining settings are the same as the default. Results are shown in Figure 12, in which the first two clients are benign, and the rest are malicious. We observe that for our reputation method the aggregation weights of malicious clients, which are their reputations, are rectified to 0 since the second round, demonstrating that our method is successful in eradicating their influence. On the other hand, the aggregate weights of malicious clients in residual-based methods, which are calculated by multiplying the parameter confidence by its standard deviation, are nearly similar and non-zero. This is because repeated median regression seldom yields 0 for the parameter confidence, which causes practically non-zero weights to be assigned to malicious clients by residual-based methods. To address this issue, the reputation model uses positive and negative observations that introduce rewards and punishments to assign divergent weights to clients. As a result, benign clients are given higher weights whereas malicious clients are eliminated from the aggregation.

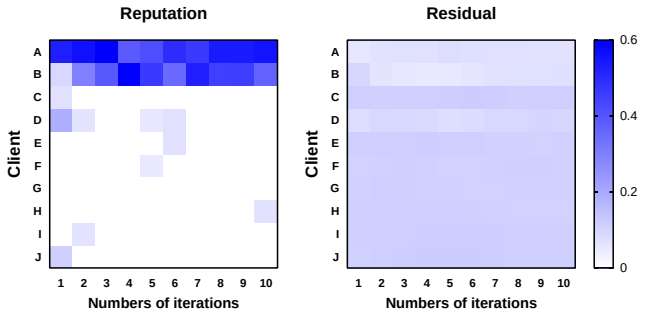


Fig. 12: The aggregation weights of clients from our reputation-based (left) and residual-based method (right) for 10 communication rounds under label flipping attack with 80% attackers.

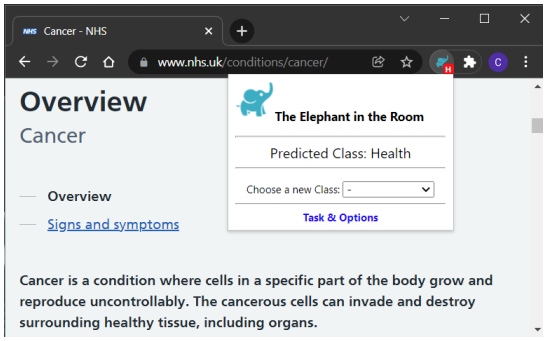


Fig. 13: *EITR* extension in action. The letter "H" inside the red frame at the bottom right of the extension's icon indicates that a health-related page has been detected.

V. The *EITR* System

In this section, we provide a high level description of our *EITR* [67] system (standing for "Elephant In the Room" of privacy). We then present some preliminary results with real users demonstrating the ability of the system to quickly learn how to classify yet unseen sensitive content, in our case COVID-19 URLs pertaining to the category Health, even in view of inaccurate user input. The system is currently being used as a research prototype to evaluate the robustness of our algorithm in a simple real-world setting. A full in depth description of the system and its performance with more users and more intricate settings, including adoption, incentives, and HCI issues, over a longer time period is the topic of our ongoing efforts and will be covered by our future work.

A. System Architecture and Implementation

The *EITR* system is based on the client-server model. The back-end server is responsible to distribute the initial classification model and the consequent updated model(s) to the clients and receive new annotations from the different clients of the system. The client is in the form of a web browser extension that is responsible to fetch and load the most recent global classification model to the users' browser from the back-end server. The loaded model can then be used to label website in real time into the 5 different sensitive topics as defined by GDPR, i.e., Religion, Health, Politics, Ethnicity and Sexual Orientation. Next, we provide more details for each part of

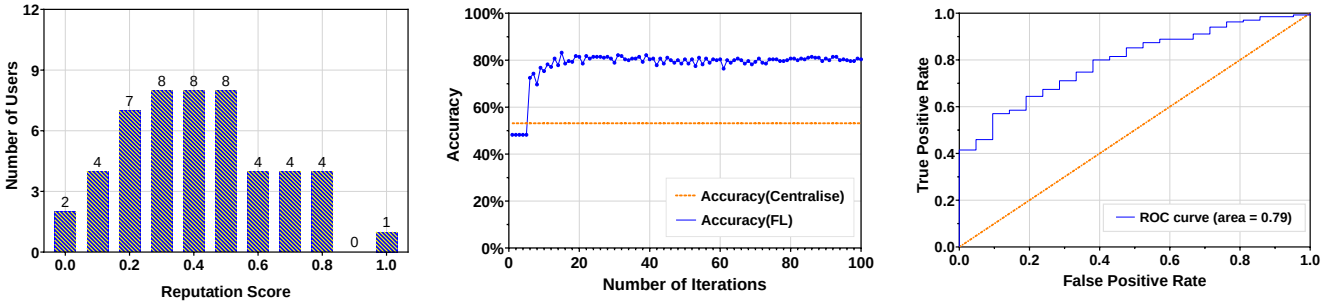


Fig. 14: Results of real-user experiment for COVID-19 related URLs with 50 users over 100 iterations. (left): the reputation score of the real users at the end of experiment, (middle): the accuracy of the centralised classifier and the global model of our reputation-based FL approach for COVID-19 URLs, and (right): the ROC curve of real-user experiment with 0.79 area under the ROC curve (AUC).

the system.

Back-end server: The back-end server is written in JavaScript using the node.js Express [40] framework. To build the initial classification model we use the dataset provided by Matic et al. [4], and the TensorFlow [41] and Keras [42] machine learning libraries. The final trained model is then exported using the TensorFlow.js [43] library in order to be able to distribute it to the system's clients (browser extension). The back-end server also includes additional functionalities such as the creation and distribution of users' tasks, i.e., a short list of URLs that the users need to visit and annotate, and an entry point that collects the resulting users annotations during the execution of the task.

Web browser extension: Currently the browser extension only supports the Google Chrome browser and is implemented in JavaScript using the Google Chrome Extension APIs [45]. To handle the classification model the extension utilises the TensorFlow.js [43] library to load, use, and update the model. The main functionality of the extension is to classify the visited website in real time and provide information to the user related to the predicted class as depicted in Figure 13. The website classification is based on the metadata (included in the website `<head>` HTML tag) and the visible text of the website. The extension also allows users to provide their input related to their agreement or disagreement with the predicted class using a drop down list as depicted in Figure 13 with the label *"Choose a new Class"*.

B. Real Users Experimental Setup

The goal of the real-user experiment is to evaluate our federated reputation-based method on real user activity (instead of systematic tests), and demonstrate that even with real users with different comprehensions of the definition of sensitive information, our method can learn new content fast and achieve higher accuracy than centralised classifiers, which is compatible with our simulation experiment.

For the setup, the participant is directed to visit the experiment website that provides the necessary information and instructions on the goal of our study, the definition of sensitive information provided by the current GDPR and how to participate in our study. In order to have access to the browser extension and the installation instructions the user must in advance give explicit consent and accept the data privacy policy. Upon successful installation of the extension, new users are asked to provide a valid email address (to contact them for the reward) and then receive their task, a list of 20 URLs, that they need to visit and provide their labels in order to successfully complete their task. The list of 20 URLs is sampled by the Dirichlet distribution with $\iota = 0.9$ for each participant from a database, which includes 300 URLs with sensitive and non-sensitive content related to COVID-19.

Ethical Considerations: We have ensured compliance with the GDPR pertaining to collecting, handling, and storing data generated by real users. To that end, we have acquired all the proper approvals from our institutions. Furthermore, the participants are directed to visit a pre-selected set of URLs selected by us to avoid collecting the actual visiting patterns of our users. In addition, the user input is only collected *if and only if* the user explicitly provide input to the drop-down list labelled *"Choose a new Class"* to avoid collecting the visiting patterns of the user accidentally while they are executing their tasks. Finally, we only use the users' email address to contact them for the reward. The mapping between the user input and their email address is based on a random identifier that is generated during the installation time of the extension.

C. Validation with Real Users

Data collection: We had 50 users participating in our experiment. In order to evaluate our reputation-based FL method using real-user data, we define a methodology to label ground truth on COVID-19 related sites.

Ground truth methodology: To set the ground truth for COVID-19 sites related to our sensitive or non-sensitive labels, we create a database of 100 websites, which we collect by searching on Google with the query "sensitive websites about COVID-19" and choose the top 100 sites returned from the query. Then, four experts in the privacy field, independently annotate them based on their professional expertise in order to achieve an agreement on whether each of those sites included sensitive or non-sensitive content.

Ground truth annotation: In order to evaluate the annotation of the 100 websites from human experts, we calculate the inter-rater agreement among them using Fleiss Kappa [61]. We obtain 0.56 of Fleiss Kappa, which is an acceptable agreement because the values of Fleiss Kappa. above 0.5 are regarded as good. Furthermore, given that COVID-19 is a controversial issue, it is difficult for humans to agree on what constitutes sensitive content relating to it. Even though, we still attain a

valid ground truth of 85 items belonging to the health sensitive category with agreement ratings of at least 0.5. Note that we also classify the above 100 websites using the centralised classifier proposed in [4] and get only 53.13% accuracy.

Result with real users: Figure 14 shows the results of accuracy and reputation score with 50 real users in the experiment. Figure 14 (left) shows that the majority of users have reputation scores falling in the intermediate range, with some having a very high reputation and a few having a very low reputation. This indicates the divergence of the user's interpretation of the sensitive information as we expect. In Figure 14 (middle) we compare the accuracy of the centralised classifier and the global model of our reputation-based FL approach for COVID-19 URLs. Despite the diversity of reputation scores of real users, our method converges as rapidly as in simulation and achieves an average accuracy of 80.36%, thereby verifying the quick convergence and high accuracy results presented in the previous sections. Figure 14 (right) shows that the ROC curve in real-user experiment yielded 0.79 AUC. Our result is acceptable in this scenario because most existing FL techniques are designed to minimise the conventional cost function and are not optimal for optimising more appropriate metrics for imbalanced data, such as AUC [62].

As we observe, with real users holding our method achieve a good performance. This means that, as new sensitive content appears and/or is defined by GDPR or new upcoming legislation, we will be able to continue training our FL model for this type of task with quick convergence and good accuracy. The empirical results in Figure 14 (middle) also shows that there is a quick convergence to the accuracy's stable value within a small number of iterations (around 30), in line with the theoretical results in Section III.

VI. Conclusion

In this paper we have shown how to use federated learning to implement a robust to poisoning attacks distributed classifier for sensitive web content. Having demonstrated the benefits of our approach in terms of convergence rate and accuracy against state-of-the-art approaches, we implemented and validated it with real users using our *EITR* browser extension. Collectively, our performance evaluation has showed that our reputation-based approach to thwarting poisoning attacks consistently converges faster than the state-of-the-art while maintaining or improving the classification accuracy.

We are currently working towards disseminating *EITR* to a larger user-base and using it to classify additional sensitive and non-sensitive types of content. This includes but it is not limited to categories defined by the users themselves for different purposes, not necessarily related to sensitive content, as well as evaluating additional attacks and threat models under our subjective-logic reputation scheme for FL. In turn, our approach can support other FL models going beyond sensitive content classification in future work.

Acknowledgements

We thank the EITR volunteers for their help. This work and dissemination efforts were supported in part by the European Commission under DataBri-X project (101070069), the TV-HGGs project (OPPORTUNITY/0916/ERCCoG/0003) co-funded by the European Regional Development Fund and the Republic of Cyprus through the Research and Innovation Foundation, and the European Research Council (ERC) Starting Grant ResolutioNet (ERC-StG-679158).

References

[1] McMahan, H., Ramage, D., Talwar, K. & Zhang, L. Learning differentially private recurrent language models. *Proceedings of ICLR*. (2017)

[2] Mammen, P. Federated Learning: Opportunities and Challenges. *ArXiv Preprint ArXiv:2101.05428*. (2021)

[3] Jøsang, A. Subjective logic. (Springer,2016)

[4] Matic, S., Iordanou, C., Smaragdakis, G. & Laoutaris, N. Identifying Sensitive URLs at Web-Scale. *Proceedings of The ACM Internet Measurement Conference*. pp. 619-633 (2020)

[5] McMahan, B., Moore, E., Ramage, D., Hampson, S. & Arcas, B. Communication-Efficient Learning of Deep Networks from Decentralized Data. *Artificial Intelligence And Statistics*. pp. 1273-1282 (2017)

[6] Castro, M., Liskov, B. Practical Byzantine Fault Tolerance. *USENIX OSDI*. **99**, 173-186 (1999)

[7] Xie, C., Koyejo, S. & Gupta, I. Zeno: Distributed Stochastic Gradient Descent with Suspicion-based Fault-tolerance. *International Conference On Machine Learning*. pp. 6893-6901 (2019)

[8] Fung, C., Yoon, C. & Beschastnikh, I. The Limitations of Federated Learning in Sybil Settings. *23rd International Symposium On Research In Attacks, Intrusions And Defenses (RAID 2020)*. pp. 301-316 (2020,10), https://www.usenix.org/conference/raid2020/presentation/fung

[9] Josang, A., Hayward, R. & Pope, S. Trust Network Analysis with Subjective Logic. *Proceedings of The Twenty-Ninth Australasian Computer Science Conference (ACSW 2006)*. pp. 85-94 (2006)

[10] Wilcox, R. Introduction to robust estimation and hypothesis testing. (Academic press,2011)

[11] Krizhevsky, A., Hinton, G. & Others Learning Multiple Layers of Features from Tiny Images. University of Toronto Technical Report, 2009.

[12] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D. & Shmatikov, V. How to Backdoor Federated Learning. *International Conference on Artificial Intelligence And Statistics*. pp. 2938-2948 (2020)

[13] Konečny, J., McMahan, H., Yu, F., Richtárik, P., Suresh, A. & Bacon, D. Federated Learning: Strategies for Improving Communication Efficiency. *Proceedings of NIPS*. (2016)

[14] Siegel, A. Robust Regression using Repeated Medians. *Biometrika*. **69**, 242-244 (1982)

[15] Fung, C., Yoon, C. & Beschastnikh, I. Mitigating Sybils in Federated Learning Poisoning. *ArXiv Preprint ArXiv:1808.04866*. (2018)

[16] Fu, S., Xie, C., Li, B. & Chen, Q. Attack-resistant Federated Learning with Residual-based Reweighting. *AAAI Workshops: Towards Robust, Secure And Efficient Machine Learning*. (2021)

[17] Sun, Z., Kairouz, P., Suresh, A. & McMahan, H. Can you Really Backdoor Federated Learning?. *The 2nd International Workshop on Federated Learning For Data Privacy And Confidentiality, in Neural Information Processing Systems*. (2019)

[18] Minka, T. Estimating a Dirichlet Distribution. (Technical report, MIT, 2000)

[19] He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*. pp. 770-778 (2016)

[20] Yin, D., Chen, Y., Kannan, R. & Bartlett, P. Byzantine-robust Distributed Learning: Towards Optimal Statistical Rates. *International Conference On Machine Learning*. pp. 5650-5659 (2018)

[21] Minsker, S. Geometric Median and Robust Estimation in Banach Spaces. *Bernoulli*. **21**, 2308-2335 (2015)

[22] Karimireddy, S., He, L. & Jaggi, M. Learning from History for Byzantine Robust Optimization. *International Conference On Machine Learning*. pp. 5311-5319 (2021)

[23] Bubeck, S. Convex Optimization: Algorithms and Complexity. *Foundations and Trends in Machine Learning, 8 (3-4)*. (2014)

[24] Barreno, M., Nelson, B., Joseph, A. & Tygar, J. The Security of Machine Learning. *Machine Learning*. **81**, 121-148 (2010)

[25] Deng, J., Dong, W., Socher, R., Li, L., Li, K. & Fei-Fei, L. Imagenet: A Large-scale Hierarchical Image Database. *2009 IEEE Conference On Computer Vision And Pattern Recognition*. pp. 248-255 (2009)

[26] Bhagoji, A., Chakraborty, S., Mittal, P. & Calo, S. Analyzing Federated Learning through an Adversarial Lens. *International Conference On Machine Learning*. pp. 634-643 (2019)

[27] Blanchard, P., El Mhamdi, E., Guerraoui, R. & Stainer, J. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. *Proceedings Of The 31st International Conference On Neural Information Processing Systems*. pp. 118-128 (2017)

[28] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. & Lerer, A. Automatic Differentiation in Pytorch. (2017)

[29] Fang, M., Cao, X., Jia, J. & Gong, N. Local Model Poisoning Attacks to Byzantine-robust Federated Learning. *29th USENIX Security Symposium*. pp. 1605-1622 (2020)

[30] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H., Patel, S., Ramage, D., Segal, A. & Seth, K. Practical Secure Aggregation for Privacy-preserving Machine Learning. *Proceedings of The 2017 ACM SIGSAC Conference On Computer And Communications Security*. pp. 1175-1191 (2017)

[31] El Mahdi, E. M., Guerraoui, R., Rouault, S. The Hidden Vulnerability of Distributed Learning in Byzantium. *International Conference On Machine Learning*. pp. 3521-3530 (2018)

[32] Curlie.org Curlie - The Collector of URLs. (2019), https://curlie.org/

[33] Commission, E. Data protection in the EU, The General Data Protection Regulation (GDPR); Regulation (EU) 2016/679. (2018), https://ec.europa.eu/%0Ainfo/law/law-topic/data-protection/

[34] California, S. California Consumer Privacy Act - Assembly Bill No. 375. (2018), https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375

[35] Government of Canada. Amended Act on The Personal Information Protection and Electronic Documents Act. (2018), https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/

[36] Government of Israel. Protection of privacy regulations (data security) 5777-2017. (2018), https://www.gov.il/en/Departments/legalInfo/data_%0Asecurity_regulation/

[37] European Commission. Personal Information Protection Commission, J. Amended Act on the Protection of Personal Information. (2017), https://www.ppc.go.jp/en/

[38] Australian Information Commissioner. Australian Privacy Principles guidelines; Australian Privacy Principle 5 - Notification of the collection of personal information. (2018), https://www.oaic.gov.au/agencies-and-organisations/

[39] Kang, J., Xiong, Z., Niyato, D., Xie, S. & Zhang, J. Incentive Mechanism for Reliable Federated Learning: A Joint Optimization Approach to Combining Reputation and Contract Theory. *IEEE Internet Of Things Journal*. **6**, 10700-10714 (2019)

[40] Expressjs.com Express - Fast, unopinionated, minimalist web framework for Node.js. (2021), https://expressjs.com/

[41] TensorFlow.org TensorFlow - An end-to-end open source machine learning platform.. (2021), https://www.tensorflow.org/

[42] Keras.io Keras - Simple. Flexible. Powerful. (2021), https://keras.io/

[43] Tensorflow.org TensorFlow.js is a library for machine learning in JavaScript. (2021), https://www.tensorflow.org/js

[44] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečny, J., Mazzocchi, S., McMahan, H. Towards Federated Learning at Scale: System Design. *Proceedings of MLSys*. (2019)

[45] Google Google Chrome Extension APIs. (2021), https://developer.chrome.com/docs/extensions/reference/

[46] Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C. & Ramage, D. Federated Learning for Mobile Keyboard Prediction. *ArXiv Preprint ArXiv:1811.03604*. (2018)

[47] Feng, J., Rong, C., Sun, F., Guo, D. & Li, Y. PMF: A Privacy-preserving Human Mobility Prediction Framework via Federated Learning. *Proceedings of The ACM On Interactive, Mobile, Wearable And Ubiquitous Technologies*. **4**, 1-21 (2020)

[48] Yu, T., Li, T., Sun, Y., Nanda, S., Smith, V., Sekar, V. & Seshan, S. Learning Context-aware Policies from Multiple Smart Homes via Federated Multi-task Learning. *IEEE/ACM International Conference On Internet-of-Things Design And Implementation (IoTDI)*. pp. 104-115 (2020)

[49] Huang, L., Shea, A., Qian, H., Masurkar, A., Deng, H. & Liu, D. Patient Clustering Improves Efficiency of Federated Machine Learning to Predict Mortality and Hospital Stay Time using Distributed Electronic Medical Records. *Journal Of Biomedical Informatics*. **99** pp. 103291 (2019)

[50] Brisimi, T., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I. & Shi, W. Federated Learning of Predictive Models from Federated Electronic Health Records. *International Journal Of Medical Informatics*. **112** pp. 59-67 (2018)

[51] Lee, J., Sun, J., Wang, F., Wang, S., Jun, C. & Jiang, X. Privacy-preserving Patient Similarity Learning in a Federated Environment: Development and Analysis. *JMIR Medical Informatics*. **6**, e7744 (2018)

[52] Ahn, E., Kumar, A., Feng, D., Fulham, M. & Kim, J. Unsupervised Deep Transfer Feature Learning for Medical Image Classification. *IEEE International Symposium On Biomedical Imaging (ISBI 2019)*. pp. 1915-1918 (2019)

[53] Lin, B., He, C., Zeng, Z., Wang, H., Huang, Y., Soltanolkotabi, M., Ren, X. & Avestimehr, S. FedNLP: Benchmarking Federated Learning Methods for Natural Language Processing Tasks. *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. (2022)

[54] Iordanou, C., Soriente, C., Sirivianos, M. & Laoutaris, N. Who is Fiddling with Prices? Building and Deploying a Watchdog Service for e-commerce. *Proceedings of ACM SIGCOMM* (2017)

[55] Iordanou, C., Kourtellis, N., Carrascosa, J., Soriente, C., Cuevas, R. & Laoutaris, N. Beyond Content Analysis: Detecting Targeted Ads via Distributed Counting. *Proceedings Of The 15th International Conference On Emerging Networking Experiments And Technologies*. pp. 110-122 (2019)

[56] Su, D., Cao, J., Li, N., Bertino, E. & Jin, H. Differentially Private k-means Clustering. *Proceedings Of The Sixth ACM Conference On Data And Application Security And Privacy*. pp. 26-37 (2016)

[57] Cormode, G. & Muthukrishnan, S. An Improved Data Stream Summary: the Count-Min Sketch and its Applications. *Journal Of Algorithms*. **55**, 58-75 (2005)

[58] Salton, G. & Buckley, C. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*. **24**, 513-523 (1988)

[59] Lopuhaa, H. & Rousseeuw, P. Breakdown points of Affine Equivariant Estimators of Multivariate Location and Covariance Matrices. *The Annals Of Statistics*. pp. 229-248 (1991)

[60] Cao, X., Fang, M., Liu, J. & Gong, N. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. *Proceedings Of NDSS*. (2021)

[61] Fleiss, J. Measuring Nominal Scale Agreement among Many Raters. *Psychological Bulletin*. **76**, 378 (1971)

[62] Yuan, Z., Guo, Z., Xu, Y., Ying, Y. & Yang, T. Federated Deep AUC Maximization for Hetergeneous Data with a Constant Communication Complexity. *International Conference On Machine Learning*. pp. 12219-12229 (2021)

[63] Shejwalkar, V. & Houmansadr, A. Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning. *NDSS*. (2021)

[64] Melis, L., Song, C., De Cristofaro, E. & Shmatikov, V. Exploiting Unintended Feature Leakage in Collaborative Learning. *2019 IEEE Symposium On Security And Privacy (SP)*. pp. 691-706 (2019)

[65] Naseri, M., Hayes, J. & De Cristofaro, E. Local and Central Differential Privacy forRobustness and Privacy in Federated Learning. *In Proceedings of NDSS*. (2022)

[66] Rieger, P., Nguyen, T., Miettinen, M. & Sadeghi, A. DeepSight: Mitigating Backdoor Attacks in Federated Learning Through Deep Model Inspection. *In Proceedings of NDSS*. (2022)

[67] EITR System, https://eitr-experiment.networks.imdea.org (2022)

APPENDIX

A. Proofs

The following are the lemmas we utilise in the proof of Theorem 1.

Lemma 1. *From Assumption 1 and 4, $\mathcal{L}(w)$ is L-smooth and μ-strongly convex. Then $\forall w_1, w_2 \in \mathcal{W}$, one has*

$$\langle \nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2\rangle \geq \frac{L\mu}{L+\mu}\left\|\mathbf{w}_1 - \mathbf{w}_2\right\|_2^2 + \frac{1}{L+\mu}\left\|\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2)\right\|_2^2 \quad (15)$$

Lemma 2. *The difference between $\mathbf{m}(\mathbf{w})$ and $\nabla\mathcal{L}(\mathbf{w})$ is bounded in every iteration:*

$$\left\|\mathbf{m}(\mathbf{w}) - \nabla\mathcal{L}(\mathbf{w})\right\|_2 \leq \left\|\mathbf{m}_0(\mathbf{w}) - \nabla\mathcal{L}(\mathbf{w})\right\|_2 + \sqrt{N}\Delta_1 \quad (16)$$

where:

$$\Delta_1 = \frac{M(\varpi(M-1) + \frac{2E}{\sqrt{M}\delta})}{\frac{Wa(M-1)(\kappa N+W)}{(\eta N+W)(\kappa N+Wa)} + 1}$$

$$E = \sup\left\{\frac{37\sqrt{2}\lambda(M+4)}{25(M-1)}\underset{i}{\text{median}}\left\{|w_{i,n}^t - \hat{B}_n x_{i,n}^t - \hat{A}_n|\right\}\right\}$$

and

$$\mathbf{m}_0(\mathbf{w}) := \underset{i}{\text{median}}\left\{\mathbf{m}_i(\mathbf{w})\right\}$$

1) Proof of Lemma 1: Let $g(\mathbf{w}) = \mathcal{L}(\mathbf{w}) - \frac{\varsigma}{2}\left\|\mathbf{w}\right\|_2^2$. Base on the assumption 4, we have $g(\mathbf{w})$ is $(L-\varsigma)$-strongly convex. from [23] 3.6, we have

$$\langle \nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2\rangle \geq \frac{1}{L}\left\|\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2)\right\|_2^2 \quad (17)$$

Hence,

$$\langle \nabla g(\mathbf{w}_1) - \nabla g(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2\rangle \geq \frac{1}{L-\varsigma}\left\|\nabla g(\mathbf{w}_1) - \nabla g(\mathbf{w}_2)\right\|_2^2 \quad (18)$$

Now We have

$$\begin{aligned}&\langle \nabla\left(\mathcal{L}(\mathbf{w}_1) - \tfrac{\varsigma}{2}\left\|\mathbf{w}_1\right\|_2^2\right) - \nabla\left(\mathcal{L}(\mathbf{w}_2) - \tfrac{\varsigma}{2}\left\|\mathbf{w}_2\right\|_2^2\right), \mathbf{w}_1 - \mathbf{w}_2\rangle \\ &\geq \frac{1}{L+\mu}\left\|\nabla\left(\mathcal{L}(\mathbf{w_1}) - \tfrac{\varsigma}{2}\left\|\mathbf{w}_1\right\|_2^2\right) - \nabla\left(\mathcal{L}(\mathbf{w}_2) - \tfrac{\varsigma}{2}\left\|\mathbf{w}_2\right\|_2^2\right)\right\|_2^2\end{aligned} \quad (19)$$

And therefore

$$\begin{aligned}&\langle \nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2\rangle - \langle \varsigma\mathbf{w}_1 - \varsigma\mathbf{w}_2, \mathbf{w}_1 - \mathbf{w}_2\rangle \\ &\geq \frac{1}{L-\varsigma}\left\|(\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2)) - (\varsigma\mathbf{w}_1 - \varsigma\mathbf{w}_2)\right\|_2^2\end{aligned} \quad (20)$$

Refer to Assumption 1, we obtain

$$\begin{aligned}&\langle \nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2\rangle \geq \frac{L\varsigma}{L-\varsigma}\left\|\mathbf{w}_1 - \mathbf{w}_2\right\|_2^2 \\ &- \frac{2\varsigma}{L-\varsigma}\langle \nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2\rangle \\ &+ \frac{1}{L-\varsigma}\left\|\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2)\right\|_2^2 \\ &\geq -\frac{L\varsigma}{L-\varsigma}\left\|\mathbf{w}_1 - \mathbf{w}_2\right\|_2^2 + \frac{1}{L-\varsigma}\left\|\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2)\right\|_2^2\end{aligned} \quad (21)$$

Let $\varsigma = -\mu$, then we conclude the proof of Lemma 1.

2) Proof of Lemma 2: We have the following equation:

$$\begin{aligned}\left\|\mathbf{m}(\mathbf{w}) - \nabla\mathcal{L}(\mathbf{w})\right\|_2 \leq &\left\|\mathbf{m}(\mathbf{w}) - \mathbf{m}_0(\mathbf{w})\right\|_2 \\ &+ \left\|\mathbf{m}_0(\mathbf{w}) - \nabla\mathcal{L}(\mathbf{w})\right\|_2\end{aligned} \quad (22)$$

from [16] inequality 18, we know $\forall i, n, \exists E > 0$

$$\sup|e_{i,n}| \leq \frac{E}{\sqrt{M}\delta}$$

Where

$$E = \sup\left\{\frac{37\sqrt{2}\lambda(M+4)}{25(M-1)}\underset{i}{\text{median}}\left\{|w_{i,n}^t - \hat{B}_n x_{i,n}^t - \hat{A}_n|\right\}\right\}$$

and the dimension of $\mathbf{w}$ is N. Hence the distance between the two aggregation functions satisfies

$$\left\|\mathbf{m}(\mathbf{w}) - \mathbf{m}_0(\mathbf{w})\right\|_2 \leq \sqrt{N}\left\|\sum_{i=1}^{M}\bar{R}_i\left(\hat{B}_i\left(M-1\right) + \frac{2E}{\sqrt{M}\delta}\right)\right\|_2 \quad (23)$$

Based on Equation 12

$$\frac{s\exp(-cs)}{\sum_{j=0}^{s}\exp(-cj)}\cdot\frac{Wa}{\eta N+W} \leq \tilde{R}_i^t \quad (24)$$

$$\tilde{R}_i^t \leq \frac{s}{\sum_{j=0}^{s}\exp(-cj)}\cdot\frac{\kappa N+Wa}{\kappa N+W} \quad (25)$$

so we have

$$\bar{R}_i = \frac{\tilde{R}_i^t}{\sum_{i=1}^{M}\tilde{R}_i^t} \leq \frac{1}{\frac{Wa(M-1)(\kappa N+W)}{(\eta N+W)(\kappa N+Wa)} + 1} \quad (26)$$

Due to our Aggregation Algorithm

$$\hat{B}_n = \underset{i}{\text{median}}\left(\underset{i\neq j}{\text{median}}\frac{w_{j,n} - w_{i,n}}{x_{j,n} - x_{i,n}}\right) \leq \varpi \quad (27)$$

Therefore, we have

$$\begin{aligned}\left\|\sum_{i=1}^{M}\bar{R}_i\left(\hat{B}_i\left(M-1\right) + \frac{2E}{\sqrt{M}\delta}\right)\right\|_2 \leq &\frac{M(\varpi(M-1) + \frac{2E}{\sqrt{M}\delta})}{\frac{Wa(M-1)(\kappa N+W)}{(\eta N+W)(\kappa N+Wa)} + 1} \\ &= \Delta_1\end{aligned} \quad (28)$$

Hence, we proof Lemma 2.

3) Proof of Theorem 1: We first consider a general problem of robust estimation of a one dimensional random variable. Suppose that there are M clients, and p percentage of them are malicious and own adversarial data. In t iteration, we have:

$$\begin{aligned}&\left\|\mathbf{w}^t - \mathbf{w}^*\right\|_2 = \left\|(\mathbf{w}^{t-1} - r\mathbf{m}(\mathbf{w}^{t-1}) - \mathbf{w}^*\right\|_2 \\ &\leq \underbrace{\left\|\mathbf{w}^{t-1} - r\nabla\mathcal{L}(\mathbf{w}^{t-1}) - \mathbf{w}^*\right\|_2}_{A} \\ &+ \underbrace{r\left\|\mathbf{m}(\mathbf{w}^{t-1}) - \nabla\mathcal{L}(\mathbf{w}^{t-1})\right\|_2}_{B}\end{aligned} \quad (29)$$

We bound part A first. We have

$$\begin{aligned}&\left\|\mathbf{w}^{t-1} - r\nabla\mathcal{L}(\mathbf{w}^{t-1}) - \mathbf{w}^*\right\|_2^2 = \left\|\mathbf{w}^{t-1} - \mathbf{w}^*\right\|_2^2 \\ &+ r^2\left\|\nabla\mathcal{L}(\mathbf{w}^{t-1})\right\|_2^2 - 2r\left\langle\nabla\mathcal{L}(\mathbf{w}^{t-1}), \mathbf{w}^{t-1} - \mathbf{w}^*\right\rangle\end{aligned} \quad (30)$$

Under the Assumption 4 and Lemma 1, we have

$$\left\langle \nabla \mathcal{L}(\mathbf{w}^{t-1}), \mathbf{w}^{t-1} - \mathbf{w}^* \right\rangle \geq \frac{L\mu}{L+\mu} \left\| \mathbf{w}^{t-1} - \mathbf{w}^* \right\|_2^2 + \frac{1}{L+\mu} \left\| \nabla \mathcal{L}(\mathbf{w}^{t-1}) \right\|_2^2 \quad (31)$$

Then we combine inequalities 31 to equation 30

$$\left\| \mathbf{w}^{t-1} - r\nabla \mathcal{L}(\mathbf{w}^{t-1}) - \mathbf{w}^* \right\|_2^2 \leq (1 - 2r\frac{L\mu}{L+\mu}) \left\| \mathbf{w}^{t-1} - \mathbf{w}^* \right\|_2^2 + (r^2 - \frac{2r}{L+\mu}) \left\| \nabla \mathcal{L}(\mathbf{w}^{t-1}) \right\|_2^2 \quad (32)$$

From Assumption 1, we can derive:

$$\left\| \nabla \mathcal{L}(\mathbf{w}^{t-1}) - \nabla \mathcal{L}(\mathbf{w}^*) \right\|_2^2 \leq L^2 \left\| \mathbf{w}^{t-1} - \mathbf{w}^* \right\|_2^2 \quad (33)$$

Combining inequalities 32 and 33, we have:

$$\left\| \mathbf{w}^{t-1} - r\nabla \mathcal{L}(\mathbf{w}^{t-1}) - \mathbf{w}^* \right\|_2^2 \leq (1 - Lr)^2 \left\| \mathbf{w}^{t-1} - \mathbf{w}^* \right\|_2^2 \quad (34)$$

Let $r < \frac{1}{L}$, we have

$$\left\| \mathbf{w}^{t-1} - r\nabla \mathcal{L}(\mathbf{w}^{t-1}) - \mathbf{w}^* \right\|_2 \leq (1 - Lr) \left\| \mathbf{w}^{t-1} - \mathbf{w}^* \right\|_2 \quad (35)$$

Then we turn to bound part B. Based on Lemma 2, we know:

$$\left\| \mathbf{m}(\mathbf{w}) - \nabla \mathcal{L}(\mathbf{w}) \right\|_2 \leq \left\| \mathbf{m}_0(\mathbf{w}) - \nabla \mathcal{L}(\mathbf{w}) \right\|_2 + \sqrt{N}\Delta_1 \quad (36)$$

Assume Assumption 1, 2, 3 and 4 holds, and $\exists \epsilon$ fulfills inequality 13. Based on Lemma 1 in [20], with the probability $1 - \xi \geq 1 - \frac{4d}{\left(1+\hat{Q}ML\upsilon\right)^a}$, we have

$$\left\| \mathbf{m}_0(\mathbf{w}) - \nabla \mathcal{L}(\mathbf{w}) \right\|_2 \leq \sqrt{\frac{2}{\hat{Q}}} D_\epsilon V_w \left(\sqrt{\frac{d \log(1 + \hat{Q}ML\upsilon)}{M(1-p)}} + C\frac{\mathcal{G}_w}{\sqrt{\hat{Q}}} + p \right) + 2\sqrt{2}\frac{1}{M\hat{Q}} = \Delta_2 \quad (37)$$

where $C = 0.4748$. After obtaining the bound of part A and B, now we have

$$\left\| \mathbf{w}^t - \mathbf{w}^* \right\|_2 \leq \underbrace{(1 - Lr) \left\| \mathbf{w}^{t-1} - \mathbf{w}^* \right\|_2}_{Bound\ A} + \underbrace{r\left(\sqrt{N}\Delta_1 + \Delta_2\right)}_{Bound\ B} \quad (38)$$

Hence, we can prove Theorem 1 through iterations using the formula of a finite geometric series,

$$\begin{aligned} \left\| \mathbf{w}^t - \mathbf{w}^* \right\|_2 &\leq (1 - Lr)^t \left\| \mathbf{w}^0 - \mathbf{w}^* \right\|_2 + \frac{1 - (1 - Lr)^t}{Lr} r\left(\sqrt{N}\Delta_1 + \Delta_2\right) \\ &\leq (1 - Lr)^t \left\| \mathbf{w}^0 - \mathbf{w}^* \right\|_2 + \frac{1}{L}\left(\sqrt{N}\Delta_1 + \Delta_2\right) \end{aligned} \quad (39)$$

B. Experimental Setting

Our simulation experiments are implemented with Pytorch framework [28] on the cloud computing platform Google Colaboratory Pro (Colab Pro) with access to Nvidia K80s, T4s, P4s and P100s with 25 GB of Random Access Memory. Table II shows the default setting in our experiments.

TABLE II: Default experimental settings

Explanation	Notation	Default Setting
prior probability	a	0.5
non-information prior weight	W	2
weight for positive observation	κ	0.3
time decay parameter	c	0.5
window length	s	10
confidence threshold	δ	0.1
value range	ϖ	2
Objective Function	$\mathcal{L}(\cdot)$	Negative Log-likelihood Loss
Learning rate	r	0.01
Batch size per client		64
The number of local iterations		10
The number of total iterations		100

C. Supplementary dataset for experiment

CIFAR-10 is a supplementary dataset assessing the robustness of our reputation scheme in image datasets to poisoning attacks. The CIFAR-10 dataset is a 32×32 colour image dataset that includes ten classes with a total number of 50 thousand images for training and 10 thousand images for testing. Here we use ResNet-18 [19] model pertaining to ImageNet [25] with 20 iterations. In CIFAR-10 dataset, attackers switch the label of "cat" images to the "dog".

Figure 15 shows that under label flipping and backdoor attack during the whole process, our reputation-based method has the highest accuracy outperforming other methods and the lowest ASR, excluding Foolsgold in CIFAR-10, yielding a result that is similar to the one in SURL.

D. Extra Experimental Result

We evaluate the proposed defence for a varying number of clients from 10 to 200 in the SURL dataset. Here, we analyze the performance of the aforementioned methods for 100 clients. The results are presented in Figure 16 and Figure 17 that correspond to the average accuracy (ACC) and attach success rate (ASR), respectively.

In the no attack scenario, we observe that our method converges between $1.7\times$ to $7.1\times$ faster than all competing state-of-the-art methods with at least as good performance (or outperforms) compared with competing methods in terms of classification accuracy, see Figure 16 (left). In addition, even under the two different attacks, our method: (i) converges between $1.6\times$ to $3.6\times$ faster than all competing state-of-the-art methods, (ii) provides the same or better accuracy than competing methods, and (iii) yields the lowest ASR compared to all other methods.

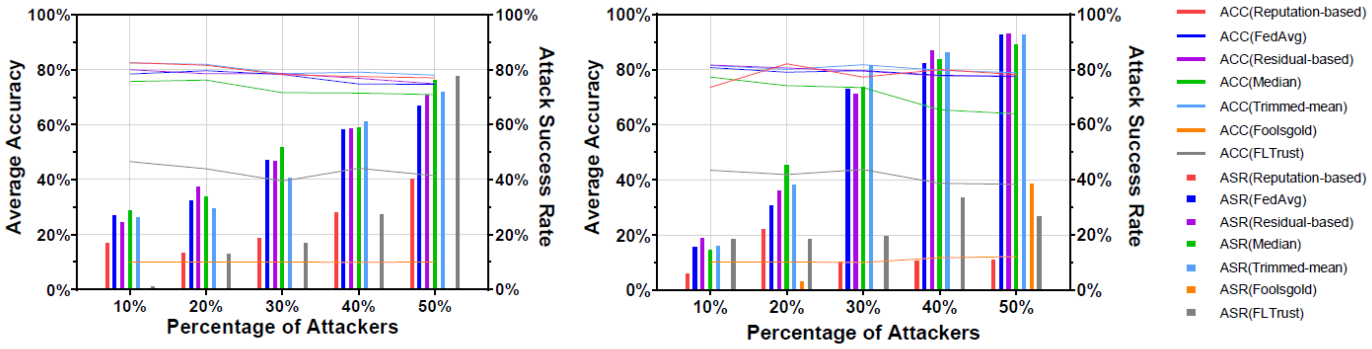


Fig. 15: Average accuracy (ACC) and attack success rate (ASR) for varying percentage of attackers from 10% to 50% under label flipping (left) and backdoor (right) attack for Reputation, FedAvg, Median, Residual-based, Median, Trimmed-Median, FoolsGold and FLTruts in CIFAR-10 dataset.

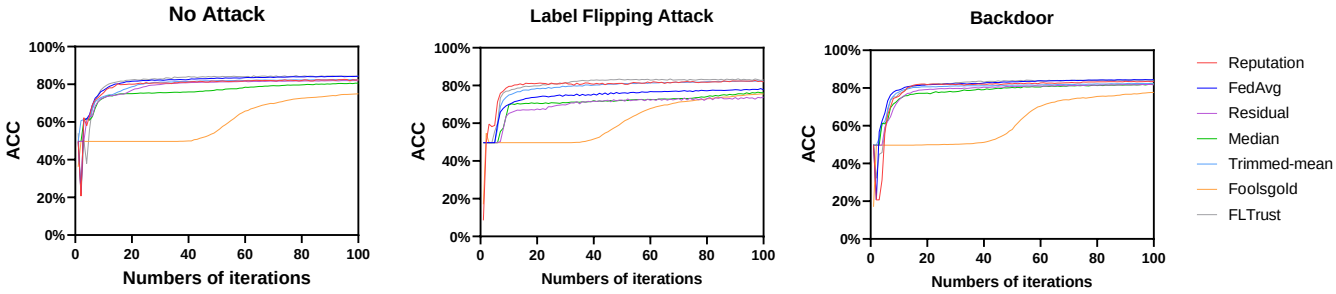


Fig. 16: Average accuracy (ACC) with no attack (left) and varying percentage of attackers from 10% to 50% under label flipping (middle) and backdoor (right) attack for Reputation, FedAvg, Residual-based, Median, Trimmed-mean, Foolsgold, FLTrust in SURL dataset.

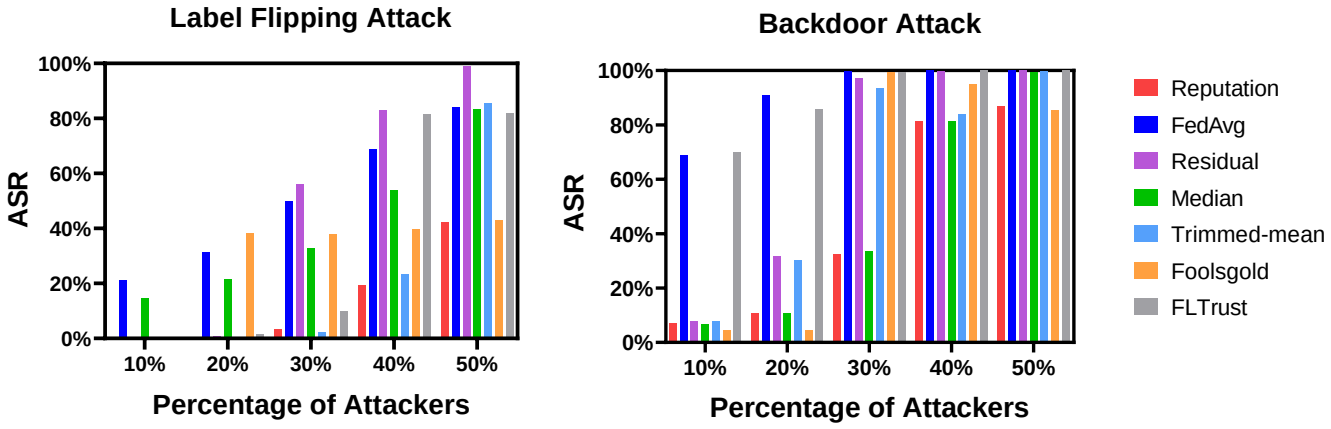


Fig. 17: Attack success rate (ASR) for varying percentage of attackers from 10% to 50% under label flipping (left) and backdoor (right) attack for Reputation, FedAvg, Residual-based, Median, Trimmed-mean, Foolsgold, FLTrust in SURL dataset.

Anexo 3: FreqyWM - Frequency Watermarking for the New Data Economy

FreqyWM: Frequency Watermarking for the New Data Economy

Devriş İşler[1,2], Elisa Cabana[3§], Alvaro Garcia-Recuero[4§], Georgia Koutrika[5], and Nikolaos Laoutaris[1]
[1]*IMDEA Networks Institute,* [2]*UC3M,* [3]*CUNEF University,* [4]*FUNDITEC,* [5]*Athena Research Center*

***Abstract*—We present a novel technique for modulating the appearance frequency of a few tokens within a dataset for encoding an invisible watermark that can be used to protect ownership rights upon data. We develop optimal as well as fast heuristic algorithms for creating and verifying such watermarks. We also demonstrate the robustness of our technique against various attacks and derive analytical bounds for the false positive probability of erroneously "detecting" a watermark on a dataset that does not carry it. Our technique is applicable to both single dimensional and multidimensional datasets, is independent of token type, allows for a fine control of the introduced distortion, and can be used in a variety of use cases that involve buying and selling data in contemporary data marketplaces.**

***Index Terms*—Intellectual property, digital rights management, watermarking, ownership rights, data economy**

I. Introduction

Data-driven decision making powered by Machine Learning (ML) algorithms is changing how society and the economy work. ML is driving up the demand for data in what has been called the fourth industrial revolution. To satisfy this demand, several data marketplaces (DMs),which are mediation platforms aiming to connect the two primary stakeholders of the data value chain, namely the data providers/sellers and the data buyers [1], have appeared in the last few years.

The problems: Unfortunately, as with all digital assets, being able to copy/store/transmit datasets with close to zero cost makes creating illegal copies very easy. Even worse, unlike media content and software, the issue of ownership is less obvious when it comes to datasets. Any movie, song, e-book, or software can usually be attributed to a director, musician, author, or company, respectively, but this is hard to do for large datasets. These large datasets in data economy are traded in a wholesale manner that involves large numbers of tuples/rows. Consider an anonymised mobility dataset logging the movement of people in a city. Such a dataset may have been produced by collecting GPS readings from the smartphones of individuals using a map application, or it may be deduced by analysing cell phone traces [2] or Call Description Records (CDRs) maintained by mobile operators. Deployment of advanced privacy enhancing technologies (PETs) such as multiparty computation [3], (fully) homomorphic encryption [4], functional encryption [5], and trusted execution environments [6] can protect data from leaking in the first place and allow (pre-agreed) computations on data without hampering the functioning of the data-driven economy, e.g., private set computation [7], encrypted databases [8], secure computation [9], secure data aggregation [10], and verifiable databases [11]. However, most such approaches face serious scalability challenges that hamper their deployment in real-world use-cases. An alternative to deploying PETs solutions, is to rely on purely legal tools and terms and conditions to protect data ownership in the context of the new data economy [12]. In fact, most DMs do exactly that – trade plaintext versions of entire data [1, 13, 14] assuming that the different parties will abide to pre-agreed terms and conditions. With weak to nonexistent ownership guarantees by technical means, it is difficult to imagine that the data economy will ever flourish and reach its projected potential [15]. Indeed, any sold copy of a dataset can be 'pirated' by a buyer-turned-seller that can then resell the same dataset in a DM thereby undercutting the rightful owner and rendering its investment useless.

Watermarking is a well-known technique for protecting ownership upon copying and unauthorized distribution, initially proposed for protecting digital media [16, 17] and software [18]. Watermarking techniques for datasets [19, 20, 21] and machine learning models [22] have been proposed recently. Watermarking generally consists of two algorithms: *generation* (or embedding) and *detection*. The generation allows an owner to embed an invisible (or visible) watermark into their data using a high entropy (watermarking) secret and produces a watermarked version of the data introducing tolerable distortion without degrading the data utility. During the detection algorithm, the owner proves its ownership on the suspected data (even if it is modified) using the same watermarking secret generated during the watermark generation. If the result of the detection is 1 (or *accept*), the owner can use it to prove their ownership on the (suspected) watermarked data. A watermarking scheme is assumed to be secure against the guess attack (where an attacker tries to expose the watermarking secret) and robust against (un)intentional alterations/-modifications (i.e., a watermark should be still detectable even under attacks such as [20, 23, 24, 25, 26, 27, 28]).

Limitations of existing watermarking techniques: Watermarking techniques, depending on the nature of their application, may have very different objectives, e.g., numerical database watermarking controlling the distortion on mean and standard deviation [21], reversible watermarking allowing owners to reconstruct the original data [29], watermarking text datasets preserving the meaning of a text [30] and/or the frequencies of the words [31], categorical watermarking preserving the (predefined) categories (e.g., gender) of a

§Work done while the author was affiliated with IMDEA Networks Institute.

dataset [32]. All these solutions focus on a specific data type in a specific domain [23, 33]. Another limitation of theirs relates to the level of control they offer to the user in terms of controlling the distortion introduced upon the original data due to the watermark. There are, for example, techniques that maintain the mean and the standard deviation of a numerical field [20, 34, 35] but, as we will show later, this can lead to arbitrary large distortion between the original and the watermarked data when considering the entire distribution of values that goes beyond the mean and the standard deviation. To address these limitations, *we introduce a novel watermarking technique that can be implemented over a wide range of data types and structures* (with some constraints that will be explained later) while *giving the data owner very precise control over the introduced distortion*.

A novel watermarking technique for data: In this paper, we present a novel *Frequency Watermarking* technique, henceforth *FreqyWM*, [1] for hiding a secret within a dataset in a manner that makes the said secret indistinguishable from the data that it protects. The main idea behind *FreqyWM* is *to modify slightly the appearance frequency of existing tokens* within a dataset in order *to create a secret in the form of a complex relationship* between the frequencies of different tokens. By making this relationship complex enough, we can reduce the probability that it appeared by chance close to zero. Therefore, by revealing knowledge of such secret relationship, a party can claim ownership over a dataset because the only practical way of knowing such a secret is to have inserted it in the data in the first place. A token may be a word, a database record, a URL, or any repeating value within a structured or semi-structured commercial dataset. Our secret is created by first selecting a number of token pairs. Then, for each pair, we slightly modulate the frequency counts of its tokens in order to make their difference yield zero under modulo N arithmetic. This can be easily done by adding or removing some instances of one, the other, or both tokens. By increasing the number of selected pairs we can make our watermark more resistant to attacks, as well as less likely to have appeared by chance.

FreqyWM can achieve several things. First and foremost, by revealing knowledge of the secret encoded by the watermark, a data seller can prove rightful ownership of a dataset to a third party, such as a DM. This can be used to distinguish a rightful owner from a pirate that may attempt to monetize a pirated dataset in a DM. If the DM, or the rightful owner detects such an event, the dataset can be removed and the pirate be banned. This would mimic what web-sites like YouTube do to protect copyrighted content. Detecting the presence of pirated copies can be achieved using content similarity [36], locality sensitive hashing [37, 38] and even hashing similarity [39] that go beyond the scope of watermarking.

In addition to proving ownership, our watermarking technique can also reveal who may have leaked (copied/pirated) a dataset in the first place. A dataset seller or a DM may create a different watermark for every buyer and in addition to encoding it into the data, store also a description of it in some immutable index (e.g., a blockchain). Then, if an unauthorized copy of the dataset is found at a latter point, the culprit can be identified by looking up its watermark against this index.

[1]Freqy pronounced as *freaky*.

Our major contributions are as follows:

• Our first contribution is the idea of using the appearance frequency of tokens to encode invisible watermarks upon datasets traded in DMs. We establish a family of such watermarks using frequency pairs and modulo arithmetic and prove that creating an optimal *FreqyWM* reduces to solving a Maximum Weighted Matching (MWM) problem [40, 41] combined with a polynomial special version of the 0/1 Knapsack problem [42] involving items of equal value but different weights.

• We extend frequency-based watermarking to make it resilient against a series of attacks. In particular, we protect our technique against a *Guess Attack* attempting to identify our watermarked pairs and secrets to impersonate the rightful owner. We make such an attack computationally hard by introducing a high-entropy secret while generating the watermark. We also protect against a *Re-watermarking Attack* mounted by having a pirate inject its own watermark upon an already watermarked dataset, and then present the former as a false proof of ownership. We thwart such an attack by describing a simple protocol capable of ordering chronologically multiple watermarks that may be carried by different versions of the same dataset. We protect against a *Destroy Attack* attempting to destroy our watermark by changing the frequency of different tokens in the dataset. By relaxing our modulo arithmetic rule used during the verification of a particular watermark pair, as well as the percentage of pairs to be detected before the entire watermark is verified (accepted), we oblige the attacker to effectively also destroy the actual data in the process of destroying the watermark. Finally, we show that our technique is robust to a *Sampling Attack* in which the attacker attempts to pirate only a random sample of the watermarked data.

• Our final contribution is an extensive performance evaluation study aiming to explain the impact of the main parameters of *FreqyWM* on major performance metrics under different attack scenarios using synthetic and real world datasets.

The main **findings** of our evaluation are as follows:

• We show that as long as there exists sufficient variation in the frequencies of different tokens, *FreqyWM* can encode robust watermarks with minimal distortion on the initial data. Our technique does not apply to uniform token appearance frequencies, because in this case there does not exist sufficient gap between different frequencies for encoding a watermark.

• Regarding the false positive probability, i.e., "detecting" a watermark on a dataset that does not carry it, our analytical bounds (in the form of closed form expressions) show that it quickly goes to zero as we increase the number of pairs.

• We demonstrate that a Guess Attack has negligible probability of success, thereby making it impossible for almost all practical cases. On the up side, the rightful owner or any party, that is given the watermarking secret for verifying the watermark, can do that very fast in linear time complexity.

• Regarding Sampling Attacks, we show that with the exception of very small samples, our detection algorithm is capable of detecting our watermark. Achieving this requires using the relaxed detection algorithm that trades robustness to attacks with false positives. For example, on a sample of 20% and with thresholds that impose tiny false positive probability, the detection probability exceeds 90%.
• In terms of Destroy Attacks, we show that a watermark that imposes (costs) a tiny 0.0002% distortion on the original data, remains detectable even under attacks that add random noise that imposes a 90% modification.
• Compared to existing solutions from the literature [30, 35] that are applicable only to numerical data and preserve only the mean value of the watermarked data, *FreqyWM* allows a data owner to control the exact amount of distortion introduced by the watermark in terms of cosine or other similarity metrics which, under [30, 35] may become unbounded. For example, a *FreqyWM* watermark that imposes only 0.0002% distortion in terms of cosine similarity, is stronger than watermarks from [35] and [30] that impose 46.72% and 4% distortion, respectively under the same metric.

II. Related Work

Database watermarking is the closest type of watermarking to our work. There are of course other types of watermarking and fingerprinting (when an owner generates a unique watermark for each intended party, e.g., buyers/data marketplaces), for example, for sequential [43] and genomic datasets [44]. However, as they focus on specialized types of data, we do not go into more details about them. Survey papers such as [23, 33, 45, 46] compare database watermarking techniques in terms of verifiability, distortion, supported data types, and other aspects. Many of these solutions are applicable only to numerical data and thus cannot be applied to a range of commercial datasets, e.g., to web-browsing click-streams.

The first known watermarking technique for relational data is a *numerical database watermarking* approach [20]. The watermark information is normally embedded in the Least Significant Bit (LSB) of features of relational databases to minimize distortion. Other numerical database watermarking solutions introduce distortion by considering the statistics of numeric values [34, 35, 47, 48, 49]. The proposed solutions in [20, 35] focus on keeping the change at minimum (i.e., median and standard deviation). *However*, numerical database watermarking unfortunately cannot be applied to datasets composed of string and numerical values (e.g., CDRs, web-browsing history) that we handle in our work.

Distortion-free database watermarking schemes have also been proposed [50, 51] that introduce fake tuples or columns in the original database. The fake tuples or columns are created based on a watermark secret by computing a secret function which makes watermarking visible and easy to remove. *However, an attacker can remove the watermark with minimum computational power*, making these approaches inapplicable to our case. *Reversible watermarking* allows owners to reconstruct the original data used for watermarking on the top of watermark verification [29, 30, 49, 52, 53, 54, 55, 56, 57]. They have similar properties as other relational watermarking techniques (e.g., private key based, robust, introducing distortion).

Categorical watermarking [32] is another watermarking approach that replaces tokens in a dataset with another token in the same category. However, this causes an undesired distortion and requires predefined categories (e.g., gender, clothing size) in the data. Consequently, its applicability on datasets consisting of different data types is limited. *Text watermarking* [30, 31] is for text files where it changes a token (e.g., by replacing a word with another similar word) trying to preserve the meaning of a text [30] and/or the frequencies of the words [31]. However, assume the dataset is a list of URLs visited by the owner, then this (insecure) change/replacement may invalidate a token (e.g., causing an invalid URL).

In the context of datasets in our use case, while prior works try to minimize the amount of distortion on median, average, or first moments of the distribution of a feature, the owner can limit the exact distortion between the original and the watermarked dataset as reflected by distance metrics that capture the shape of the entire distribution of a feature. Our results in Section IV-D have shown that the latter can deviate arbitrarily if an owner tries to control only the first most important moments.

III. Frequency-based Watermarking

In this section we provide an overview of *FreqyWM* and the notations used throughout the paper in Table I.

D_o	The original data to watermark.
D_w	Watermarked (data) version of D_o.
tk_i	ith token.
f_i^o	Frequency of ith token in D_o.
f_i^w	Frequency of ith token in D_w.
R	A high entropy secret.
L_{wm}	A list of chosen token pairs for watermarking.
L_{sc}	A list of secrets required for watermark detection.
L_e	A list of eligible token pairs for watermarking.
k	Threshold for detecting a watermark.
t	Threshold to accept a pair as watermarked.
b	A budget threshold for distortion that watermarking can introduce.

TABLE I: Notation.

Running Example. To provide the intuition behind our watermarking approach, assume a scenario where an owner holds a real click-stream dataset consisting of visited URLs (e.g., the dataset by [58]). Such datasets are desired by modern data analytic-based applications [59] where their frequency histograms (e.g., the number of clicks/visits, popularity of likes in social networks) are used as an essential source of information. For instance, assuming the appearance frequencies (histogram) visualized in Figure 1 via a tabular form, the most frequent token is `youtube.com`, the second one is `facebook.com`, and so forth. After watermarking, it is important that the *ranking* of the tokens based on the frequency shall not change while the frequency appearances can be modified. For instance `youtube.com` shall be the most frequent URL (token) visited in the watermarked dataset. Another important distortion metric on the histogram is *similarity*. It

is important that owners shall have control over the change in similarity. Since the similarity metric can be varied depending on the application that a dataset will be used, owners can assign a *budget* to determine the minimum similarity desired on the frequency distribution after watermarking. Based on the above, we derive two natural constraints on the data utility to allow an owner to control distortion, without limiting watermarking to a specific data type:

• *Ranking Constraint:* Watermarking should preserve the ranking of token frequency distribution (histogram). Preservation of ranking does not of course imply that frequencies of individual tokens need to remain intact. • *Similarity Constraint:* The similarity between original and watermarked frequency distributions (histograms) should not be any less than $(100 - b)\%$ where b is a budget. Input b is determined by the owner to keep distortion due to watermark generation within b. [2]

To satisfy these constraints and overcome the shortcoming of existing watermarking techniques, we introduce a new *private-key based* watermarking scheme, *FreqyWM*, that is *blind* (does not require the original data), *primary-key free* (does not need attributes that uniquely specify a tuple in a relation in a dataset), *robust*, and *secure* against *guess*, *sampling*, *destroy*, and *false-claim* attacks with a high utility and a good trade-off between the complexity of the transformation and algorithmic efficiency of the solution.

A. Overview of our Approach

FreqyWM consists of two main algorithms: the watermark generation algorithm, $WMGenerate$, and the watermark detection algorithm, $WMDetect$. $WMGenerate$ generates watermarked data based on a *budget* b capturing how much the watermarked data may differ from the original one, e.g., in terms of cosine (or other) similarity metrics of their corresponding token frequency distributions. By calling $WMGenerate$, the owner creates a watermarked version of their data consisting of tokens such that ownership can be proved. $WMDetect$ detects if a suspected dataset holds the watermark of the owner using the owner secrets produced by $WMGenerate$ and two thresholds (k and t). If $WMDetect$ outputs *accept/verified*, this evidence would prove that the owner can claim ownership of the watermark and thus the data. By nature, $WMDetect$ can be computed as many times as desired in private while it can be computed *only once* in public, because it would mean that the potential data owner shall reveal the secret leading to such watermark to the public (or whomever must verify it, e.g., a judging third party). As part of our future work, we are also looking at public verifiability without revealing the private key (Section VII).

We describe the general idea behind *FreqyWM*, illustrated in Figure 1. We use our running example. Of course, our technique is general and can be applied to any repeating token beyond just URLs, as we explain in Section IV-C.

Watermark Generation. Assume that the data owner holding a list of URLs visited creates a dataset D_o using the *domain* of each URL in the list as a token and sets a budget b for the similarity constraint. $WMGenerate$ has the following steps:

• *Histogram Generation.* Since *FreqyWM* aims to preserve the appearance frequency of tokens, it first creates a histogram of the original dataset D_o such that it sorts all unique tokens in descending order of their frequency (e.g., YouTube is the most visited, Facebook is the second, and so on).

• *Generation of Eligible Tokens. FreqyWM* cannot modify the frequencies randomly because of the ranking constraint. Therefore, it identifies a list L_e of eligible pairs of tokens that are candidates to be watermarked using some secret R.

• *Optimal Selection*. With the identification of eligible pairs, *FreqyWM* ensures that the ranking is preserved after watermarking. However, the similarity constraint is yet to be satisfied. To keep the similarity at least at $(100 - b)\%$, *FreqyWM* selects pairs of tokens from eligible pairs for watermarking, denoted by L_{wm}, based on the budget constraint b. For this purpose, *FreqyWM* benefits from solving two well-known problems: *Maximum Weight Matching* (MWM) and *Equally Valued 0/1 Knapsack problem* (QKP). To do so, eligible pairs are converted to a graph representation where vertices represent a token, and an edge represents a pair. *FreqyWM* applies Maximum Weight Matching to the graph representation (discussed in detail later). By applying MWM, *FreqyWM* selects the pairs from eligible pairs requiring the minimum change in total; however, it does not necessarily mean that the similarity between the original histogram and watermarked histogram will be at least $(100-b)\%$. To choose another set of pairs satisfying the Ranking Constraint from the pairs derived after MWM, an *Equally Valued 0/1 Knapsack problem* needs to be solved. The more the token pairs are selected to watermark, the more robust *FreqyWM* is, since the number of tokens to attack (e.g., remove/identify) increases. To fulfill the budget b, QKP selects a maximum number of pairs such that the similarity between the original frequency histogram and the watermarked one is *at least* $(100 - b)\%$.

• *Frequency Modification*. Until now, *FreqyWM* determines the final pairs of tokens for watermarking but frequency appearances are yet to be modified to create the watermarked histogram. Therefore, *FreqyWM* modifies the frequencies of the selected tokens where the frequencies of a pair of tokens would be equal to 0 (as a watermark embedding rule) in some modulo that is calculated based on secrets and tokens in the pair. To make it more comprehensible and show how the modifications occur, let us assume that the frequencies of a chosen pair, e.g., `youtube.com` and `instagram.com`, are 1098 and 537, respectively. Assume also that a modulo value, say 129, is computed based on the secrets and the tokens (e.g., `youtube.com` and `instagram.com`). The difference between the two frequencies in modulo 129 is 45. To set the difference to 0, we need to change the appearance frequencies for Youtube and Instagram in the dataset. 45 is divided (by 2) as 23 (by ceiling) and 22 (by flooring). The new frequencies of `youtube.com` and `instagram.com` need to become $1098 - 23 = 1075$ and $537 + 22 = 559$

[2]Although in our experiments we use cosine similarity, any similarity metrics can be deployed without any loss of security and change in *FreqyWM*.

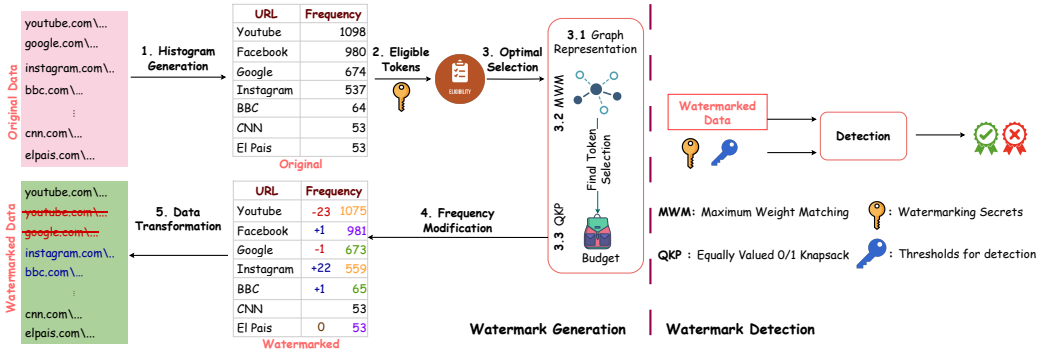


Fig. 1: *FreqyWM* illustrated based on a (Top Level Domain, TLD) URL dataset. URLs chosen as a pair for watermarking are represented with the same colored frequencies (e.g., `Youtube` and `Instagram`) while the ones not selected are colored black (e.g., `CNN`).

such that $(1075 - 559) \mod 129 \equiv 0$. We can do that by removing 23 instances of Youtube from the dataset, and adding 22 more instances of Instagram. However, when the remainder (i.e., $(1098 - 537) \mod 21 \equiv 16$) is greater than half of the modulo, we add the modulo result calculated as $(\lceil (1098 - 537) \div 21 \rceil) \times (1098 - 537)$ to the difference. This way, we never have to eliminate remainders that exceed half of the modulo. As it will be evident in the next section, this observation enables us to determine eligible tokens.

• *Data Transformation. FreqyWM* adds/removes tokens based on the frequencies and produces a watermarked dataset D_w.

Watermark Detection. An owner wishes to verify if a (watermarked) dataset D'_w (a modified version of D_w) is watermarked by using the secrets stored from the watermark generation. To determine the confidence level in the detection (e.g., the minimum number of detected watermarked tokens), the owner provides some threshold values (k and t). With the watermarking secret and the thresholds, the detection returns *accept/verified* or *reject*.

Algorithm I: Watermark Generation

Input: D_o, b
Output: D_w, L_{sc}
$D_o^{hist} = \textbf{Preprocess}(D_o)$
$R \leftarrow \{0,1\}^\lambda$, $z \leftarrow Z^+$
foreach $\{tk_i, tk_j\}_{i \neq j} \in D_o$ **do**
  $s_{ij} = H(tk_i || H(R || tk_j)) \mod z$
end
$L_e \leftarrow \textbf{Eligible}(D_o^{hist}, \{s_{ij}\})$
$L_{wm} \leftarrow \textbf{OptMatch}(D_o^{hist}, L_e, \{s_{ij}\}, b)$
foreach $\{tk_i, tk_j\} \in L_{wm}$ **do**
  $D_o^{hist}.\textbf{Update}(f_i^w, f_j^w, s_{ij})$
end
$D_w \leftarrow \textbf{Create}(D_o^{hist}, D_o)$
$L_{sc} = \{L_{wm}, R, z\}$
Result: $D_w, L_{sc} = \{L_{wm}, R, z\}$

B. Detailed Description of FreqyWM

1) Watermark Generation: The data owner holds the original data D_o and defines a budget b that decides how much distortion a watermark can introduce. For comprehensibility, assume that D_o is a single-dimensional dataset, e.g., a dataset with one attribute (see Section IV-C for how to apply *FreqyWM* to multi-dimensional datasets). D_o consists of repeating values called *tokens* that can be of *any* data type, which enables *FreqyWM* to be data-type agnostic. The goal of watermarking is to generate the optimal watermark, i.e., *with the largest number of watermarked pairs* within the given budget b.

The generation algorithm (Algorithm I) follows these steps:

Histogram Generation: It pre-processes D_o to generate a histogram $D_o^{hist} = \textbf{Preprocess}(D_o)$. D_o^{hist} consists of a set of tokens, $\{tk_0, \ldots, tk_{|D_o^{hist}|}\}$ (e.g., tk_0 =`youtube.com`) where each tk_i has an (original) appearance frequency f_i^o (e.g., there are 1098 YouTube visits). The histogram D_o^{hist} is sorted in a descending order of frequency. To keep the distortion introduced by the watermark at minimum (e.g., after watermarking, YouTube is still the most visited, followed by Facebook, although their frequencies may have changed), we calculate two boundaries for each token tk_i: an upper boundary u_i and a lower boundary l_i. The boundaries allow us to determine how much change we can introduce to the token and whether a token pair is eligible as explained later. Naturally, for the token with the highest frequency in the histogram, it is $u_0 = \infty$ because we can increase the frequency of tk_0 as much as we want, while the lower boundary of the last token, $tk_{|D_o^{hist}|-1}$, is set to its frequency as $l_{|D_o^{hist}|-1} = f_{|D_o^{hist}|-1}$ because we can remove at so many appearances. For the rest of the boundary calculations of each token tk_i, u_i is defined as the difference between $f_{(i-1)}^o$ and f_i^o, while l_i is assigned as $f_i^o - f_{(i+1)}^o$. Note that once the boundaries are set, they remain same until frequency modification.[3]

Generation of Eligible Tokens: In cryptography, $\lambda \in \mathbb{N}$ is a security parameter, i.e., a variable measuring the probability with which an adversary can break a cryptographic scheme [60]. In other words, λ provides a way of measuring how difficult it is for an adversary to break a cryptographic scheme. *FreqyWM* requires randomization to be secure by ensuring that an attacker has only negligible advantage to recover the watermark and create collision for false claim (e.g., coming up with another watermarking secret which returns accept on

[3]The frequencies of some tokens may have high importance. An owner can filter the dataset and exclude them from watermarking.

data not watermarked by it). Thus, we choose a hash function to overcome the collision. In detail, a hash function H (chosen from a family of such functions) is a deterministic function from an arbitrary size input to a fixed size output, denoted $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$. The hash function [60] is *collision resistant* if it is hard to find two different inputs $m_0 \neq m_1$ that hash to the same output $H(m_0) = H(m_1)$.

Based on the above, to determine token pairs for watermarking, *FreqyWM* first generates a high entropy random number, i.e., secret, $R \leftarrow \{0,1\}^\lambda$ and an integer $z \in \mathbb{Z}^+$. Then, it uses R and z to compute s_{ij} values for modulo operation as: $s_{ij} = H(tk_i||H(R||tk_j)) \mod z$, where $||$ denotes concatenation. A set L_e of all eligible pairs is generated by an algorithm $Eligible$ based on given pairs $\{tk_i, tk_j\}$ and corresponding s_{ij} values as $L_e \leftarrow Eligible(D_o^{hist}, \{s_{ij}\})$. A pair is accepted as eligible if it satisfies that the boundaries of each token in the pair are at least $\lceil s_{ij}/2 \rceil$ where $s_{ij} \geq 2$. s_{ij} cannot be 0 or 1 because of modulo operation since modulo 0 is undefined and modulo 1 is 0. Note that the size of L_e is bounded by $[0, \binom{|D_o^{hist}|}{2}]$ where 0 means that there is *no eligible pair* while $\binom{|D_o^{hist}|}{2}$ means that *all the possible pairs* of tokens are eligible. After the eligible pairs are constituted, the boundary check is not necessary anymore since whichever set of pairs (that does not have a common token among) is chosen, the ranking will be preserved.

Optimal Selection: The eligible pairs are defined by ensuring the ranking constraint. However, to determine which subset of eligible pairs shall be selected such that chosen optimal number of pairs of tokens, denoted by a set L_{wm}, respect the budget constraint, it runs optimal matching algorithm from the eligible pairs L_e using the frequencies and s_{ij} values as $L_{wm} \leftarrow OptMatch(D_o^{hist}, L_e, \{s_{ij}\}, b)$. In Section III-B2, we show that for our optimal selection solution, we acutely reduce our problem to *Maximum Weight Matching (MWM)* and *Equally Valued 0/1 Knapsack problem (QKP)* problems to solve. We also devise two heuristics: *greedy* and *random*.

Frequency Modification: Based on L_{wm}, the algorithm creates new frequencies of tokens chosen from the optimal matching algorithm $(f_i^w - f_j^w) \mod s_{ij} \equiv 0$. This, of course, changes the boundaries of tokens; however, we do not need to update the boundaries as they are not needed anymore.

Data Transformation: It generates or removes tokens based on new frequencies. Note that the position of where to add tokens is important for security of *FreqyWM* against guess attack. Therefore, new tokens should be added in random positions (see Section IV-C for more discussion). As a final step, it returns the list of tokens D_w and stores L_{wm}, z value, and the random value R as a list L_{sc}.

2) Optimal and Heuristic Approaches : Given that all watermarked pairs have equal value in terms of proving ownership of the data, an optimal watermark is just a watermark of maximum size in terms of watermarked pairs, within the defined constraints (*similarity* and *ranking*). *Optimal Matching*. Let us now define our optimal watermarking. Let $G = \{V, E\}$ be a connected undirected graph which is the representation of frequencies driven from eligible pairs L_e. $V = \{v_1, v_2, \ldots, v_{|V|}\}$ where v_i represents tk_i and $E = \{e_1, e_2, \ldots, e_{|E|}\}$ where $e(v_i, v_j)$ is the edge between v_i and v_j. The weight of an edge $e(v_i, v_j)$, $w(e_i)$, is equal to $T - ((f_i^o - f_j^o) \mod s_{ij})$ where T is a big value (e.g., $T > C$ where C is the highest difference between two frequencies in the eligible pairs). Then, our optimal watermarking problem reduces to finding the maximum number of edges (pairs) such that *no edge* has a common vertex and b is not exceeded.

Definition 1 (Optimal Watermarking)**.** *Let $OptWM(G(V,E), b)$ be the optimal watermarking with a budget of b among an eligible set of items L_e represented as a connected undirected graph $G(V,E)$. The optimal watermarking produces the maximum number of edges (pairs) while not exceeding the budget b defined below:*

$$MAX\ |M^w|,\ \ M^w = \{e_1, ..., e_{|M|}\}\ s.t.\ sim(D_o^{hist}, D_w^{hist}) \geq (100 - b)$$

where M^w denotes the chosen pairs for watermarking.

This problem is reduced to two well-known problems with polynomial time solutions: *Maximum Weight Matching* (MWM) and *Equally Valued 0/1 Knapsack problem* (QKP) (which we have a special case where all values are equal). While the general $0/1$ Knapsack problem is known to be NP-Hard [42], this special equally valued $0/1$ Knapsack problem would have a polynomial time (greedy) solution. Hence, our optimal pairing problem is reduced and solved as follows:

• Find the maximum weight matching $M = e_1, e_2, \ldots, e_{|M|}$ as $M = MWM(G(V,E))$. Notice that M includes the edges such that the sum gives the maximum weight. It refers to minimum weight for us since the weights are defined as $T - (f_i - f_j \mod s_{ij})$ which makes the highest frequency difference have the smallest weight and the smallest one have the highest weight. With this conversion, we identify the edges distorting the histogram minimally.

• After finding the edges via MWM, we have one more constraint which is the budget b. The matching algorithm has to return the maximum number of matchings for which the distortion (e.g., based on cosine similarity) does not exceed b which can be solved via QKP where the value of each item is 1, and the weight is recomputed as $T - w(e_i)$. Recomputation is necessary because for the QKP we want to add as many items as possible that will be bounded by b. Therefore, it finds the set of edges L_{wm} in M such that the selected edges do not exceed the budget b by employing the QKP as $L_{wm} = QKP(M, b)$ where $L_{wm} = e_1, e_2, \ldots, e_{|L_{wm}|}$ and value of each e_i is 1 ($val(e_i) = 1$). Showing the optimality of the resulting watermark according to Definition 1 can be proven via proof-by-contradiction. In a nutshell, if our solution is not optimal, it means that one of the solutions produced by MWM and QKP cannot be optimal. However, since MWM and QKP are both assumed optimal, this contradicts our statement and thus our solution is optimal.

Heuristic Matching Algorithms. We define two heuristic algorithms: 1) *greedy*; and 2) *random*. In the *greedy* algorithm, all the eligible token pairs are sorted in an ascending order by their remainders as $rm_{ij} \equiv (f_i^o - f_j^o) \mod s_{ij}$. The algorithm

starts selecting a pair respectively for watermarking where b would not be exceeded when it is chosen (i.e., comparing the similarities of original and watermark histograms). This continues until b is exhausted or there is no more item to visit. The *random* matching algorithm follows the same steps as the greedy algorithm except it does not sort the eligible pairs but rather selects a pair randomly from L_e.

3) Watermark Detection: In detection, the data owner wishes to know if there is a watermark of its in a token dataset D'_w to claim ownership. The owner holds its secret list $L_{sc} = \{L_{wm}, R, z\}$ where L_{wm} is the list of watermarked token pairs, R is the high entropy value, and z is the (modulo) integer, all generated by the watermark generation, along with two thresholds: (1) t, a *threshold* to decide if a certain pair is watermarked; and (2) k, the *minimum number of watermarked pairs* required to conclude whether D'_w is a watermarked dataset. How to set t and k depends on the robustness an owner wants (see Sections III-B4 and IV-A2). If the owner wants to prove ownership to a third party, it has to reveal its secrets to that party. This causes to prove the ownership once in public (see Section V-D). Our watermark detection algorithm (Algorithm II) proceeds as follows:

(1) It builds the histogram list D_w^{hist} of the suspected dataset D'_w as in the watermark generation algorithm. The algorithm does not calculate the boundaries, just the token frequencies.

(2) For each token pair $\{tk_i, tk_j\}$ in L_{wm}, if the pair exists in D_w^{hist}, the algorithm generates s_{ij} values as $s_{ij} = H(tk_i||H(R||tk_j)) \mod z$.

(3) Then, it decides whether it will accept a given token pair (tk_i, tk_j), as watermarked or not by checking if the following statement holds: $(f_i - f_j) \mod s_{ij} \leq t$.

(4) After finding which pairs are watermarked, it checks whether their number is over the *minimum number of pairs*, k, needed to conclude that D'_w is watermarked by the owner, and returns $accept$ (verified) or $reject$, accordingly.

4) Probabilistic Analysis of False Positives: We develop a statistical bound in the form of the closed form expression derived from Markov's inequality theorem, to demonstrate that the false positive probability (i.e., accepting a dataset as watermarked when it is not) goes to zero if we increase the minimum number of pairs k that has to be accepted, or if we decrease the threshold t to accept a pair as watermarked. Recall that the m-th token pair $\{tk_i, tk_j\} \in L_{wm}$ is accepted as watermarked, if $(f_i - f_j) \mod s_{ij} \leq t$. We represent the probability that this "watermarking statement" holds as $P(X_m = 1) = p_m$, for $m = 1, ..., n$. Let us assume that p_m's follow a `Uniform[0,1]` distribution. The probability of having at least k successes in n trials can be written as $P(S_n \geq k) = \sum_{i=k}^{n} P(S_n = m)$. We now study the behavior of $P(S_n \geq k)$ depending on the behavior of t and k by using the *Sandwich Rule* and Markov's upper bound obtained by its inequality theorem $P(S_n \geq k) \leq \frac{\mu}{k}$, where $\mu = \sum_{m=1}^{n} p_m$ is the mean of S_n. Our analysis shows that if we decrease t, the probability of accepting a dataset as watermarked goes to zero and if we increase k, it will be hard to "accept" a dataset as watermarked. For further details, see the full version [61].

Algorithm II: Watermark Detection

Input: $D'_w, L_{sc} = \{L_{wm}, R, z\}, k, t$
Output: $accept/reject$
D_w^{hist} = **Preprocess**(D'_w) $count = 0$, $result = reject$
foreach $\{tk_i, tk_j\} \in L_{wm}$ **do**
  if ***Found***(tk_i, tk_j, D_w^{hist}) **then**
    $s_{ij} = H(tk_i||H(R||tk_j)) \mod z$
    if $(f_i - f_j) \mod s_{ij} \leq t$ **then**
      $count++$
    end
  end
end
if $count \geq k$ **then**
  $result = accept$
end
Result: result

IV. Experimental Evaluation

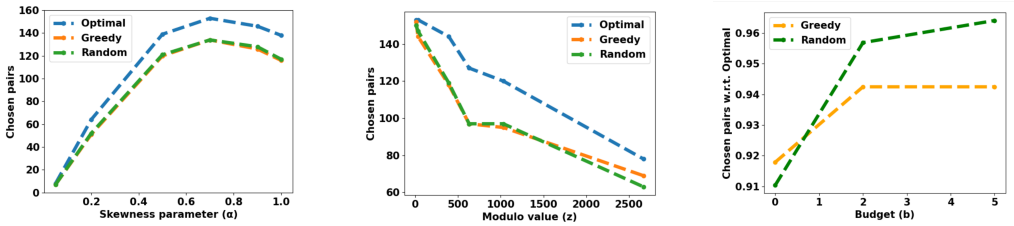
All of our experimental results are produced on a standard laptop machine with dual-core Intel Core(TM) i7 − 5600U CPU 2.5GHz, 16.00 GB RAM, 64-bit OS, and implemented in Python language. We deployed SHA256 as a hash function.

A. Synthetic Experiments

For our synthetic experiments, we generated synthetic datasets using a $power - law$ distribution [64] with different skewness values α as $[0.05, 0.2, 0.5, 0.7, 0.9, 1]$. The sample size is $1M$ and the number of tokens is $1K$ for each different α value. When α is 0, it is a uniform distribution in which there are no eligible tokens to watermark. When α is 1, the original dataset D_o is skewed with a very long tail with almost equal values. In this setting, we evaluate how the parameters (a modulo value z, a budget b, and skewness parameter α) are affecting the number of chosen pairs for watermarking and the performance of *optimal*, *greedy*, and *random* approached in terms of number of chosen pairs.

Figure 2a shows the correlation between skewness of a dataset α and the size of chosen pairs when budget $b = 2$ and modulo value $z = 1031$. When a dataset is almost uniform (i.e., $\alpha = 0.05$), the solutions can select very few pairs since there are not many eligible items (i.e., the upper and lower boundaries are not enough, in fact many of them are 0). When α starts increasing, the differences between the frequencies of tokens increase. Thus, the number of eligible items increases which also affects the number of chosen pairs under a given budget. However, at some point (i.e., $\alpha = 0.7$), the number of chosen pairs decreases due to the tail of (histogram) frequencies becoming uniform. The same figure shows the superior performance of the optimal solution. The gap between optimal and the heuristics is around 20% for most α values while the two heuristics perform similar the one with the other (0.02% in average).

Figure 2b illustrates how the modulo value z affects the size of chosen pairs. When we pick smaller modulo value z, we would have a higher number of chosen pairs. The reason is that a smaller z leads to smaller remainders s_{ij} that need to be eliminated, thereby yielding more selected pairs within a given budget b. When z is very small (i.e., 10), the three



(a) Effect of different skewness parameters (α) on chosen pairs by *Optimal*, *Greedy*, and *Random*.

(b) Effect of different modulo values (z) on chosen pairs by *Optimal*, *Greedy*, and *Random*.

(c) Chosen pairs by *Greedy* and *Random* with respect to *Optimal*.

Fig. 2: Effects of parameters on the size of chosen pairs for watermarking.

Dataset	Size	Token	Distinct Tokens	$\lvert L_e \rvert$	Optimal	Greedy	Random	Gen (sec)	Detect (sec)
Chicago Taxi [62]	9.68GB	`Taxi ID`	6573	33308	805	770	773	182.51	0.609
eyeWnder [58]	247MB	`URL`	11479	257	38	33	31	420.81	0.053
Adult [63]	4MB	`Age`	73	72	21	20	17	0.03	0.001

TABLE II: Validation results on real world datasets. **Dataset**: Dataset used **Size**: The size of original dataset. **Token**: Definition of the token (e.g., the name(s) of the attributes). **Distinct Token**: The number of distinct tokens. $|L_e|$: The number of eligible pairs. **Optimal**: The number of chosen pairs by the optimal matching. **Greedy**: The number of chosen pairs by the greedy matching. **Random**: The number of chosen pairs by the random matching. **Gen**: Running time of watermark generation. **Detect**: Running time of watermark detection.

approaches are very close (also see Section V-A for the effect of z in terms of security). However, when z increases, our optimal approach selects many more pairs than greedy and random. Figure 2c shows how the budget selection affects the performance comparison between the heuristics and the optimal. We set modulo value $z = 1031$ and use the dataset with the skewness $\alpha = 0.7$. When we increase the budget b, the heuristics get closer to the optimal performance. This is expected since even the optimal algorithm cannot select more than all the eligible pairs and with a large budget even the heuristics can approach that.

1) Limit of z: We stated that z is selected from Z^+; however, by analyzing the frequency histogram we can derive the upper and lower boundaries. Since the minimum value (lower bound) z can take is 2, we delve into investigation of the upper bound of z. Note that since the token with the highest frequency has the upper bound of infinity, there will be *at least* one pair that could be used for watermarking. Assume a watermarking pair candidate (tk_i, tk_j). Their frequencies, f_i and f_j, are changed such that $(f_i^w - f_j^w) \mod s_{ij} \equiv 0$. To have an upper bound for z, let us investigate which pair of tokens results in the highest difference. If we can determine the highest difference, say r_{max}, then r_{max} can be assumed as the upper bound for z since it is the highest remainder. Now, considering D_o^{hist}, the highest difference is between tk_0 (the token having the highest frequency) and $tk_{|D_o^{hist}|-1}$ (the token having the lowest frequency). That means that the largest remainder for any pair is $r_{max} = (f_0^0 - f_{|D_o^{hist}|-1}^0)$. Thus it is natural to accept that the upper bound of z is r_{max}. To conclude, z can be chosen from $(2, r_{max})$. Overall, r_{max} can be calculated as $\forall f_i, f_j \in D_o^{hist}$ s.t. $f_i \geq f_j$; $r_{max} = \max(\{f_i - f_j\})$. Hence, the upper bound for z is calculated. However, note that this value can be small and can be an advantage to an attacker. As discussed in Section IV-A, z affects the number of chosen pairs; thus, it correlates with the mix of possible attacks and is use-case scenario dependent. We plan to investigate this observation theoretically and experimentally in terms of security, robustness, and utility in the future.

2) Limit of t: Another critical parameter is $t \in [0, s_{ij} - 1]$. Note that since s_{ij} has an upper bound as $z - 1$, the highest value assigned to t is $z - 1$. While in our experimental study we chose t as a constant value, t could be also a percentage. Assume that an owner wishes to state that it wants 50% of error tolerance. Now, setting $t = s_{ij} \times 0.5$ states that a pair, say tk_i and tk_j, will be accepted as a watermarked if $(f_i - f_j) \mod s_{ij} \leq s_{ij}/2$. Thus, t represents the robustness level an owner desires. For instance, if $t = 0$ then the watermark becomes fragile since it cannot tolerate any changes in D_w, thus missing watermarked pairs (i.e., high false negatives). On the other hand, when $t = 100$, it is more robust and can tolerate modifications in D_w; however, it also means that it accepts more false positives (see also Section III-B4).

B. Validation Using Real World Datasets

Next we apply *FreqyWM* to three real world datasets from different domains: (1) `Chicago Taxi` dataset [62]; (2) A real click-stream dataset logging the URLs visited by a group of users of the `eyeWnder` advertisement detection add-on [58]; (3) `Adult` dataset [63]. Our intention is to validate our main conclusion using real data from the previous evaluation with synthetic data, i.e., that the heuristic approaches perform well enough compared to the optimal. Furthermore, we aim to report the real processing time on an ordinary machine for generating and detecting the watermark using these datasets.

For our watermark generation, we set the modulo value $z = 131$ and the budget $b = 2$. We run our algorithm 30 times and take the mean of total computations. Table II presents our validation results. `Taxi ID`, `URL`, and `Age` were chosen as tokens for `Chicago Taxi`, `eyeWnder`, and `Adult`, respectively. After generation, for `Chicago Taxi`, `eyeWnder`, and `Adult` datasets, our optimal solution chose 805, 38, and 21 pairs, respectively. Considering the heuristic approaches, greedy chose 770 pairs for `Chicago Taxi`, 33 pairs for `eyeWnder`, and 20 pairs `Adult` while random chose 773, 31, and 17 pairs, for `Chicago Taxi`, `eyeWnder`, and `Adult`, respectively. Running times of computations for watermark generation on the `Chicago Taxi`, `eyeWnder`, and `Adult` datasets were 182.51 secs, 420.81 secs, and 0.03 secs, respectively (where we exclude histogram and watermarked data generations). For watermark detection, the total detection time for each watermarked datasets was less than 1 sec. As it can be interpreted from Table II, the number of chosen pairs increases when the number of eligible pairs increases. For instance, while `eyeWnder` has more distinct tokens (11479) than `Chicago Taxi` has (6573), `eyeWnder` has fewer eligible pairs (257) than `Chicago Taxi` has (33308). Thus, *FreqyWM* has selected more pairs for `Chicago Taxi` (805) than it selected for `eyeWnder` (38).

C. Watermarking Multi-Dimensional Data

During our discussion so far, we set the token as a single attribute. However, as we previously stated, a token does not necessarily need to be restricted to a single attribute of a multi-dimensional dataset. Therefore, a token can be also defined as combination of more than one attributes (e.g., `[Age, WorkClass]`) in the `Adult` dataset. We ran *FreqyWM* on such token represented as `[Age, WorkClass]` with the same parameter setting in Section IV-B and the number of tokens (i.e., distinct `[Age, WorkClass]` attributes in the real dataset) were 481. The size of pairs chosen for watermarking was 20. With multi-dimensional data removing a token appearance is as simple as with single-dimensional data. Increasing, however, a token's frequency is more involved. The reason is that just repeating the value of the token would leave other fields not being part of the token with a value to be set, e.g., all the other fields beyond Age and WorkClass in the `Adult` dataset. A naive solution would be select a random appearance of the token and copy its other fields every time that an additional instance of the token needs to be added to the watermarked dataset. This, however, could create semantic inconsistencies if there are constraints to be respected for individual attributes or combinations of them. Making sure that added appearances of a token do not lead to semantic inconsistencies that, in addition to degrading the quality of the data, could also give away the existence of a watermarked pair to an attacker. This analysis requires domain knowledge about what each dataset represents. Such knowledge, however, is orthogonal to all previous steps of our algorithms and, thus, can be appended as a last step based on one's domain knowledge of the data whenever a token's frequency needs to be increased. We are currently investigating them and the effect of *FreqyWM* on data utility of such dataset with unique attributes as it is difficult to determine as addressed by [23].

D. Comparison to Related Works

As stated previously, we cannot directly apply (numerical) database watermarking to datasets similar to the ones we used for validation. However, one naive approach would be to convert a dataset to a numerical representation (e.g., a histogram) and watermark this numerical representation. In a nutshell, the histogram of a given dataset based on a predefined token is generated and then, the histogram is treated as a two dimensional database consisting of primary keys which are the tokens and an attribute consisting of integer values which are the frequencies. Later, a database watermarking is employed on this histogram. Then as in *FreqyWM* data transformation (e.g., removing/adding tokens) occurs according to the (new) watermarked histogram computed by the database watermarking. However, applying a numerical database watermarking is not really straightforward since it will distort the underlying dataset unexpectedly as a result of change in histogram data (e.g., cosine similarity) and would require modification in their watermarking techniques (e.g., how to create a watermarked dataset from the watermarked numerical representation). However, since this is the closest and simplest approach, we compare against it.

To actualize the above approach, we considered two numerical database watermarkings: 1) Shehab et al. [35] (referred as WM-OBT) due to partitioning approach (i.e., grouping tokens before watermarking) similar to *FreqyWM*; 2) Li et al. [30] (referred as WM-RVS) due to being one of the most recent reversible watermarking schemes introducing very small distortion compared to other same family of watermarkings.

More specifically, WM-OBT follows a data partition approach in which a watermark, defined as a bit sequence, is inserted on a group of partitions. Each data partition is filled by tokens and the frequencies of the tokens in each partition are modified/distorted by solving a minimization (if a watermark bit is 0) or maximization (if a watermark bit is 1) problem via a genetic algorithm [65], in which the objective function is in the form of a sum of sigmoid functions. WM-RVS treats each numeric value individually and changes its decimal part by selecting the random least significant position based on the watermarking key/bit and attributes. Furthermore, to apply WM-OBT and WM-RVS on a histogram generated from a dataset, we adjusted them such that their solutions produced are integers since a frequency count cannot be a decimal value.

For comparison, we investigate them based on two constraints: 1) change in the original histogram after watermarking (i.e., cosine similarity with watermarked histogram), and 2) the ranking of the tokens after watermarking.

We ran *FreqyWM*, WM-OBT, and WM-RVS on our synthetic data with skewness parameter 0.5 (with $1K$ distinct tokens and $1M$ sample size) where we set $b = 2$, and $z = 131$ for *FreqyWM*. We set parameters for WM-OBT and WM-RVS such that the parameters are proportional to the

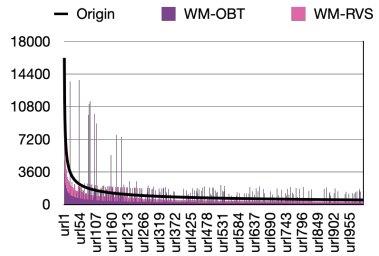


Fig. 3: Comparisons of the watermarked histograms generated from WM-OBT (purple) and WM-RVS (fuchsia) w.r.t. the original data histogram (black) for the synthetic dataset with dummy token names.

experimental settings of Shehab et al. [35] and Li et al. [30]. For WM-OBT, we use genetic algorithm (GA) technique for optimization [65] where we fix the number of partitions as 20 (where each partition has around 50 tokens), watermark bit sequence as $[1, 1, 0, 1, 0]$, condition as 0.75, and we allow the change (constraint) between $[-0.5, 10]$. The decoding threshold minimizing the probability of decoding error is calculated as 0.0966. For WM-RVS, we use the same bit sequence as in WM-OBT without creating it from the chaotic encryption. Also, let us note that WM-OBT took more than 30 minutes to run for such a small size dataset due to its optimization while WM-RVS was in the order of seconds. Figure 3 visualizes how the watermarked data histograms look like with respect to the original data histogram after applying WM-OBT and WM-RVS based on the experiments.

Similarity. In *FreqyWM*, even with 2% budget, the similarity between the original histogram and the watermarked histogram is 99.9998%, indicating that not all the budget was exhausted. On the other hand, for WM-OBT and WM-RVS, the similarities are 54.28% and 96%, respectively. The mean and standard deviation of the changes introduced to the histogram by WM-OBT are 444 and 855.91, respectively while they are -69.43 and 414.10 for WM-RVS, respectively.

Ranking. Preserving the ranking is important because it allows us not to sacrifice the utility of a dataset, e.g., preserving the popularity of URLs. *FreqyWM* by definition maintains the ranking of tokens. However, our analysis showed that WM-OBT and WM-RVS changed the ranking of 998 and 987 out of the total 1000 tokens, respectively!

The results on similarity and ranking support our claim that applying a numerical database technique on histogram data would result in unexpected and uncontrolled distortion that seriously undermines the utility of the original data.

V. Security and Robustness Analysis

This section discusses the security and robustness of our *FreqyWM* method against four attacks: **guess**, **sampling**, **destroy**, and **re-watermarking (false-claim)** attacks which are well-known attacks in watermarking as studied by [23].In order to measure the robustness against sampling and destroy attacks, we run our optimal solution on a dataset where the skewness parameter $\alpha = 0.5$ (with $1K$ distinct tokens and $1M$ sample size), unless stated otherwise, the modulo value $z = 131$, and the budget $b = 2$ and it selected 139 pairs for watermark. We run the experiments for 100 times and compute the average accepted pairs over all repetitions.

A. Guess (Brute-Force) Attack

In the guess attack, the probabilistic polynomial time adversary tries to guess the watermark, i.e., the secret embedded in the data. This is possible only if it can figure out a subset of token pairs $\{tk_i, tk_j\}_l$ (where $\binom{|D_w^{hist}|}{2} \geq l \geq k$) based on the watermarked data D_w, the random value R, and the modulo value z where the watermark detection algorithm based on these inputs (for some fixed k and t) returns `accept`. Assuming that the hash function is collision resistant, R is random, and z is an integer, the probability of the attacker being successful can be formally defined as:

$$Pr[R \leftarrow \{0,1\}^{\lambda}; (D_w, L_{sc} = \{\{tk_i, tk_j\}_{|L_{wm}|}, R, z\}) \leftarrow WmGenerate(D_o, b) : \mathcal{A}(D_w) \rightarrow L'_{sc} = \{\{tk_i, tk_j\}_l, R^*, z^*\}| WmDetect(D_w, L'_{sc}, k, t) = 1] \leq \mathsf{negl}(\lambda)$$

Considering the typical parameter values, the probability of success becomes negligible.

B. Sampling Attack

In this attack, $\mathcal{A}$ copies a random subsample from the watermarked dataset D_w in an attempt to exploit (pirate/steal) it while hoping that the watermark won't be detectable within the extracted sample. The attack is run for different sample sizes from 1% to 90%, extracted from the original watermarked dataset D_w. For each percentage and subsample we apply the detection algorithm and compute the percentage of accepted pairs. Also, for each subsample detection experiment, we deploy different values of the threshold t for accepting a pair as watermarked as $t = \{0, 1, 2, 4, 10\}$. The attack scenario is as follows: $\mathcal{A}$ randomly selects $x\%$ of D_w where x defines the percentage for the sampling attack (e.g., 1) as a subsample size of $1M \times \frac{x}{100}$. When the owner suspects the dataset (possible subsampled), it scales it up to the size of D_w by multiplying the frequency counts by $\frac{100}{x}$ by using its info from the (original) watermarked dataset (e.g., via info added to its metadata). For instance, for 1% sampling attack, a subsample would have total of $1M \times 0.01 = 10K$ where each f_i is multiplied by approximately 0.01. Note that if the sample size is greater than the number of distinct tokens, which is the number of items in D_w^{hist}, the sample will have all the distinct tokens with a high chance. This also means that all the chosen watermarked pairs are in the subsample. Our results show that the size of the extracted subsample does not greatly affect the number of accepted pairs if it is greater than the number of unique tokens ($1K$). Since the frequencies of the tokens vary, the value of t does affect the result of the detection. For example, with $t = 0$ the detection algorithm can detect around 36% (in average) of the watermarked pairs. When t increases from 1 to 10, the performance of the detection increases (in average) from 72% to 99.5%.

Let us now see the results when the size of the extracted subsample is very low, so that it might not contain at least

1 token from the total $1K$ of unique tokens that the original watermarked dataset has. Figure 4 shows the results for sample size proportions between 0.0007% and 0.5%. Observe that if the sample size is greater than $5\times$ the number of unique tokens ($1K$), the detection algorithm stabilizes its performance for detecting the watermark. Below $2\times$ ($2K$), the performance starts to decrease with higher velocity. In these extreme cases, the detection algorithm will have more difficulties to detect the data as watermarked. However, the utility of the data is highly degraded since the subsample sizes are very small compared to the original size of $1M$ tokens. This causes a small number of distinct tokens to be found in the subsample.

Effect of modulo bases. As seen previously, t is crucial for detecting whether a pair is watermarked. For small values of t to be sufficient to fend off sampling attacks, the remainders need to be small numbers that are covered by t. One way to achieve this is by ensuring that the modulo bases used (i.e., the s_{ij}'s) are relatively small numbers when compared to the actual appearance frequencies of watermarked pairs. When this does not apply, the method will of course fail. For instance, assume a watermarked pair involving frequencies $f_i = 540, f_j = 440$ which under base $s_{ij} = 100$ leave a remainder of 0. W.l.o.g, lets assume that a 50% frequency attack leads to a dataset with $f_i = 270, f_j = 220$ which leads to a remainder of $(270 - 220) \mod 100 \equiv 50$. Now if t is chosen smaller than 50 then the watermarked pair will not be detected. The reason is that mod 100 leaves large remainders when applied to f_i and f_j that are in the same order with 100. In our experimental results f_i's were always much larger numbers than the employed s_{ij}, thereby, even small t's would detect a pair under a sampling attack. To determine the optimal t and how robust it is against attacks, a further investigation is needed as it depends on various parameters such as z, s_{ij} as well on the mix of expected attacks as discussed later. Note that our experimental results show that s_{ij} values are ~ 2 order of magnitude smaller than z. Furthermore, we also tested the sampling attack in other watermarked datasets with different values of the skewness parameter and obtained similar results.

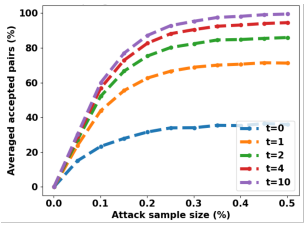


Fig. 4: Sampling attack results with very small sample and $\alpha = 0.5$.

C. Destroy Attack

In this case the attacker $\mathcal{A}$ tries to damage the watermark. The no-security-by-obscurity principle [66] allows $\mathcal{A}$ to know that it can destroy the watermark in a way that it cannot be detected by the owner. $\mathcal{A}$ computes the histogram of watermarked data D_w. $\mathcal{A}$ modifies the frequencies of tokens as it pleases by allowing re-ordering (changing the popularity/rank of the tokens) or without allowing re-ordering. We define these two attacks and discuss *FreqyWM*'s robustness against them.

1) Destroy Attack without re-ordering: In this attack type, $\mathcal{A}$ can modify the frequencies without changing the order of frequencies. We introduce two types: (1) attacker changes the frequencies randomly by the given boundaries and (2) attacker changes the frequencies by (at most) some percentage.

Changing the frequencies randomly within the boundaries. $\mathcal{A}$ calculates the boundaries for each token. Then, $\mathcal{A}$ chooses a random value r_i for each tk_i as $r_i \leftarrow (-l_i, u_i)$. $\mathcal{A}$ changes the frequency of tk_i and updates u_{i+1} of tk_{i+1} by r_i.

Changing the frequencies by (at most) some percentage. $\mathcal{A}$ changes the frequencies of tokens up to some percentage (e.g., 1%). To illustrate, $\mathcal{A}$ calculates the boundaries as u_i and l_i for each tk_i where it sets the percentage to 1%. It calculates $u'_i = floor(u_i \times 0.01)$ and $l'_i = floor(l_i \times 0.01)$. Then it gets a random value r_i between $(-l'_i, u'_i)$. It hereby changes tk_i by at most $\pm 1\%$. After every change ($f'_i = f_i + r_i$), the boundary of the next element is updated. Thus, the attack never changes the ranking/ordering since l'_i and u'_i are already in the boundaries.

Figure 5 shows how robust *FreqyWM* is against these two destroy attacks. We compare the success rate (the percentage of accepted token pairs given threshold for accepting a pair t) of detection algorithm with respect to modified watermarked data after the attacks. We also include in the figure a second dataset of skewness $\alpha = 0.7$ *that does not carry the watermark*, and report on how many of its pairs would be falsely verified for different values of t. For an attack in which the frequencies are changed by (at most) some percentage (represented by the red line in the figure), *FreqyWM* can detect around 90% of the pairs when $t = 0$. When t is increased, after a point where $t \geq f_i - f_j \mod s_{ij}$, the success rate converges at around 90%. For an attack where the frequencies are changed randomly within the upper and lower frequency boundaries (green line in Figure 5), *FreqyWM* can detect more than 35% of the pairs when $t = 0$. Note that the latter is more powerful than the former attack. There is a direct proportion between t and the success rate. As shown, the success rate reaches to 90% when t goes to 10.

From Figure 5, we can interpret in what parameter setting false negative (rejecting a watermarked pair) and false positive (accepting a pair as watermarked while it is not) can be avoided. Thus, the watermarking detection algorithm can successfully detect a watermarked dataset attacked and reject a dataset that was not watermarked. For instance, the rate of false positive increases when the threshold for accepting a pair t increases while the minimum number of accepted pairs for detection k decreases which is the area under the results of the dataset (not watermarked) with a different skewness parameter (the area under the orange line). On the other hand, the rate of false negative increases when the threshold accepting a pair t decreases while threshold for detecting a watermark k increases which is the area above the results of the attack without re-ordering (the area above the green line) if we consider a very strong attack. To avoid false negatives/positives, convenient parameter settings (i.e., t and k) for detecting a

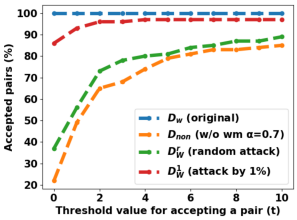


Fig. 5: Percentage of verified pairs for the following datasets: (1) D_w : the original watermarked dataset $\alpha = 0.5$ without any attack/modification, (2) D_{non} : a non-watermarked dataset defined over the same token space but with $\alpha = 0.7$, (3) D^r_w : D_w after attacked by random attack without reordering, (4) D^1_w : D_w after attacked by changing frequencies at most 1%.

watermark lie between these two areas (between the orange and the green lines in Figure 5). However, if a weaker attack (changing the frequencies by some percentage) is considered, the range of these parameters increases (the area between the red and orange line). Hence, the detection algorithm can detect a watermarked dataset and reject a dataset not watermarked by the owner with a careful parameter setting. For instance, adjusting t (and k) based on the nature of the data and the specific application context can enable us to reduce the false positives/negatives. This is an interesting future work.

2) Destroy Attack with re-ordering: In this attack type, an attacker $\mathcal{A}$ can modify the frequencies as it pleases without observing any ordering restrictions. Note that this attack introduces more noise than the attack without re-ordering which reduces the usability of watermarked data D_w. $\mathcal{A}$ modifies the frequencies with various percentages $[10\%, 30\%, 50\%, 60\%, 80\%, 90\%]$ where the success rates are $[94\%, 88\%, 82\%, 79\%, 78\%, 76\%]$ respectively. *FreqyWM* can detect the watermark with 76% chance up to modifications of 90% in frequencies approximately (where $t = 4$).

D. Re-watermarking/ False-Claim Attack

This attack is mounted by an attacker $\mathcal{A}$ creating a new watermark on the watermarked data D_w, generated by an honest owner. $\mathcal{A}$ generates its own watermarked data by simply inserting D_w into the watermark generation algorithm as data to produce $D^{\mathcal{A}}_w$. Then $\mathcal{A}$ can present $D^{\mathcal{A}}_w$ and claim the ownership of $D^{\mathcal{A}}_w$ (since $\mathcal{A}$ can prove its ownership claim by introducing its watermarking secret list $L^{\mathcal{A}}_{sc}$). This attack creates a dispute since both the real owner, who created D_w, and $\mathcal{A}$ have proofs of their ownerships. The dispute can be arbitrated by introducing a judge (a trusted third party as suggested by [67]) to the watermarking scheme. Both parties, $\mathcal{A}$ and the real owner, introduce their secrets and their watermarked data. $\mathcal{A}$ sends its secrets $L^{\mathcal{A}}_{sc}$ and its watermarked data $D^{\mathcal{A}}_w$. The real owner sends its secrets L_{sc} and its watermarked data D_w. The judge computes watermark detection algorithm on each received data for each secret which creates four outputs. The judge compares these results and identifies the real owner since only the secret of the real owner can produce accept on both data. To show practicality of our defense against the re-watermarking attack, we implemented the attack above. Our results show that the first watermark is detected with 92% on $D^{\mathcal{A}}_w$ under $t = 0$. The attacker's only way to succeed is to perform successful guess or destroy attack which it cannot perform as shown previously.

VI. Discussion

We propose possible adjustments to *FreqyWM* for more sophisticated properties and discuss some challenges as follows:

• *Incremental FreqyWM.* In the literature, there exist watermarking techniques that allow to update a watermark on a dataset without computing insertion from scratch [57]. We believe that an incremental FreqyWM can be built on top of dynamic maximum weighted matching [68, 69] works but we leave such investigations to future work.

• *Multi-watermarks.* One might watermark a dataset multiple times with different intentions: 1) to have a chronological order in the versions (e.g., data provenance); and 2) to falsely claim ownership (see Section V-D). Non-withstanding the motivation, we run an experiment to calculate the discrepancy between the original (histogram) one and the final one after 10 insertions assuming a budget $b = 0.002$ for each iteration on a sample dataset with $\alpha = 0.5$. The resulting similarity between the original and the latest watermarked version is 0.003%. As it is evident, *FreqyWM* did not introduce 0.02% but rather very tiny distortion (see also the full version [61] for further analysis, i.e., ML analysis). Hence, successive re-watermaking can be practical with *FreqyWM*, but we plan to extensively investigate it in the future.

•*Challenging datasets.* Apart from datasets with close to uniform frequencies, *FreqyWM* can also be challenged when the range of token values is too wide, e.g., sales' datasets with many decimal values, resulting to very few (if any) repetition of the same value. One natural solution to this is to first bucketize (cluster) the widely ranged data and then apply *FreqyWM* at the level of the bucket as opposed to the exact token value.

VII. Conclusions and Future Work

We proposed *FreqyWM*, a novel frequency watermarking technique for protecting the ownership of data in the emerging new data economy. We analysed the performance of *FreqyWM* and showed how *FreqyWM* can encode watermarks with minimal distortion on the original data, provided that the data has sufficient variability in terms of token frequencies. We analysed *FreqyWM*'s robustness to generic attacks. *FreqyWM* is applicable to large numbers of tuples sold in wholesale manner in modern DMs. An interesting, yet challenging, research direction is to consider how to watermark small sets or even individual tuples used in distributed data operations such as replication and remote hosting and/or query execution. We are currently looking at more attack scenarios and at devising systematic procedures for optimizing the parameters and also how to apply *FreqyWM* to multidimensional datasets by overcoming the challenges mentioned in Section IV-C. We also investigate integrating data privacy (e.g., differentially-private fingerprinting [70]).

Acknowledgements

Devriş İşler was supported by the European Union's HORIZON project DataBri-X (101070069). Nikolaos Laoutaris was supported by the MLEDGE project (REGAGE22e00052829516), funded by the Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU/PRTR.

References

[1] S. A. Azcoitia and N. Laoutaris, "A survey of data marketplaces and their business models," *SIGMOD Rec.*, vol. 51, no. 3, pp. 18–29, 2022. [Online]. Available: https://doi.org/10.1145/3572751.3572755

[2] A. Lutu, D. Perino, M. Bagnulo, E. Frias-Martinez, and J. Khangosstar, "A characterization of the covid-19 pandemic impact on a mobile network operator traffic," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3419394.3423655

[3] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Found. Trends Priv. Secur.*, 2018. [Online]. Available: https://doi.org/10.1561/3300000019

[4] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, USA, 2009. [Online]. Available: https://searchworks.stanford.edu/view/8493082

[5] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Theory of Cryptography Conference, TCC*. Springer, 2011. [Online]. Available: https://doi.org/10.1007/978-3-642-19571-6_16

[6] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *TrustCom/BigDataSE/ISPA*. IEEE, 2015. [Online]. Available: https://doi.org/10.1109/Trustcom.2015.357

[7] Y. Li, D. Ghosh, P. Gupta, S. Mehrotra, N. Panwar, and S. Sharma, "PRISM: private verifiable set computation over multi-owner outsourced databases," in *SIGMOD: International Conference on Management of Data, Virtual*. ACM, 2021. [Online]. Available: https://doi.org/10.1145/3448016.3452839

[8] R. Poddar, T. Boelter, and R. A. Popa, "Arx: An encrypted database using semantically secure encryption," *Proc. VLDB Endow.*, 2019. [Online]. Available: http://www.vldb.org/pvldb/vol12/p1664-poddar.pdf

[9] N. Anciaux, L. Bouganim, P. Pucheral, I. S. Popa, and G. Scerri, "Personal database security and trusted execution environments: A tutorial at the crossroads," *Proc. VLDB Endow.*, 2019. [Online]. Available: http://www.vldb.org/pvldb/vol12/p1994-anciaux.pdf

[10] X. Ren, L. Su, Z. Gu, S. Wang, F. Li, Y. Xie, S. Bian, C. Li, and F. Zhang, "HEDA: multi-attribute unbounded aggregation over homomorphically encrypted database," *Proc. VLDB Endow.*, 2022. [Online]. Available: https://www.vldb.org/pvldb/vol16/p601-gu.pdf

[11] W. Zhou, Y. Cai, Y. Peng, S. Wang, K. Ma, and F. Li, "Veridb: An sgx-based verifiable database," in *SIGMOD: International Conference on Management of Data*. ACM, 2021. [Online]. Available: https://doi.org/10.1145/3448016.3457308

[12] P. Jougleux, "Data ownership (and succession law)," in *Facebook and the (EU) Law: How the Social Network Reshaped the Legal Framework*. Springer, 2022, pp. 129–143.

[13] J. Kennedy, P. Subramaniam, S. Galhotra, and R. C. Fernandez, "Revisiting online data markets in 2022: A seller and buyer perspective," *SIGMOD Rec.*, vol. 51, no. 3, pp. 30–37, 2022. [Online]. Available: https://doi.org/10.1145/3572751.3572757

[14] R. C. Fernandez, P. Subramaniam, and M. J. Franklin, "Data market platforms: Trading data assets to solve data problems," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 1933–1947, 2020. [Online]. Available: http://www.vldb.org/pvldb/vol13/p1933-fernandez.pdf

[15] F. Banterle, "Data ownership in the data economy: a european dilemma," *EU Internet Law in the Digital Era: Regulation and Enforcement*, pp. 199–225, 2020. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3277330

[16] M. Asikuzzaman and M. R. Pickering, "An overview of digital video watermarking," *IEEE Trans. Circuits Syst. Video Technol.*, 2018. [Online]. Available: https://doi.org/10.1109/TCSVT.2017.2712162

[17] M. Begum and M. S. Uddin, "Digital image watermarking techniques: A review," *Inf.*, 2020. [Online]. Available: https://doi.org/10.3390/info11020110

[18] H. Ma, C. Jia, S. Li, W. Zheng, and D. Wu, "Xmark: Dynamic software watermarking using collatz conjecture," *IEEE Trans. Inf. Forensics Secur.*, 2019. [Online]. Available: https://doi.org/10.1109/TIFS.2019.2908071

[19] X. Zhou, H. Pang, K. Tan, and D. Mangla, "Wmxml: A system for watermarking XML data," in *International Conference on Very Large Data Bases (VLDB)*. ACM, 2005. [Online]. Available: http://www.vldb.org/conf/2005/papers/p1318-zhou.pdf

[20] R. Agrawal and J. Kiernan, "Watermarking relational databases," in *Proceedings ofInternational Conference on Very Large Data Bases, VLDB*, 2002. [Online]. Available: http://www.vldb.org/conf/2002/S05P03.pdf

[21] R. Agrawal, P. J. Haas, and J. Kiernan, "A system for watermarking relational databases," in *ACM SIGMOD International Conference*, 2003. [Online]. Available: https://doi.org/10.1145/872757.872865

[22] T. Wang and F. Kerschbaum, "RIGA: covert and robust white-box watermarking of deep neural networks," in *WWW: The Web Conference*, 2021. [Online]. Available: https://doi.org/10.1145/3442381.3450000

[23] S. Rani and R. Halder, "Comparative analysis

of relational database watermarking techniques: An empirical study," *IEEE Access*, vol. 10, pp. 27 970–27 989, 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3157866

[24] N. Agarwal, A. K. Singh, and P. K. Singh, "Survey of robust and imperceptible watermarking," *Multim. Tools Appl.*, 2019. [Online]. Available: https://doi.org/10.1007/s11042-018-7128-5

[25] R. Agrawal, P. J. Haas, and J. Kiernan, "Watermarking relational data: framework, algorithms and analysis," *VLDB J.*, 2003. [Online]. Available: https://doi.org/10.1007/s00778-003-0097-x

[26] T. Ji, E. Yilmaz, E. Ayday, and P. Li, "The curse of correlations for robust fingerprinting of relational databases," in *RAID : International Symposium on Research in Attacks, Intrusions and Defenses*. ACM, 2021. [Online]. Available: https://doi.org/10.1145/3471621.3471853

[27] E. Quiring, D. Arp, and K. Rieck, "Forgotten siblings: Unifying attacks on machine learning and digital watermarking," in *IEEE European Symposium on Security and Privacy, EuroS&P*. IEEE, 2018. [Online]. Available: https://doi.org/10.1109/EuroSP.2018.00041

[28] A. Cohen, J. Holmgren, R. Nishimaki, V. Vaikuntanathan, and D. Wichs, "Watermarking cryptographic capabilities," *SIAM J. Comput.*, 2018. [Online]. Available: https://doi.org/10.1137/18M1164834

[29] X. Tang, Z. Cao, X. Dong, and J. Shen, "Pkmark: A robust zero-distortion blind reversible scheme for watermarking relational databases," in *IEEE International Conference on Big Data Science and Engineering*, 2021. [Online]. Available: https://doi.org/10.1109/BigDataSE53435.2021.00020

[30] W. Li, N. Li, J. Yan, Z. Zhang, P. Yu, and G. Long, "Secure and high-quality watermarking algorithms for relational database based on semantic," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–14, 2022.

[31] M. L. P. Gort, M. Olliaro, A. Cortesi, and C. F. Uribe, "Semantic-driven watermarking of relational textual databases," *Expert Syst. Appl.*, 2021. [Online]. Available: https://doi.org/10.1016/j.eswa.2020.114013

[32] C. Lin, T. Nguyen, and C. Chang, "LRW-CRDB: lossless robust watermarking scheme for categorical relational databases," *Symmetry*, 2021. [Online]. Available: https://doi.org/10.3390/sym13112191

[33] S. Kumar, B. K. Singh, and M. Yadav, "A recent survey on multimedia and database watermarking," *Multim. Tools Appl.*, vol. 79, no. 27-28, pp. 20 149–20 197, 2020. [Online]. Available: https://doi.org/10.1007/s11042-020-08881-y

[34] M. H. Jony, F. T. Johora, and J. F. Katha, "A robust and efficient numeric approach for relational database watermarking," in *IEEE International Conference on Sustainable Technologies for Industry 4.0 (STI)*, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9732582

[35] M. Shehab, E. Bertino, and A. Ghafoor, "Watermarking relational databases using optimization-based techniques," *IEEE Trans. Knowl. Data Eng.*, 2008. [Online]. Available: https://doi.org/10.1109/TKDE.2007.190668

[36] D. Ibosiola, B. A. Steer, Á. García-Recuero, G. Stringhini, S. Uhlig, and G. Tyson, "Movie pirates of the caribbean: Exploring illegal streaming cyberlockers," in *Proceedings of the Twelfth International Conference on Web and Social Media, ICWSM*. AAAI Press, 2018. [Online]. Available: https://aaai.org/ocs/index.php/ICWSM/ICWSM18/paper/view/17835

[37] W. Zhou, J. Hu, and S. Wang, "Enhanced locality-sensitive hashing for fingerprint forensics over large multi-sensor databases," *IEEE Trans. Big Data*, 2021. [Online]. Available: https://doi.org/10.1109/TBDATA.2017.2736547

[38] Y. Lei, Q. Huang, M. S. Kankanhalli, and A. K. H. Tung, "Locality-sensitive hashing scheme based on longest circular co-substring," in *Proceedings of the 2020 International Conference on Management of Data, SIGMOD*. ACM, 2020. [Online]. Available: https://doi.org/10.1145/3318464.3389778

[39] D. Chang, M. Ghosh, S. K. Sanadhya, M. Singh, and D. R. White, "Fbhash: A new similarity hashing scheme for digital forensics," *Digit. Investig.*, 2019. [Online]. Available: https://doi.org/10.1016/j.diin.2019.04.006

[40] C. N. K. Osiakwan and S. G. Akl, "The maximum weight perfect matching problem for complete weighted graphs is in pc*," *Parallel Algorithms Appl.*, 1995. [Online]. Available: https://doi.org/10.1080/10637199508915506

[41] Z. Galil, "Efficient algorithms for finding maximum matching in graphs," in *ACM CSUR*, 1986.

[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[43] E. Ayday, E. Yilmaz, and A. Yilmaz, "Robust optimization-based watermarking scheme for sequential data," in *International Symposium on Research in Attacks, Intrusions and Defenses, RAID*, 2019. [Online]. Available: https://www.usenix.org/conference/raid2019/presentation/ayday

[44] T. Ji, E. Ayday, E. Yilmaz, and P. Li, "Robust fingerprinting of genomic databases," *CoRR*, vol. abs/2204.01801, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2204.01801

[45] M. Kamran and M. Farooq, "A comprehensive survey of watermarking relational databases research," in *arXiv preprint arXiv:1801.08271*, 2018.

[46] A. S. Panah, R. G. van Schyndel, T. K. Sellis, and E. Bertino, "On the properties of non-media digital watermarking: A review of state of the art techniques," *IEEE Access*, 2016. [Online]. Available: https://doi.org/10.1109/ACCESS.2016.2570812

[47] M. E. Farfoura, S. Horng, J. Lai, R. Run, R. Chen, and M. K. Khan, "A blind reversible method for watermarking relational databases based on a time-stamping protocol," *Expert Syst. Appl.*, 2012. [Online].

Available: https://doi.org/10.1016/j.eswa.2011.09.005

[48] Y. Li and R. H. Deng, "Publicly verifiable ownership protection for relational databases," in *Proceedings of th ACM Symposium on Information, Computer and Communications Security, ASIACCS*. ACM, 2006. [Online]. Available: https://doi.org/10.1145/1128817.1128832

[49] D. Hu, D. Zhao, and S. Zheng, "A new robust approach for reversible database watermarking with distortion control," *IEEE Trans. Knowl. Data Eng.*, 2019. [Online]. Available: https://doi.org/10.1109/TKDE.2018.2851517

[50] H. M. El-Bakry and M. Hamada, "A novel watermark technique for relational databases," in *Artificial Intelligence and Computational Intelligence - International Conference, AICI 2010, Sanya, China, October 23-24, 2010, Proceedings, Part II*, ser. Lecture Notes in Computer Science. Springer, 2010. [Online]. Available: https://doi.org/10.1007/978-3-642-16527-6_29

[51] S. M. Darwish, H. A. Selim, and M. M. El-Sherbiny, "Distortion free database watermarking system based on intelligent mechanism for content integrity and ownership control," *J. Comput.*, 2018. [Online]. Available: https://doi.org/10.17706/jcp.13.9.1053-1066

[52] Y. Zhang, B. Yang, and X.-M. Niu, "Reversible watermarking for relational database authentication," 2008.

[53] W. Wang, C. Liu, Z. Wang, and T. Liang, "FBIPT: A new robust reversible database watermarking technique based on position tuples," in *International Conference on Data Intelligence and Security, ICDIS*. IEEE, 2022, pp. 67–74. [Online]. Available: https://doi.org/10.1109/ICDIS55630.2022.00018

[54] G. Gupta and J. Pieprzyk, "Reversible and blind database watermarking using difference expansion," *Int. J. Digit. Crime Forensics*, 2009. [Online]. Available: https://doi.org/10.4018/jdcf.2009040104

[55] K. Jawad and A. Khan, "Genetic algorithm and difference expansion based reversible watermarking for relational databases," *J. Syst. Softw.*, 2013. [Online]. Available: https://doi.org/10.1016/j.jss.2013.06.023

[56] M. B. Imamoglu, M. Ulutas, and G. Ulutas, "A new reversible database watermarking approach with firefly optimization algorithm," *Mathematical Problems in Engineering*, 2017. [Online]. Available: https://doi.org/10.1155/2017/1387375

[57] C. Chang, T. Nguyen, and C. Lin, "A reversible database watermark scheme for textual and numerical datasets," in *IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD*. IEEE, 2021. [Online]. Available: https://doi.org/10.1109/SNPD51163.2021.9704991

[58] C. Iordanou, N. Kourtellis, J. M. Carrascosa, C. Soriente, R. Cuevas, and N. Laoutaris, "Beyond content analysis: detecting targeted ads via distributed counting," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies, CoNEXT*. ACM, 2019. [Online]. Available: https://doi.org/10.1145/3359989.3365428

[59] G. Cormode, S. Maddock, and C. Maple, "Frequency estimation under local differential privacy," *Proc. VLDB Endow.*, 2021. [Online]. Available: http://www.vldb.org/pvldb/vol14/p2046-cormode.pdf

[60] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014. [Online]. Available: https://www.crcpress.com/Introduction-to-Modern-Cryptography-Second-Edition/Katz-Lindell/p/book/9781466570269

[61] D. İşler, E. Cabana, A. Garcia-Recuero, G. Koutrika, and N. Laoutaris, "Freqywm: Frequency watermarking for the new data economy," IMDEA Networks Technical Report, Tech. Rep., 2022.

[62] "Chicago Data Portal," 2022, https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew.

[63] "Adult Dataset," 1996, https://archive.ics.uci.edu/ml/datasets/Adult.

[64] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Rev.*, 2009. [Online]. Available: https://doi.org/10.1137/070710111

[65] D. Goldberg and K. Sastry, *Genetic algorithms: the design of innovation*. Springer, 2007.

[66] A. Kerckhoffs, "A. kerckhoffs, la cryptographie militaire, journal des sciences militaires ix, 38 (1883)," in *Journal des sciences militaires*, 1883.

[67] A. Adelsbach, S. Katzenbeisser, and H. Veith, "Watermarking schemes provably secure against copy and ambiguity attacks," in *ACM workshop on Digital rights management*, 2003. [Online]. Available: https://doi.org/10.1145/947380.947395

[68] S. Behnezhad, "Dynamic algorithms for maximum matching size," in *ACM-SIAM Symposium on Discrete Algorithms, SODA*. SIAM, 2023. [Online]. Available: https://doi.org/10.1137/1.9781611977554.ch6

[69] S. Solomon, "Fully dynamic maximal matching in constant update time," in *IEEE Annual Symposium on Foundations of Computer Science, FOCS*. IEEE Computer Society, 2016. [Online]. Available: https://doi.org/10.1109/FOCS.2016.43

[70] T. Ji, E. Ayday, E. Yilmaz, and P. Li, "Differentially-private fingerprinting of relational databases," *CoRR*, vol. abs/2109.02768, 2021. [Online]. Available: https://arxiv.org/abs/2109.02768

Anexo 4: Understanding the price of data in commercial Data Marketplaces

Understanding the Price of Data in Commercial Data Marketplaces

1st Santiago Andrés Azcoitia
IMDEA Networks Institute
Universidad Carlos III de Madrid
Leganés, Spain
santiago.azcoitia@imdea.org

2nd Costas Iordanou
Cyprus University of Technology
Limassol, Cyprus
kostas.iordanou@cut.ac.cy

3rd Nikolaos Laoutaris
IMDEA Networks Institute
Leganés, Spain
nikolaos.laoutaris@imdea.org

***Abstract*—A large number of Data Marketplaces (DMs) have appeared in the last few years to help owners monetize their data, and data buyers optimize their marketing campaigns, train their ML models, and facilitate other data-driven decision processes. In this paper, we present a first of its kind measurement study of the growing DM ecosystem, focused on understanding which features of data are actually driving their prices in the market. We show that data products listed in commercial DMs may cost from few to hundreds of thousands of US dollars. We analyze the prices of different categories of data and show that products about telecommunications, manufacturing, automotive, and gaming command the highest prices. We also develop classifiers for comparing data products across different DMs, as well as a regression analysis for revealing features that correlate with data product prices of specific categories, such as update rate or history for financial data, and volume and geographical scope for marketing data.**

***Index Terms*—Data economy, data marketplaces, measurement, data pricing**

I. Introduction

Data-driven decision making powered by Machine Learning (ML) algorithms is changing how the society and the economy work and is having a profound positive impact on our daily life. A McKinsey report predicted that data-driven decision-making could reach US$2.5 trillion globally by 2025 [30], whereas a recent market study within the scope of the European Data Strategy estimates a size of 827 billion euro for the EU27 [14]. ML is driving up the demand for data in what has been called the fourth industrial revolution.

To satisfy this demand, several data marketplaces (DMs) have appeared in the last few years. DMs are mediation platforms that aim to connect data providers (acting as sellers) to data consumers (acting as potential buyers), and to manage data transactions between them. This ecosystem includes open data repositories [28], [33], general-purpose [2], [7], [18], [19], [21], and specialized or niche DMs targeting specific industries, such as automotive [13], [50], financial [8], [55], marketing [41], [42], and logistics [65], to name a few.

An issue of paramount importance is that of *data pricing*. Some marketplaces leave it to sellers to set a price for their data products. Many of them do not list prices of their products, but leave it to buyers and sellers to agree on a price after a negotiation. Due to the elusive nature of the traded "commodity", pricing is a very complex matter, even more than in the case of material goods [53]. Unlike oil, to which it is often compared [17], data can be copied / transmitted / processed with close to zero cost. Even the use of the term commodity is a gross oversimplification of what data is. Notice that whereas two liters of gasoline yield a similar mileage on two similar cars under similar driving styles, nothing of this sort applies to data since 1) two datasets of equal volume may carry vastly different amounts of usable information, 2) the same information may have tremendously different value for Service A than for Service B, and 3) even if the per usage value of two services is the same, Service A may use the data 1,000 times more intensely than Service B leading to extremely different produced benefits. Some authors compared data to labor, too [6]. However, unlike labor, data is non-rivalrous meaning that its supply is not affected by its consumption, and thus selling data for a Service A does not prevent a provider from selling (a copy of) the same data for a Service B.

The research community at the intersection between computer science and economics has studied several aspects of data pricing. Still its elusive nature, and the complex business models under which it is made available makes it very hard to prescribe a price for data. Ultimately it is the market that decides and sets prices via complex mechanisms and feedback loops that are hard to capture. Despite some other works trying to measure the price of personal data of individuals [12], [43], [51], there is no systematic measurement study about the price of data products traded in commercial data marketplaces.

Our Contributions: In this paper we present what is, to the best of our knowledge, the first systematic measurement study of marketplaces for B2B data products. This ecosystem, despite being quite vibrant commercially, remains completely unknown to the scientific community. Very basic questions such as "*What is the range of prices of data traded in modern DMs?*", "*Which categories and types of data products command the highest prices?*", "*Which are the features, if any, that correlate with the most expensive data products?*" appear to have no answer and evade most meaningful speculations.

Our research has been supported by MLEDGE project (REGAGE22e00052829516), funded by the Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU/PRTR, and by the European Union's HORIZON project DataBri-X (101070069).

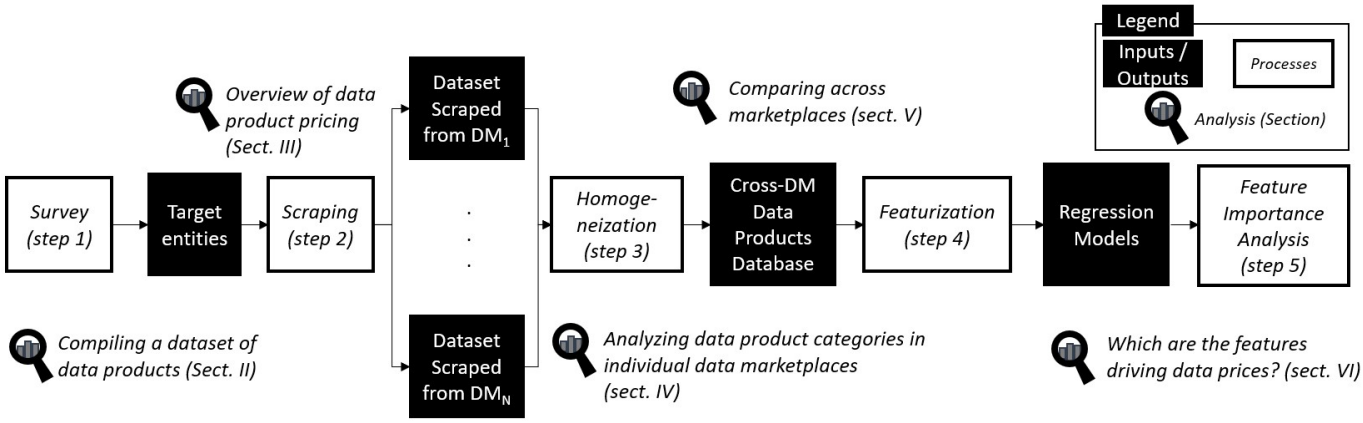


Fig. 1: Summary of our methodology

To answer such questions we followed the methodology summarized in Fig. 1. First, we checked existing surveys for compiling a list of data marketplaces [4], [57], [60]–[62]. We then selected 10 of them that fulfill necessary criteria for a measurement study. For these ones we developed custom crawlers for retrieving information about the products they trade. Using these crawlers, and adding the portfolio of another 30 data providers, we obtained information for more than 210,000 data products and a catalog of more than 2,100 distinct sellers[1]. We also developed data product category classifiers, meaning ML models for identifying products of similar categories across marketplaces, and executed 9 different regression models to understand which features are actually driving their prices.

Our Findings: Analyzing the collected data we observed that the majority of data products were either given for free, or did not carry a fixed price, but rather were up for direct negotiation between the seller and interested buyers. Focusing on the ones that carried a price, some 4,200 of them, we observed that:

• Prices vary in a wide range from few, to several hundreds of thousands of US dollars. The median price for data products sold under a *subscription* model is US$1,400 per month, and US$2,200 for those sold as an *one-off* purchase.

• Using classifiers, we enriched our sample by consistently labeling products according to AWS's categories.

• We found that those related to *telecoms*, *manufacturing*, *automotive* and *gaming* command the highest median prices, and that the most expensive ones relate to *retail and marketing*.

• Using regression models, we managed to fit the prices of commercial products from their features with R^2 above 0.84.

• Due to the heterogeneity of the sample there is no single feature that drives the prices, but instead we spotted meaningful features that drive the prices of specific categories of data. For example, data update rate is a key price driver for *financial* and *healthcare*-related products, whereas geo-spatial localization and the possibility of connecting data points from the same owner are for *marketing* data.

[1]Please, find datasets generated during our research, and code to reproduce our experiments at https://gitlab.com/sandresazcoitia1/data-pricing-tool.

• Overall our models use features related to the category and description of the different data products (i.e., 'Financial', 'Retail', 'stock', 'contact', 'list', etc.), features related to the data products volume and units, as well as singular characteristics extracted from the data products description (i.e., words like 'custom', 'accuracy', 'quality', etc.) to forecast the data product price. Features related to '*what*' and '*how much*' data a product contains are driving 66% of its price.

Like in all measurement studies of Internet-scale phenomena, we will refrain from claiming that any of our findings are "typical" or "representative". What we do claim, however, is that to the best of our knowledge, our measurement study is the first one that attempts to characterize the DM sector, and our above mentioned quantitative results were previously totally unknown. Also, as it will become evident from our methodology later, and to the best of our knowledge, we collected all publicly available pricing information that was accessible during the time of our study.

The remainder of the paper is structured as Fig. 1 shows. First, we frame the scope of our analysis and show some initial outcomes of our measurement study in Sect. II. In Sect. III, we present an analysis on data product pricing in commercial marketplaces. Furthermore, Sect. IV dives deeper into analyzing AWS' DM and DataRade, which account for the largest number of price references in our sample. We then develop tools for enriching our sample and we compare across DMs in Sect. V. Finally, in Sect. VI, we apply several methodologies for analyzing the importance of different metadata features in determining the price of commercial data products.

II. Compiling a dataset of data products

Existing works and surveys on commercial data marketplaces [4], [57], [60]–[62], an extensive web search and a consultation with experts in the area allowed us to compile a list of data marketplaces and understand the different business models they use to compete in this ecosystem. From our analysis, we identified a subset of DMs that fulfilled the criteria for using them as sources of data for a reproducible measurement study. Such criteria include that they grant access to their product catalog without requiring an account, or

through an account but without a vetting process or upfront paid registration, that they have a reasonably large catalog that includes sufficient descriptions of their data products, and that they include a clear description of their pricing policy. Out of the 180 initial DMs, only 10 companies fulfilled all of the above criteria. Most of them did not make it to the list simply because they do not allow non-paying users to browse their catalogs. For example, marketing-related private marketplaces such as *Liveramp*, *LOTAME* or *TheTradeDesk* neither provide public per-product information nor any price references. However, they do provide information about their data partners. By analyzing this information, we did find that 45% of providers in those private marketplaces sell through general-purpose public ones, such as *AWS* or *DataRade*, as well, and hence we have included their products in this study. We also discarded several otherwise *scrapable* general-purpose DMs such as *Data Intelligence Hub* (DIH), *Google Cloud DM* because they included only free data products. We chose to scrape the largest of these free open data marketplaces, *Advaneo*, to help in training our data product category classifiers.

TABLE I: Summary of scraped DMs

Marketplace	#Products	#Paid prod.	#Sellers
Advaneo	198,743	1	N/A
AWS	4,263	2,674	262
DataRade	1,592	1,592	1,262
Snowflake	889	889	200
Knoema	158	158	142
DAWEX	160	160	79
Carto	8,182	5,283	42
Crunchbase	9	9	15
Veracity	115	95	38
Refinitiv	187	187	76
Other providers	777	775	30

Table I lists the 10 DMs that we use as data sources in our study. Overall, we include 6 general-purpose and 4 niche DMs, as well as 30 data providers[2] that, in addition to commercializing their own 777 data products through DMs, provide valuable pricing information on their own websites.

We developed our own web crawler to render and download web pages, and specialized parsers for extracting metadata. We followed common crawling good practices [31]. For example, we avoided visiting several times the same product page in each scraping round and we set up a random wait time from 1 to 2 minutes after requesting a web page in order to avoid flooding the target servers with requests.

We collected information related to 215,075 products from 2,115 distinct sellers in total. We noticed the huge market fragmentation with lots of data providers working with a large number of marketplace platforms. This is natural in a cross-industry nascent market, though hard for data providers to manage. In fact, most data providers (81%) work with only one DM in addition to selling their products through their own web site. 45% of providers in niche financial and marketing-related marketplaces sell through general-purpose DMs, such as *AWS* or *DataRade*, as well. We also spotted DMs advertising and offering their products in other DMs (e.g., *Battlefin* or *CARTO* through *AWS*). Finally, small and niche providers (58% of them) are focusing on one product only.

We scraped all available metadata for data products such as the product id, title, description, source, seller and, when available, its geographic scope, volume, category, use cases, update rate, historic time span, format, etc. We searched for and eliminated duplicates from a single seller within the same DM. We paid special attention to information related to pricing and actual prices of data products.

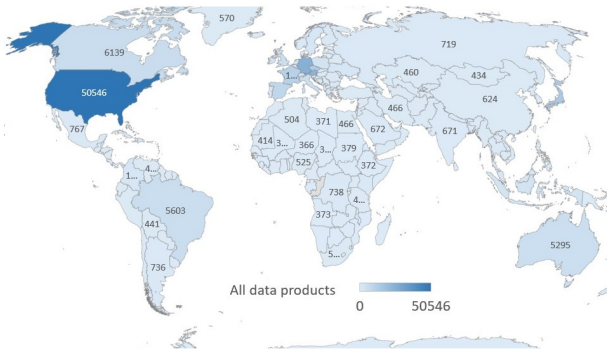


Fig. 2: Data products by country

Regarding the geographical scope of data products, we found that DMs aggregate information from different countries. 14,472 (7%) of the products did not inform about their scope, and 1,177 (around 10% out of the 11,823 paid products) claimed to be global. Figure 2 shows the number of data products covering each country. Regarding the number of *paid* data products, US leads this ranking: around 30% of paid products cover this country. Canada (9.3%), UK (9.2%), Germany (7.6%), France (7.4%), and Spain (7.1%) follow the US in the ranking of countries by number of *paid* products.

III. Overview of Data Product Pricing

It may appear initially surprising that, despite being commercial entities in the B2B space, most of the surveyed and some of the scraped DMs offer predominately free (most of the time open) data. Again we point to the fact that these are privately held companies [2], [21] and not open data NGOs or government initiatives. Our conjecture is that since DMs are two-sided platforms, pre-populating them with free data is a very reasonable bootstrapping strategy, since it can attract the initial "buyers", which in turn will attract commercial sellers and thus help the marketplace grow its revenue.

Next, we focus on the 11,823 paid data products, for which we managed to extract information about their pricing, and whose price is higher than zero. Despite being few compared to the free ones, this sample provides valuable insights about the current status of commercial DMs, as well as to where this segment of the economy is heading to, and how.

[2]42matters, Airbtics, Apptopia, Benzinga, Bizprospex, BoldData, BookYourData, bronID, BuiltWith, DataScouts, Demografy, ebCard, Enigma, ESGAnalytics, HGXN, IFDAQ, ipinfo.io, MultimediaLists, MyDex, OikoLab Weather, Onclusive. Open Corporate, PanXchange, Pipecandy, Shutterstock, Storm Glass, TelephoneListsBiz, Unwrangle, USASalesLeads, and Walklists.

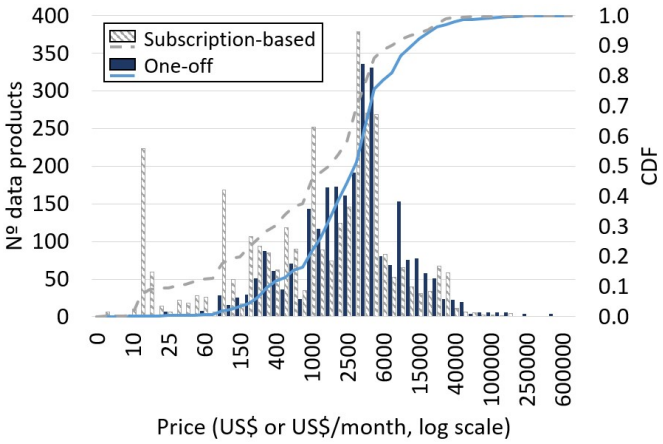


Fig. 3: Histogram and CDF of data products

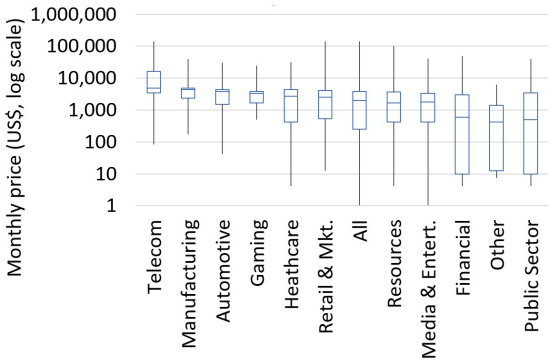
There is a great magnitude of pricing schemes for data products, such as seller-led, buyer-led (bidding), revenue-sharing, tiered-pricing, subject to negotiation, usage-based, etc [4], [44], [53]. Predominant among the 11,823 non-free data products are the *subscription-based* model (i.e., buyer paying for a subscription to get access to data for a period of time), and the *one-off* model (i.e., lump sum payment for data), seller-led in both cases. The first one is used mostly for "live" data usually accessed via an API (e.g., IoT sensor data), whereas the second is used for more static data, which are usually downloaded as one or more files.

4,162 products from 443 distinct providers provided clear information about their prices. Figure 3 shows a histogram and the corresponding CDF of monthly prices for data products. Regarding those offered under a *subscription model*, we see prices across a wide range up to US$150,000 per month. Cheap products below US$100 per month are often curated and cleaner versions of open data. For example, a seller offers a historical compilation of quarterly reports submitted to the US Securities and Exchange Commission (SEC), also downloadable from their websites. They also include low-cost "promotion samples" of more expensive products from well-known sellers, such as GIS data and supporting metadata for a small area of some US cities. The median price is US$1,417 per month. Almost one-third of all products, including targeted market data for example, are sold for US$2-5k monthly.

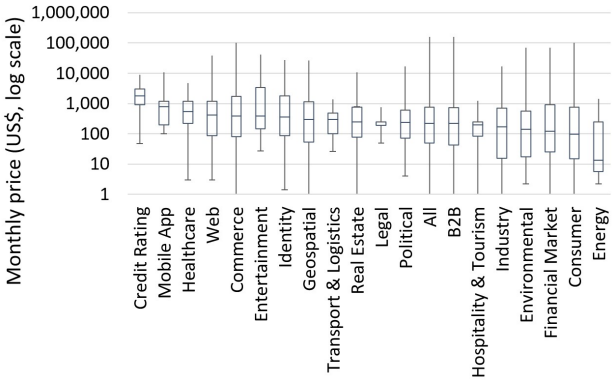
Comparing to products sold under a *one-off model*, (1) the latter tend to be more expensive: median price US$2,176 vs. US$1,417 per month for *subscription-based* products; maximum price US$500,000, more than 3 times higher than the maximum in *subscription-based* access, and (2) *one-off* products have a price histogram more normally distributed around its median at US$2,176. Within the heterogeneous set of products within the US$1,000-4,000 interval, we found a large group of voluminous targeted contact data. Interestingly, we observe a long tail of valuable products in Fig. 3.

IV. Analyzing Data Product Categories in Individual Data Marketplaces

To get a more in-depth understanding of data pricing, we analyzed the catalog of AWS' DM and DataRade, the ones with the largest base of paid products with prices. The former tags data products in 10 different categories, whereas the latter allows data products to be positioned in a hierarchy with more than 300 categories and more than one (out of 150) use cases. Specifically, a product can belong to none, one, or several categories. For instance, credit card transaction data products are classified both as '*Financial*' and '*Retail, Location and Marketing*', whereas those related to weather are not labeled in AWS. We have marked such unclassified products as '*Other*' in our sample.



(a) Subscription prices by industry in AWS.



(b) Subscription prices by category in DataRade.

Fig. 4: Monthly price by data category in AWS and DataRade.

Figure 4 shows box plots of products by first level category in AWS and DataRade. '*Telecom*', '*Manufacturing*' and '*Automotive*' categories exhibit a median price significantly above the global ($\times 2.6$, $\times 2.3$ and $\times 2$, respectively) in AWS, whereas '*Credit Rating*', '*Mobile App*' and '*Healthcare*' data show a higher median price ($\times 8.3$, $\times 3.7$ and $\times 2.5$ above the overall median, respectively) in DataRade. In both cases, the most expensive products are related to marketing ('*B2B*', '*Consumer*' and '*Commerce*' in DataRade).

V. Comparing across marketplaces

Comparing information about data products from different marketplaces is not a straightforward task since i) they provide metadata of different granularity and level of detail, and ii) they use different categorization to describe their products. To overcome these challenges, we developed a methodology to homogenize the categorization of data collected in order to be able to compare similar products across marketplaces.

A. Dealing with different levels of detail

Some marketplaces provide more information than others about their offers. To sort this out, we built a common cross-DM database utilizing a superset of all the different description fields found in different data marketplaces. Apart from their category and text descriptive fields, data product records include the time scope, the volume and units, any potential limitations (e.g., maximum number of users), add-ons, granularity of the information, geo-scope at country level, data delivery methods, update frequency and data format.

We normalized and stored in this cross-DM database all the information from the scraped datasets. We managed to fully automate the extraction of most of the fields (18 out of 27), which were directly scraped from the web pages of the different DMs. This extraction was semi-automated for 5 fields, meaning that they were automatically extracted for certain marketplaces, or retrieved from product descriptions for others, in a process that required a manual check afterwards. For example, *update rate* of data is usually included in the general description of a data product, but the presence of the word '*monthly*' may not necessarily point to a monthly update rate. Information about data volume or data subject units was automatically extracted only for DataRade and BookYourData, and required computer-aided manual typing in the rest of the DMs (we highlight and extract numbers and their context from data descriptions). Manual checks were performed by three different experts. Any ambiguities and disagreements were resolved by majority voting.

B. Dealing with different categorization systems

In Sect. IV we showed that every marketplace has its own way to classify data. Furthermore, boundaries between tags are often blurry, and the criteria followed by different DMs to label a data product with a certain category tag are not necessarily coherent. For example, only certain marketplaces mark '*credit card transaction*' data products as '*financial*', whereas all DMs label them as related to '*marketing*'. Thus, even if we find apparently comparable categories across different marketplaces, we may miss relevant data products due to inconsistencies in their categorization processes.

We addressed this issue by developing a series of natural language processing naïve Bayes (NB) classifiers [20], [22], [39]. In our first attempt, we wanted to identify similar data products – those that belong to the same category – between two different (source and destination) DMs. As a result, we trained both multinomial and complement versions of NB classifiers to detect data products from the source DM that belong in a certain category by using feature vectors based on the information provided by the data product description from the source DM. We used bag of words [36] and data preprocessing steps such as removing stop words and words with numbers, using stemming and TF-IDF transformation [47], [56]. Then we validated the resulting classifier against a manually labeled sample from the destination DM. Manual labeling was performed by three different experts. Any ambiguities and disagreements were resolved again by majority voting.

We utilized the above methodology to build different classifiers to help us compare data products between the two DMs including more price references, namely DataRade (destination DM) and AWS (source DM). We generated our feature vectors based on AWS data product descriptions (source DM) and applied the resulting classifiers to DataRade data products (destination DM). We were interested in finding out: (1) what percentage of products from those categories could we identify in DataRade, (2) whether categorization and pricing were coherent between them, and (3) whether we could enrich our metadata by adding AWS's inferred categories to all products.

We utilized our cross-DM database to generate the train/test datasets at 80/20 split in order to train and test the corresponding classifiers. We observed that multinomial classifiers outperformed the complement NB for this task so we proceeded with the former ones. The resulting classifiers yield an acceptable F_1 score above 0.85 (average for 50 executions with different random 80/20 train/test splits). In fact, they identified meaningful and reasonable stems when tagging products related to each category. For example, for the two categories including more data products:

Financial: 'system', 'sec', 'exchang', 'type', 'file', 'form', 'edgar', 'secur', 'act', and 'compani'.

Retail, Location and Marketing: 'locat', 'topic', 'b2b', 'score', 'echo', 'trial', 'compani', 'visit', 'intent', 'consum'.

We then validated the models against a manually labeled sample from DataRade. Manual labeling was performed by three different experts. Any ambiguities and disagreements were resolved again by majority voting. The validation set included 745 manually pre-labeled with both '*Financial*' and '*Retail, Location and Marketing*' tags. The models trained only with data from AWS did not perform so well on the validation set (F_1 scores of 0.73 and 0.43 for '*Financial*' and '*Retail, Location and Marketing*' data). To generalize further our methodology and improve its accuracy, we enriched the train data with information from other DMs. In particular:
(1) The ***Financial*** classifier was trained with 95,208 labeled descriptions of products from 4 different entities (Advaneo, Carto, AWS, and Refinitiv), and 45,298 financial products.
(2) The ***Retail, Location and Marketing*** classifier was trained with 3,828 descriptions from 3 entities (AWS, BookYourData and TelephoneLists), including 1,614 marketing products.

By adding products belonging to the same category from other DMs we observed better balance between precision and recall and an overall improvement of model generalization. We also observed an increase of the F_1 score in the test set. Particularly, adding information from Refinitiv improves the

F_1 score from 0.73 to 0.79. In the case of '*Retail, Location and Marketing*', adding information from specialized marketing DMs (e.g., BookYourData), drastically improves the F_1 score from 0.43 to 0.74. We tested multiple classifiers, with and without stemming, and we found that using word-based instead of stem-based features led in general to more accurate results in both cases (+5% F_1 score). Table II shows the accuracy obtained by both classifiers.

TABLE II: Score of data product classifiers

	Accuracy	**Precision**	**Recall**	**F_1 Score**
Test - Financial	0.93	0.97	0.81	0.88
Test - Retail	0.95	0.96	0.88	0.91
Val. - Financial	0.89	0.72	0.88	0.79
Val. - Retail	0.78	0.81	0.68	0.74

We used them to label data products in DataRade, and we located 619 and 701 '*Financial*' and '*Retail, Location and Marketing*' data products, which represent 39% and 44% of the total sample, respectively. As happened in AWS, not only do those categories contain the largest number of products in DataRade, but the most expensive ones are tagged as '*Retail, Location and Marketing*', as well.

We repeated the process for the rest of the 11 AWS data categories, and this way we managed to enrich our sample by homogeneously labeling products based on their descriptions. The four categories with highest median prices are the same as in AWS, but in a different order. Again, most products belong to '*Financial*' and '*Retail, Location and Marketing*', and the most expensive ones belong to the latter category.

Does this methodology work if we switch source and destination DMs? In order to answer this question, we trained NB classifiers to detect products in AWS related to relevant use cases and categories in DataRade. In this case, DataRade acted as the source DM, i.e., it provided descriptions and tagging information to train the classifiers, whereas AWS' role was the destination DM, whose products we labeled with some of DataRade's tags and driven by the criteria we learned from the source DM. In particular, we focused on products belonging to the '*B2B Marketing*', '*Audience Targeting*' and '*Risk Management*' use cases in DataRade, some 46, 48 and 30 products out of 745 respectively. Since the training set is imbalanced and the number of samples is low, complement NB outperformed multinomial NB in this case. We trained the classifiers and obtained the log-probability of belonging in each category for all the data products in AWS. As a result, at least 16 out of the top 20 data products showing the highest log-probability turned out to be useful for those specific use cases, according to the assessment of three different experts.

VI. Which are the features driving data prices?

So far we have seen an overview of data pricing, looked at the prices of particular categories, developed and applied a methodology to homogeneously label products across marketplaces in our sample. Our final goal is to understand the prices of data in commercial data marketplaces.

For that purpose, we first extract features to train regression models for predicting the prices of real commercial data products. We do not intend to build state-of-the-art price predictors, but rather to understand which features are driving the price of data. Therefore, we conduct feature importance analysis on the resulting regression models and we find out which features have the highest impact on the observed prices for the different data products in our corpus.

A. Building a feature matrix to feed regression models

An additional preprocessing step is needed in order to transform the fields of our cross-DM database into a set of valuable features that can be ingested by ML regression algorithms. This process uses the NLTK [9] and Scikit-learn [52] Python libraries and includes mainly the following steps:

1) Extraction of '*word*' features from the title and the textual description of each data product. We use bag of words [36] and data preprocessing steps such as removing stop words and words with numbers, TF-IDF transformation [56], and stemming [47]. In addition, we have sellers' names removed from the vocabulary, so as to avoid bias introduced by knowing their identity. Finally, we prepare matrices for different vocabulary lengths and optimize each algorithm for this parameter.
2) Breakdown of volume-related fields in 13 different groups depending on their nature. For example, we separate data products targeting 'entities' or 'companies', from those whose subjects are 'individuals' in different features. The resulting comparable units are in turn normalized, and a new overarching feature ('units') measuring the percentage of units covered is added to compare products across groups of units.
3) Calculation of country-level binary features to indicate whether a certain country is covered by a data product.
4) Homogenization of the units of time when measuring the time scope of the products, what we will call *history*.

Before feeding the models, we reduce the number of input features by discarding those that have a unique value, which may appear when filtering the complete dataset by *category*. Next, we unify groups of features showing a high cross-correlation among them, i.e., $R^2 \geq 0.9$.

As a result of this *featurization* process, we reduce each sample product to a feature vector and produce a feature matrix to train our regression models. Table III lists feature groups and some examples of their individual features. We organize features in 10 disjoint sets according to their nature and the basic questions they answer about data products.

We evaluated the linear correlation of individual features with respect to data product prices. Not surprisingly, it turns out that none of them is linearly correlated to price, as opposed to what we found for specific sellers. Our challenge now is measuring which features and groups of features are more significant in determining the price of data products in commercial marketplaces.

TABLE III: List of feature groups

Question	Group	Definition	Nº features	Example of features
What?	**Category**	Labels attached to the product that define the type of data it contains	11	'Weather', 'Gaming', 'Financial'
	Description	Stem-like features obtained from data product descriptions	up to 2000	'wordmarket', 'wordidentifi', 'wordlist'
	Identifiability	Tells whether the product allows the buyer to recognize the activity of individuals or to identify specific companies	2	'idSessions', 'IdCompanies'
How much?	**Volume**	Normalized nº units covered broken down by the nature of such units	14	'units', 'people', 'entities'
	Update rate	Defines the frequency between data updates as announced by the seller	11	'real time', 'monthly', 'hourly'
How?	**Delivery method**	Defines how the buyer can have access to data	8	'S3Bucket', 'Download', 'FeedAPI'
	Format	Defines the way in which data is arranged	17	'txt', 'shapefile', 'xls'
	Add-ons	Tells whether the product attaches any add-on or has any limitations	2	'ProfServices', 'Limitations'
When?	**History**	Time scope included	1	'History'
Where?	**Geo scope**	Metrics about countries included in the data product	up to 249	'Nº Countries', 'USA', 'Canada'

TABLE IV: Accuracy achieved by regression models

Model	Financial			Marketing			Healthcare			All		
	R^2	MAE	MSE	R^2	MAE	MSE	R^2	MAE	MSE	R^2	MAE	MSE
RF	0.85	0.2	0.14	0.86	0.21	0.13	0.78	0.25	0.15	0.84	0.23	0.16
kN	0.78	0.31	0.26	0.74	0.33	0.24	0.77	0.26	0.17	0.69	0.37	0.31
GB	0.82	0.23	0.16	0.8	0.28	0.19	0.73	0.27	0.19	0.79	0.3	0.22
DNN	0.73	0.33	0.35	0.77	0.30	0.22	0.68	0.26	0.18	0.72	0.33	0.28

TABLE V: Top 10 most relevant features not related to volume by category and regression model

Financial			Marketing			Healthcare		
RF	**kNN**	**GBR**	**RF**	**kNN**	**GBR**	**RF**	**kNN**	**GBR**
S3Bucket	Email	S3Bucket	IdSessions	History	csv	wordhealth	csv	wordlist
wordsubmit	Download	wordmonthli	Download	USA	yearly	wordtrend	daily	Del. Methods
Download	daily	wordstock	REST API	IdSessions	REST API	wordmedic	wordmarket	wordhospit
txt	IdCompanies	worddeliv	wordcustom	Nº Countries	wordqualiti	wordglobal	wordgo	wordidentifi
wordedgar	USA	Del. Methods	USA	Financial	wordaccur	csv	Limitations	wordamerica
wordcustom	wordmarket	txt	yearly	Others	wordidentifi	Del. Methods	location data	wordhealth
wordlist	Retail	wordneed	monthly	wordcontact	wordwebsit	wordinsight	wordpopul	wordreport
wordcontact	wordcontact	wordsubmit	IdCompanies	Email	UI Export	wordreport	wordprofil	wordstudi
wordsystem	real time	wordreport	wordname	UI Export	wordcover	wordregion	wordinsight	wordupdat
wordcompar	wordprice	wordcontact	location data	Download	wordfield	wordlist	Download	wordcontact

B. Analyzing feature importance

Regression models can be used for feature importance analysis. Next we use a range of such techniques to understand which features have the higher impact on data product prices.

1) Optimizing Regression models: Owing to their stochastic nature, training several regression algorithms and comparing their outcomes is key to obtaining robust conclusions. Consequently, we have tested variations of 9 different regressors with different values for their main parameters (e.g., num. of estimators, depth, etc.) as included in the Scikit-learn [52] Python library, and inputs of different vocabulary lengths. Such models work with the log instead of the absolute value of product prices as the dependent variable so as to normalize the distribution of prices and avoid negative price predictions. We were hoping to find at least 3 models that produce sufficiently accurate price predictions, measured as the R^2score of their output w.r.t. actual prices.

To reduce the complexity of each model, we removed low-value features, i.e., those that had a negative leave-one-out (LOO) value, provided the accuracy of the model was not negatively affected. A feature having negative LOO value means that the model improved its average accuracy in 10 random executions for different train and test data splits when such feature was removed from the input matrix. Finally, we performed a cross-validation to check the variance of the accuracy of the model when training and testing in 5-folds, and 20-random training-test splits of the input data.

We found that three target models worked reasonably well (i.e., they yield an R^2 score greater or equal to 0.70), namely Random Forest [10], k-Nearest Neighbours [38], and Gradient Boosting [23], [46] regression models. On the contrary, we discarded linear, Elastic-Net [68], Ridge [32], Bayesian Ridge [45], and Lasso [64] regressions even though they worked well in specific simulations.

In addition, we also tested a Deep Neural Network regressor using the TensorFlow [1] and Keras [34] libraries. We followed all common good practices recommended for such activity by first standardizing the input data. We tested RELU/Leaky RELU activation functions for all hidden layers, and a linear activation function for the output layer. As loss function we used the mean absolute error (MAE). To avoid overfitting we randomly applied Drop-out between training epochs and to avoid dying/exploding neurons we also applied Batch normalization between all layers. We used the Adam optimizer [35] with a tuned learning rate decay to train the

model faster at the beginning and then decrease the learning rate with further epochs to make training more precise. Finally, we used Callbacks to stop the training at the optimal epoch.

Table IV presents a summary of the accuracy obtained by regressor and category of data products, including the R^2 score, the MAE and the mean squared error (MSE) with regards to the actual log prices. For the sake of robustness, our results were consistent across subsequent 5-fold and 20 random train/test splits: R^2 score showed a standard deviation below 4% of the average in each round. Note that due to the total (low) number of observations that we have in our datasets, DNN models are not recommended, nevertheless, we wanted to explore them since we believe that they will further improve our results as soon as we manage to increase the overall size of our datasets. Consequently, we avoided using any DNN model in the feature importance analysis.

2) Analyzing the importance of individual features: We carried out this process for financial, marketing, healthcare and all data products in our sample. Financial and marketing data were the most popular data categories, whereas healthcare data was chosen as a relevant disjoint category of less though increasingly popular products showing a different behavior in terms of prices. As a result, we obtained at least one model that achieves a R^2 score of 0.78 by category and accurately fits the prices of data products (see Tab. IV). We ran two different individual feature importance analysis:

1) measuring the accuracy lost by randomly shuffling the values of a certain feature among samples (permutation importance analysis [63]), and
2) measuring the prediction accuracy lost when one individual feature is removed from the inputs (leave-one-out or LOO value).

We have found that 50% of the positive LOO and 67% of the ΔR^2 score by shuffling values owe to the top 10 most relevant features on average for specific categories of data. Note that we would need more than 25 features to achieve equivalent scores if we include all the products. Whereas features related to units and the volume of data clearly lead the ranking for financial and marketing data products, they are less important for healthcare-related ones.

We cross-validated our results in 5-fold executions of both methods and took averages in order to disregard features that showed to be important only in specific tests. As regards robustness, we compared the top-20 ranking of every individual test to the top-20 average ranking of that algorithm and category. It turns out that both rankings have at least 5 features in common in 95% of the cases, and a median of 13 common individual features.

Table V lists other features not related to data volume in descending order of importance. Next we provide some details about the most important features of each specific category:

Financial: Not only do volume-related features such as '*units*' and '*entities*' rank number one, but they are on average four times more important than the second feature in the ranking. Other features relate to specific characteristics of financial data products and help models identify data products either by their category (e.g., '*Retail*') or their description. For instance, RF relies on the word '*edgar*', which stands for SEC's Electronic Data Gathering, Analysis, and Retrieval System, all algorithms identify business '*contact*' lists, a family of financial products, and they also use '*stock*' and '*market*'. The word '*custom*' helps identify information about *customers*, but also refers to the valuable possibility of personalizing data products (e.g., select which companies we want financial data from). Features related to delivery methods (e.g., '*S3bucket*' or '*Download*') and update rate (e.g., '*real time*' or '*daily*') stand out in terms of relevance, as well.

Marketing: With regards to marketing data products, features related to volume, such as '*units*' and '*entities*' lead the ranking, as well. Again categories (e.g., '*Financial*', '*Others*') and specific words pointing to relevant characteristics of data play a relevant role, too. For example, words like '*contact*' are used to locate contact lists, a family of marketing products, the stems '*qualiti*' and '*accur*' refer to the high-quality and accuracy of data, as advertised by sellers. A number of features, such as the stem '*identifi*', emphasize the value of identification for marketing data. In addition, the presence of '*IdSessions*' and '*IdCompanies*' features indicates that being able to reconstruct sessions of anonymized individuals and being able to identify merchants are price drivers for marketing products. Unlike financial data, the fact that a dataset includes *'location data'* is also used to set prices of marketing data. Finally, the scope of data is important, as suggested by features like '*USA*' and '*Nº Countries*' ranking high in the results of RF and kNN models.

Healthcare: The '*what*' is more important than the '*how much*' when fitting the observed prices of healthcare products. This is due to the heterogeneity of data products belonging in this category, ranging from contact lists of healthcare practitioners and hospitals to data about clinical trials or specific medications. Therefore, stems like '*trial*', '*hospit*' or '*studies*' help in identifying what a dataset is about. The stem '*go*' refers to an official check-in and rating system that was used to limit the spread of COVID in the US. Features related to the update rate, data format ('*csv*'), the number of available delivery options ('*Del. Methods*') and the presence of '*Limitations*' (e.g., limited number of reports, or limited data exports included) determine product prices, too.

3) Analyzing the importance of groups of features: Since LOO is often negligible for individual features, we have repeated this analysis for groups of features answering to the same question regarding the data product (see Tab. III). In this case, we have used the following two methods:

1) Measuring the prediction accuracy lost when a group of features is removed from the input dataset (LOO).
2) Measuring the average (in 20 random train/test split executions) Shapley value of each group of features.

The Shapley value is defined as the average R^2 score added by combining the information of a certain group of features with every possible mix of the rest of groups. This is a well-known and widely-used concept in game theory, economics and ML [24], [59], and it is considered a 'fair' method to

TABLE VI: LOO values by feature group

Group	Financial			Marketing			Healthcare			All		
	RF	kNN	GBR	RF	kNN	GBR	RF	kNN	GBR	RF	kNN	GBR
Description	0.027	0.025	0.066	0.021	0.034	0.098	0.054	0.425	0.052	0.023	-0.020	0.079
Volume	0.092	0.182	0.167	0.171	0.138	0.199	0.048	0.014	0.052	0.138	0.123	0.142
Geo Scope	-0.005	-0.007	-0.001	-0.003	-0.006	0.000	0.015	0.000	-0.011	-0.003	-0.002	0.000
Del. Method	0.005	0.032	0.011	0.000	0.018	0.008	0.019	0.017	0.003	0.002	0.010	0.008
Format	0.002	0.004	0.010	0.007	0.001	0.023	0.007	0.030	0.000	0.002	0.007	0.006
Category	-0.002	0.001	0.001	-0.001	-0.003	0.001	0.013	-0.033	-0.006	0.001	0.000	0.003
Add-ons	-0.001	0.007	-0.001	-0.001	0.000	0.001	0.000	0.022	0.000	0.001	0.001	0.000
Identifiability	-0.002	0.016	0.002	-0.001	0.006	0.004	0.010	0.000	-0.009	0.000	0.008	0.000
History	-0.001	0.000	0.000	-0.003	0.004	0.000	0.009	0.000	0.000	0.002	0.000	-0.001
Update Rate	0.001	0.023	0.001	0.036	0.000	0.016	0.010	0.021	0.000	0.021	-0.002	0.014

TABLE VII: Shapley values by feature group

Group	Financial			Marketing			Healthcare			All		
	RF	kNN	GBR	RF	kNN	GBR	RF	kNN	GBR	RF	kNN	GBR
Description	0.155	0.266	0.222	0.247	0.153	0.152	0.232	0.290	0.236	0.113	0.176	0.187
Volume	0.211	0.216	0.184	0.290	0.241	0.241	0.168	0.125	0.131	0.211	0.210	0.174
Format	0.087	0.006	0.086	0.027	0.046	0.094	0.090	0.077	0.082	0.072	0.087	0.071
History	0.072	0.000	0.059	0.009	0.037	0.036	0.063	0.001	0.046	0.058	0.010	0.037
Update Rate	0.088	0.056	0.084	0.060	0.032	0.050	0.046	0.145	0.041	0.067	0.034	0.067
Del. Method	0.036	0.054	0.044	0.093	0.075	0.049	0.030	0.040	0.035	0.062	0.062	0.074
Identifiability	0.034	0.038	0.028	0.052	0.027	0.048	0.040	0.001	0.031	0.056	0.022	0.039
Geo Scope	0.056	0.046	0.050	0.032	0.044	0.036	0.030	0.001	0.040	0.061	0.015	0.024
Category	0.071	0.021	0.044	0.018	0.043	0.037	0.017	0.031	0.039	0.070	0.063	0.055
Add-ons	0.021	0.003	0.021	0.012	0.028	0.038	0.048	0.053	0.041	0.055	0.026	0.045

distribute the gains obtained by cooperation. In our case, we applied the Shapley value to distribute the gains in accuracy of our regression models among the groups of features that contributed to achieving such an accuracy. Furthermore, we ran 5-fold feature importance analysis in the case of LOO, in a similar way as we did for individual features, and 20 calculations of the Shapley values for random 80/20 train/test splits of our input data.

Whereas LOO measures gains or loses in accuracy of a model when features belonging in a group are removed from the input matrix, Shapley values better capture the complementarity among groups and take into consideration their individual predictive power, as well. Table VI and Table VII list the LOO and the Shapley values by group of features in descending order of importance. The standard deviation of Shapley values across executions is acceptable (average below 0.029 for financial and marketing datasets, 0.057 for healthcare-related data, and below 0.017 for all the data), and the ranking of relevant feature groups remains stable.

Figure 5 plots the percentage of the sum of Shapley and LOO values that each feature group represents, what we call their *predictive power*, and illustrates how important each group is for determining the prices of each category of products. We have piled together and colored in gradients groups responding to the same question about data products.

Note that the algorithms, in the absence of certain features, try to replace or infer them through other features in order to come up with the best estimation possible. We have observed that this happens with '**category**' labels or '**add-ons**', and it is also the reason why LOO values are generally smaller than the corresponding Shapley values.

By looking at Fig. 5, we can confirm that features related to '**volume**' and '**descriptions**' are the most relevant groups driving data prices: at least half of the predictive power owes to those two groups of features according to their Shapley

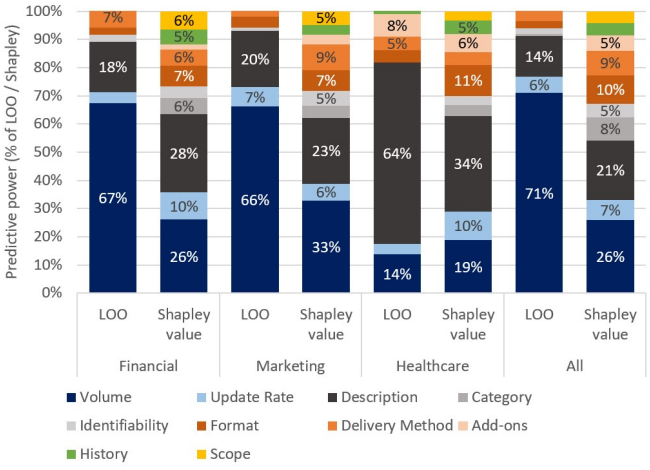


Fig. 5: Predicting power of feature groups

values. While '*volume*' is clearly the most relevant group for marketing data products, it is not so relevant for healthcare-related data due to the heterogeneity of products belonging in this category and due to their lower price-sensitivity to volume.

Data '**update rate**' and its '**format**' are consistently relevant across all data categories, but to a lesser extent (6-11% of the prediction score), whereas the Shapley values of the other groups differ across categories: '**history**' (meaning the time span of data delivered) is more relevant for financial and healthcare-related data, '**delivery methods**' are more relevant for marketing data, and ´**identifiability**' is important in general, but especially for marketing products. These results are in line with our discussion based on the relevance of individual features in the previous section.

In summary, it is mostly '*what*', as captured in product description and categories, and '*how much*' data is being traded that determine the price of a product. Since relevant descriptive features are diverse and strongly differ across

data categories, we failed to find a single feature other than '*units*' that, with some aforementioned exceptions, consistently shows a significant *predictive power*. However, we did find interesting features driving the prices of specific categories of data, such as update rate for financial products, and the ability to provide exact locations and those related to identifiability for marketing data. '*How*' data is delivered to buyers proved to be important too, and accounts for 15-24% of *predictive power* according to Shapley. Finally, historical time span ('*when*') and geographical scope ('*where*') of data products, whose score oscillates around 5% for every data category, are less relevant in driving their prices.

VII. Related works

Even though several surveys related to data marketplaces have been recently published [4], [57], [60]–[62], our work is, to the best of our knowledge, the first empirical measurement study that deals with the prices of data products sold in commercial data marketplaces.

In fact, the lack of empirical data around dataset prices is considered as a key challenge in data pricing research [53]. According to some authors, some techniques to set the prices of digital products [58] or cloud services [66] are applicable to data products, as well. Some authors proposed auction designs to set the prices of digital goods and data products [26], [27]. Novel AI/ML data marketplace architectures have been proposed under the concept of value-based pricing [3], [16], [49] and the value of privacy [48]. Moreover, some authors defined pricing strategies and marketplaces based on differential privacy [25], [40] or queries to a database [15], [37]. All of them work on analyzing the theoretical properties for fair, arbitrage-free pricing, but leave the responsibility of actually defining absolute prices to both buyers and sellers. Quality-based pricing [29] is the one closest to our approach. According to it, the value of data must be assessed by evaluating and assigning weights to certain quality features. Even though some additional works have provided data pricing strategies for sellers based on this idea [67], we are not aware of any measurement study that has been able to derive weights for such features from real market data.

The pricing of personal data of individuals has received attention from the privacy and measurement communities. There are measurement studies based on prices carried over the Real Time Bidding protocol [43], [51] as well as more traditional survey-based studies [12]. These works report prices for the data and the attention of individuals and, therefore, have nothing to do with B2B datasets traded in modern DMs.

Cross-marketplace analysis and discoverability of data has been pointed out as a significant challenge by data marketplace vision papers [54]. Google Dataset Search has proposed a standard for providing metadata for their crawlers [11]. Discoverability is the *leit motiv* of DMs and data aggregators, such as DataRade, but do not touch upon pricing questions.

Finally, part of this work explaining the challenges in scraping and comparing across data marketplaces and outlining the design of a data quotation tool was published as a workshop paper [5]. This paper is adding substantially new material, such as the procedures we used to populate a cross-DM database, the development and test of classifiers to compare across DMs (see Sect. V), and the training of regression models to fit data prices and carry out feature importance analysis (see Sect. VI).

VIII. Conclusions and future work

Our work has provided a first glimpse into the growing market for B2B data. Despite having worked in a range of pricing topics in the past, prior to conducting this study, we did not have the slightest idea even for fundamental questions such as "What are typical prices for data products sold online?", or "What types of data command higher prices?". Our work has produced answers to those and many other questions. We have seen that while the median price for data is few thousands, there exist data products that sell for hundreds of thousands of dollars. We have also looked at the categories of data and the specific per-category features that have the highest impact on prices. Having scraped metadata for hundreds of thousands of data products listed by 10 real-world data marketplaces and other 30 data providers we found fewer than ten thousand that were non-free and included prices. We believe that this is due to prices being often left to direct negotiation between buyers and sellers, and also because most marketplaces use free data to bootstrap their marketplace and attract the first "buyers" and then commercial sellers.

Moreover, the paper represents a first step towards developing a price recommendation tool for new data products [5], and has even provided a first implementation of some of its key components, namely i) the metadata and taxonomy required to describe data products, ii) crawlers and parsers to automate the collection of such information from key leading marketplaces, iii) classifiers to compare across them, and iv) regression models to understand which are the most relevant features driving product prices. The significant monthly growth rate we have seen at AWS and other marketplaces makes us believe that in the future the paid catalog of data marketplaces is bound to grow and therefore, we will continue monitoring them to see how they evolve.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Tal- war, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[2] Advaneo. Access to the world of data. https://www.advaneo-datamarketplace.de/. Last accessed: Oct'22

[3] A. Agarwal, M. Dahleh, and T. Sarkar. A Marketplace for Data: An Algorithmic Solution. In Proc. of ACM EC, 2019.

[4] S. Andrés Azcoitia and N. Laoutaris. A Survey of Data Marketplaces and their Business Models. SIGMOD Record, 2022.

[5] S. Andrés Azcoitia, C. Iordanou, N. Laoutaris, "Measuring the Price of Data in Commercial Data Marketplaces," ACM Data Economy Workshop, 2022.

[6] I. Arrieta-Ibarra, L. Goff, D. Jiménez-Hernández, J. Lanier, and E. G. Weyl. Should we Treat Data as Labor? Moving Beyond "Free". AEA Papers and Proceedings, 108:38–42, 2018

[7] AWS. Amazon Web Services Marketplace. https://aws.amazon.com/marketplace. Last accessed: Oct'22

[8] Battlefin. Better your investments using alternative data. https://www.battlefin.com/. Last accessed: Oct'22.

[9] E. L. Bird, Steven and E. Klein. Natural Language Processing with Python. O'Reilly Media Inc, 2009.

[10] Breiman. Random forests. Mach. Learn., 45(1):5–32, Oct. 2001.

[11] D. Brickley, M. Burgess, and N. Noy. Google Dataset Search: Building a Search Engine for Datasets in an Open Web Ecosystem. In Proc. of ACM WWW conf., 2019.

[12] J. P. Carrascal, C. Riederer, V. Erramilli, M. Cherubini, and R. de Oliveira. Your Browsing Behavior for a Big Mac: Economics of Personal Information Online. In Proc. of ACM WWW Conf., 2013

[13] Caruso. Your solution. one platform. multibrand in-vehicle data. https://www.caruso-dataplace.com/. Last accessed: Oct'22.

[14] G. Cattaneo, G. Micheletti, and al. The European Data Market Monitoring Tool. Key Facts and Figures, First Policy Conclusions, Data Landscape and Quantified Stories. Final Study Report. European Commission, 2020.

[15] S. Chawla, S. Deep, P. Koutris, and Y. Teng. Revenue Maximization for Query Pricing. Proc. of the VLDB Endow., 13, 2019.

[16] L. Chen, P. Koutris, and A. Kumar. Towards Model-Based Pricing for Machine Learning in a Data Marketplace. In Proceeding of ACM SIGMOD, 2019.

[17] Clive Humby. Data is the New Oil! Keynote at ANA Senior Marketer's Summit, Kellogg School, 2006.

[18] DataRade. Datarade. choose the right data with confidence. https://datarade.ai/. Last accessed: Oct'22.

[19] Dawex. DAWEX Data Exchange, unleash the value of your data. https://www.dawex.com/. Last accessed: Oct'22.

[20] L. Denoyer and P. Gallinari. Bayesian Network Model for Semi-structured Document Classification. Inf. Process. Manage., 40(5), 2004.

[21] DIH. Data intelligence hub. extract value from data securely. https://dih.telekom.net/. Last accessed: Oct'22.

[22] P. Domingos and M. Pazzani. On the optimality of the Simple Bayesian Classifier under Zero-one Loss. Mach. Learn., 1997

[23] J. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics, 29, 2000

[24] A. Ghorbani and J. Zou. Data shapley: Equitable Valuation of Data for Machine Learning. Proc. of the ICML, 2019.

[25] A. Ghosh and A. Roth. Selling privacy at auction. In Proc. of the ACM EC '11, 2011.

[26] A. V. Goldberg and J. D. Hartline. Competitiveness via Consensus. In Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '03, page 215–222, USA, 2003. Society for Industrial and Applied Mathematics.

[27] A. V. Goldberg, J. D. Hartline, and A. Wright. Competitive Auctions and Digital Goods. In Proc. of the ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, 2001.

[28] Harvard. Dataverse. https://dataverse.harvard.edu/. Accessed: Oct'22.

[29] J. R. Heckman, E. Boehmer, E. H. Peters, M. Davaloo, and N. G. Kurup. A Pricing Model for Data Markets. In Proc. iConference 2015.

[30] N. Henke, J. Bughin, and al. The age of analytics: Competing in a Data-driven World. McKinsey Global Institute, 2016.

[31] M. Hils, D. W. Woods, and R. Böhme. Measuring the Emergence of Consent Management on the Web. In Proc. of the ACM IMC'20, page 317–332. 2020.

[32] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased Estimation for Nonorthogonal Problems. Technometrics, 42(1):80–86, Feb. 2000.

[33] Kaggle. Datasets. https://www.kaggle.com/datasets. Accessed: Oct'22.

[34] Keras. Simple. flexible. powerful. https://keras.io/. Accessed: Jun '22.

[35] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, Proc of ICLR '15, 2015.

[36] Y. Ko. A Study of Term Weighting Schemes using Class Information for Text Classification. In Proc. of ACM SIGIR 2012.

[37] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. Querymarket Demonstration: Pricing for Online Data Markets. Proc. of the VLDB Endow., 5, 2012.

[38] O. Kramer. Unsupervised k-Nearest Neighbor regression. 2011.

[39] G. Krishnaveni and T. Sudha. Naïve Bayes Text Classification - a Comparison of Event Models. Imperial Journal of Interdisciplinary Research, 3, 2016.

[40] C. Li, D. Y. Li, G. Miklau, and D. Suciu. A theory of pricing private data. ACM Transactions on Database Systems 39(4), 2015.

[41] LiveRamp. Data marketplace. https://liveramp.com/our-platform/data-marketplace/. Last accessed: Oct'22.

[42] LOTAME. Private data exchange (pdx). trusted data relationships made easy. https://www.lotame.com/pdx/. Last accessed: Oct'22.

[43] C. C. Lukasz Olejnik, Minh-Dung Tran. Selling off privacy at auction. In Proc. of the NDSS Symposium, 2014.

[44] A. Löser, F. Stahl, A. Muschalle, and G. Vossen. Pricing Approaches for Data Markets. In Proc. of the International Workshop on Business Intelligence for the Real-Time Enterprise, 2012.

[45] D. J. C. MacKay. Bayesian interpolation. Neural Comput., 4(3), 1992.

[46] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting Algorithms as Gradient Descent. In Proc. of the International Conference on Neural Information Processing Systems, 1999.

[47] S. Matic, C. Iordanou, G. Smaragdakis, and N. Laoutaris. Identifying Sensitive Urls at Web-scale. In Proceedings of the ACM IMC, 2020.

[48] C. Niu, Z. Zheng, F. Wu, S. Tang, X. Gao, and G. Chen. Unlocking the Value of Privacy: Trading Aggregate Statistics over Private Correlated Data. In Proc. of ACM SIGKDD, 2018.

[49] O. Ohrimenko, S. Tople, and S. Tschiatschek. Collaborative Machine Learning Markets with Data-replication-robust Payments. CoRR, 2019.

[50] Otonomo. One-stop shop for vehicle data. https://otonomo.io/. Last accessed: Oct'22.

[51] P. Papadopoulos, N. Kourtellis, P. R. Rodriguez, and N. Laoutaris. If you are not paying for it, you are the product: How much do advertisers pay to reach you? In Proc. of the ACM IMC, 2017.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. Journal of Machine Learning Research.

[53] J. Pei. Data Pricing – from Economics to Data Science. In Proc. of the ACM SIGKDD, page 3553–3554, 2020.

[54] M. F. Raul Castro Fernandez, Pranav Subramaniam. Data Market Platforms: Trading Data Assets to Solve Data Problems. In Proc. of the VLDB Endow., 2020.

[55] Refinitiv. Data catalog. our data, your way. https://www.refinitiv.com/en/financial-data. Last accessed: Oct'22.

[56] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. Info. Processing and Management, 1988.

[57] F. Schomm, F. Stahl, and G. Vossen. Marketplaces for data: An initial survey. ACM SIGMOD Record, 2013.

[58] C. Shapiro and H. R. Varian. Information Rules: A Strategic Guide to the Network Economy. Harvard Business School Press, 2000.

[59] L. S. Shapley. A Value for n-Person Games. RAND Corporation, 1952.

[60] M. Spiekermann. Data marketplaces: Trends and monetisation of data goods. Intereconomics, 2019.

[61] F. Stahl, F. Schomm, L. Vomfell, and G. Vossen. Marketplaces for digital data: Quo vadis? Computer and Information Science, 10, 2017.

[62] F. Stahl, F. Schomm, and G. Vossen. The Data Marketplace Survey Revisited. Westf. Wilhelms-Univ., ERCIS, 2014.

[63] C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis. Conditional Variable Importance for Random Forests. BMC Bioinformatics, 2008.

[64] R. Tibshirani. Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society (Series B), 1996.

[65] Veracity. Veracity by DNV GL. Find the Right Tools for your Industry Needs. https://store.veracity.com/. Last accessed: Oct'22.

[66] C. Wu, R. Buyya, and K. Ramamohanarao. Cloud Pricing Models: Taxonomy, Survey, and Interdisciplinary Challenges. ACM Computing Surveys, 2019.

[67] H. Yu and M. Zhang. Data Pricing Strategy based on Data Quality. Computers and Industrial Engineering, 112:1–10, 2017.

[68] H. Zou and T. Hastie. Regularization and Variable Selection via the Elastic Net. Journal of the Royal Statistical Society. Series B. (Statistical Methodology), 2005.

Anexo 5: Try Before You Buy

MLEDGE Report: Try-Before-You-Buy reloaded and federated

Alexandr Goultiaev Tolstokorov
alexandr.goultiaev@imdea.org
IMDEA Networks Institute
Leganés, Madrid, Spain

Santiago Andrés Azcoitia
santiago.azcoitia@imdea.org
IMDEA Networks Institute
Leganés, Madrid, Spain

Nikolaos Laoutaris
nikolaous.laoutaris@imdea.org
IMDEA Networks Institute
Leganés, Madrid, Spain

1 Motivation

As data is increasingly used to drive decision-making processes, companies need obtaining suitable data and insights from third parties to improve the accuracy and efficiency of their models. A number of commercial data marketplaces (DMs) have appeared in the market to mediate between providers and consumers and to manage data sharing and transactions between them [1].

The operation of a data marketplace entails complex technical and economic challenges due to the elusive nature of data as an economic good. In particular, the value of data is strongly dependent on the use case, and two similar datasets could be very valuable for a ML task A and not so valuable for a ML task B, and data from different providers may yield very different values in training a single ML model, irrespective of the amount of information they carry [2]. In summary, buyers are not able to realise the value of a piece of data for them until they are able to test the data on their own model.

A number of solutions have been proposed to circumvent this problem, known as Arrow's information paradox. On the one hand, commercial data marketplaces are addressing this by sharing outdated data samples, by providing metadata to potential buyers, and most interestingly by allowing potential data buyers try versions or samples of a dataset in "sandboxed" environments that do not allow buyers to download data. On the other hand, most DM proposals from the research community have responded to this challenge by allowing buyers to share their ML models with DMs and letting the platform provide data according to the price paid by users [3, 4], or find the combination of datasets [5, 6] that best suits the ML task. These solutions, which have not been implemented yet, share a common drawback: buyers are supposed to trust the platform and share a most likely sensitive and confidential ML model. Such ML models are often complex, which makes training them with different combinations of eligible datasets to select the most suitable ones computationally expensive. Finally, none of these solutions evaluate the cost of processing a data transaction, which can be prohibitive and hence can threaten the viability of the DM.

Our objective is to provide more scalable and explainable alternatives to these proposals that avoid buyers sharing confidential intellectual property with DMs, and make the data purchasing process more scalable. This solution will also be useful in federated environments in the cloud edge, where data will be distributed in different edge nodes, and federated clients in those nodes will be in charge of evaluating data assets and returning the results to a centralized control node that will be accessed by data buyers.

***Our planned contributions*:** This paper will introduce a practical architecture of a data marketplace that evaluates and charges for data and for processing transactions. Moreover, it introduces the idea of valuation functions (VFs) and puppet valuation models (PVMs) to reduce the processing costs, and shows that data buyers are able to find suitable data tailored to their use case without necessarily sharing information about their (often) complex ML tasks. Then we plan to test different PVMs and VFs for different use cases related to image classification problems and a forecasting taxi-ride demand in different districts of Chicago [7] use case, and we plan to investigate the efficiency vs. accuracy trade-off and evaluate the cost of processing and selling such data.

2 Problem Setting

Let us assume that the buyer is seeking to buy data in order to optimise a certain AI/ML model ($\mathcal{M}$), i.e., to maximise the value of a metric $a(\cdot)$ which we will call *accuracy* and can accommodate any model performance metrics such as F1 score, precision, mean absolute error, etc. We will assume the buyer has also a limited budget B to pay for the cost of such transaction.

Let us denote by $\mathcal{S}$ the set of sellers able to provide suitable data for the task $(\mathcal{M}, a)$ at a given DM. We will denote by $\mathcal{C}$ the catalogue (set of possible data inputs) for $(\mathcal{M}, a)$ based on data from those sellers. We will assume the pricing function $p : \mathcal{C} \longrightarrow \mathbb{R}^+$ is known by the DM and subadditive. Based on the way real data marketplaces work, we will assume that the price the buyer pays for data is set according to different criteria, namely i) the volume of samples purchased, ii) a realistic distribution obtained from existing prices in commercial DM catalogues, iii) a random uniform price distribution that has nothing to do with the value of data [1], or iv) according to the accuracy it brings to the task $(\mathcal{M}, a)$.

Apart from presenting the buyers different eligible datasets $d \subseteq \mathcal{C}$ and their prices $p(d)$, we will assume that the data marketplace can evaluate the performance of an AI/ML model trained with data it has access to, and buyers can therefore ask the DM for $a(d)$ to make the decision of which dataset(s) to buy. We will assume that the DM will charge the buyer for the corresponding processing cost a quantity that will depend on the task and the data to evaluate, denoted as $\gamma(\mathcal{M}, a, d)$. We will assume that this cost is proportional to the processing time invested in the transaction, and we will take real costs of IaaS in public cloud platforms to evaluate the processing charges issued by the DM.

For now, we will assume that the DM is also willing to perform some free valuation for the buyer, by providing him with the revealed accuracies for a small subset of $\mathcal{C}$ which we denote the known set $\mathcal{K} \subset \mathcal{C}$. The rest of the datasets for which accuracy is unknown belong to the unknown set $\mathcal{U} \subseteq \mathcal{C}$, therefore the datasets on sale in the catalogue are $\mathcal{C} = \mathcal{K} \cup \mathcal{U}$. This assumption is relaxed later as the purchasing strategies developed can work in the setting where nothing is provided for free $|\mathcal{K}| = 0$.

Therefore, we will assume that the cost of a data transaction for the buyers, denoted as $c(d)$, includes a fixed cost component (compensating, for example, for the registration of a transaction in a blockchain ledger, or for the resulting costs or transferring the data) denoted as c_f, the cost of acquiring the data ($p(d)$), and a variable cost component proportional to the amount of processing required to select d and calculate the corresponding payoffs for sellers, which we will denote as c_p and that will in turn depend on the datasets $d' \in K$ the buyer has asked the DM to evaluate to make this decision.

$$c(d) = p(d) + c_f + c_p = p(d) + c_f + \sum_{d' \in K} \gamma(\mathcal{M}, a, d') \tag{1}$$

2.1 Purchasing Strategies

In this setting, the problem of selecting suitable data products using the capabilities of data marketplaces is not straightforward and requires careful purchasing strategies to optimise $argmax_{d \in \mathcal{C}} a(d)$ subject to $c(d) < B$. Optimally, buyers would like to get the $d^\star, \forall d \subseteq \mathcal{C}, a(d^\star) \geq a(d)$. But for finding $d^\star$ a buyer needs to ask the DM to evaluate any possible combination of data products, which usually leads to prohibitive processing costs. Since it is not feasible to reveal the value of any possible combinations of elements in $\mathcal{C}$, buyers need purchasing strategies to select promising data to be evaluated by the DM and, eventually, be bought to increase the accuracy of their model.

***Impossibility of maximising absolute quality with a deterministic algorithm*:** Given that qualities of the datasets in $\mathcal{U}$ are unknown, it follows that no deterministic algorithm is guaranteed to be able to buy the affordable dataset of highest (*absolute*) quality.

Lemma 1 *Given prices and qualities for datasets in $\mathcal{K}$, prices for datasets in $\mathcal{U}$, and a budget B for buying datasets and revealing unknown qualities at cost R per individual dataset, it is impossible to guarantee that the dataset with highest quality and price up to B will be identified in $\mathcal{K} \cup \mathcal{U}$.*

This can be easily established via the following counter-example. Assume that o is the optimal dataset that has $p(o) \leq B$ and $a(o) \geq v$ for any $v \in \mathcal{C}$ and that $o \in \mathcal{U}$. Assume also that $|\mathcal{U}| > B/R$. This means that independently of the actual price $p(o)$, the buyer may not be able to identify o even if he uses all his budget to reveal the price of $\lfloor B/R \rfloor$ datasets in $\mathcal{U}$.

2.1.1 Greedy Algorithm

Granted the impossibility of maximising *absolute* quality, let us define the alternative notion of *guaranteed* quality for the buyer as the maximum quality dataset that he can afford at a certain point in time with the available information and budget at that time. Knowing the prices and qualities in $\mathcal{K}$, the prices in $\mathcal{U}$, and having a budget B, if we take the set $\mathcal{K}$ to be sorted in order of decreasing accuracy and the set $\mathcal{U}$ to be sorted in order of increasing price. Then the initial guaranteed quality of the buyer is the first and therefore highest accuracy dataset in the known set $a(k(1))$ where $k \in \mathcal{K}$. If, he starts exploring $\mathcal{U}$ then his guaranteed quality can become *at least*:

$$g^* = \max\left(a(k(1)), \max_{u \in \mathcal{U}}(a(u))\right) \tag{2}$$

Algorithm 1 Greedy Algorithm for Maximum Guaranteed Quality

Require: $\mathcal{K}$ (known datasets), $\mathcal{U}$ (unknown datasets), B (budget), R (cost to reveal)

Initialize:
    $remaining\_budget \leftarrow B - B - p(k(1))$
    $best\_dataset \leftarrow k(1)$
    $i \leftarrow 0$
    Sort $\mathcal{K}$ in order of descending accuracy, and $\mathcal{U}$ in order of increasing price
while $remaining\_budget > R$ **do**
    Pay R to learn the quality of $\mathcal{U}(i)$
    if $a(\mathcal{U}(i)) > best\_dataset$ and $p(\mathcal{U}(i)) \leq B - R * i$ **then**
        Set $best\_dataset = a(\mathcal{U}(i))$ and `leftover` $= B - p(\mathcal{U}(i)) - R * i$
    end if
    $i \leftarrow i + 1$
end while
return $best\_dataset$

2.1.2 Bandit Algorithm

Depending on the distributions of qualities and prices in the unveiled set $\mathcal{K}$ and the unknown set $\mathcal{U}$, heavier bias towards exploration might lead to better quality found and perhaps even the global maximum dataset being found. For this case, we can postulate that a buyer is willing to risk a certain drop in guaranteed quality g^* for an increase in the expected accuracy e^* in the hope of finding a dataset $u(k)$ whose quality is higher than the guaranteed maximum $a(u(k)) > g^*$. This can occur if the dataset $u(k)$ is outside the exploration depth achieved by the greedy algorithm, $k > \frac{B - p(g^*)}{R}$. We denote this *risk* factor s, which dictates how much sacrifice in quality the buyer is willing to incur. It is provable that reserving the price of a cheaper yet slightly lower accuracy dataset in $\mathcal{K}$ as the exploration starting point can lead to deeper exploration and perhaps better results. This alternate starting point in $\mathcal{K}$ we denote as $k(j) \in \mathcal{K}$.

$$e^* = \max\left(a(k(j)), \max_{u \in \mathcal{U}}(a(u))\right) \tag{3}$$

To choose our starting point, we compute the probability of choosing a dataset $k(j)$ in $\mathcal{K}$ and reserving its price $p(k(j))$, denoted as ε_j, as the ratio of sacrifice in quality $\Delta a(j) = a(k(1)) - a(k(j))$ to increase in leftover budget $\Delta l(j) = l(k(j)) - l(k(1))$, $\varepsilon_j = \frac{\Delta l(j)}{\Delta a(j)}$. The weights are normalized so that $\sum_{j=1}^{|\mathcal{K}|} \varepsilon_j = 1$.

Algorithm 2 Bandit Algorithm for Maximum Expected Quality

Require: $\mathcal{K}$ (known datasets), $\mathcal{U}$ (unknown datasets), B (budget), R (cost to reveal), s (risk)

Initialize:
    $datasets\_explored \leftarrow$ boolean array for datasets in $\mathcal{U}$
    $linear\_regression\_performance \leftarrow$ array for performance tracking
    $remaining\_budget \leftarrow B$
    $best\_dataset \leftarrow k(j) \in \mathcal{K}$
    $t_p \leftarrow 0$
while $remaining\_budget > R$ **do**
    Calculate $average\_error$ as average of $linear\_regression\_performance$
    Update t_p using GET_ADJUSTED_TP($average\_error, s, t_p$)
    Calculate ρ between known datasets in $\mathcal{K}$ and unveiled datasets in $\mathcal{U}$
    if t_p suggests using linear regression ($\rho > t_p$) **then**
        Apply linear regression on known datasets
        Predict accuracies in $\mathcal{U}$
        Select a dataset in $\mathcal{U}$ based on predictions and affordability
        Update $linear\_regression\_performance$
    else
        Select and reveal a dataset from $\mathcal{U}$ with heavy bias on cheaper datasets
    end if
    Update $datasets\_explored$ and $remaining\_budget$
    if revealed dataset has higher accuracy and is affordable **then**
        Update $best\_dataset$
    end if
end while
return accuracy of $best\_dataset$

3 Scenarios

For the Benchmark testing we have used the task of image classification on the MNIST dataset. To simulate different realistic DM scenarios where our purchasing strategies could be tested and compared we have introduced non-iid distributions of labels, dataset sizes, image quality, Moreover, as part of prior research and surveys in commercial data market [1], we have identified four pricing schemes commonly present in DMs, namely volume-based pricing, accuracy-correlated pricing, random pricing, and pricing according to a market-based distribution. Next we provide additional details on how we carried out non-iid distributions, and on the different price schemes considered in the tests.

3.1 Non-iid: Label and Size Distribution

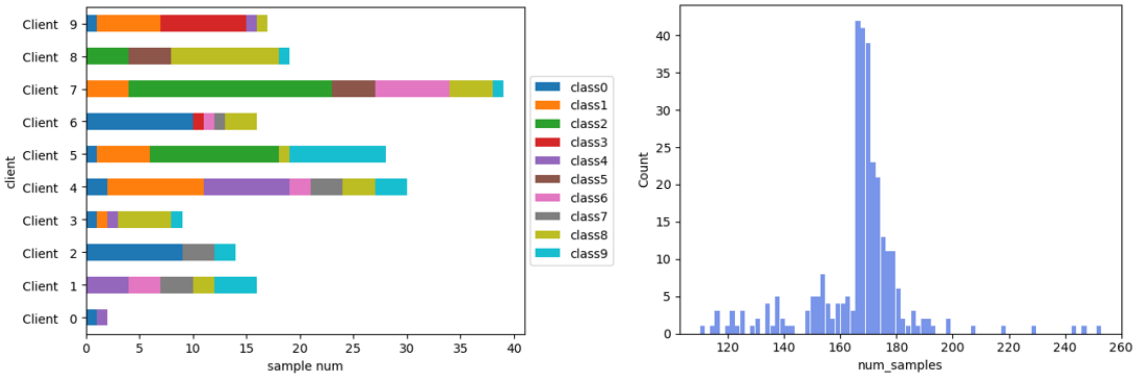
Using a Dirichlet distribution, we are able to split the MNIST dataset into partitions simulating the datasets on sale on a DM using a α parameter to decide the non-iid degree as shown in Figure 1.

3.2 Non-iid: Quality Distribution

To simulate the differing quality of images offered by the sellers in a DM we introduce different types of noise to the images, namely: Gaussian noise, speckle noise and salt and pepper noise as show in Figure 2

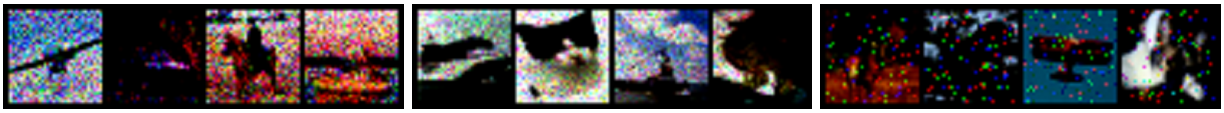
3.3 Pricing Schemes

Volume-based pricing: This is the case in which the price per image is the same for all sellers.



(a) Label Distribution of 10 sellers (b) Size distribution of 100 partitions

Figure 1: Representation of the resulting simulated DM catalogue with MNIST Non-iid split.



(a) Gaussian Noise representation (b) Speckle Noise representation (c) S&P Noise representation

Figure 2: Representation of the resulting images after applying different noises.

Random pricing: In this case the price per dataset is set independently and following a uniform random distribution.

Pricing correlated with accuracy: Some literature works propose that DMs charge prices relevant to the utility brought to the buyer. We will assume that in our case the utility of a dataset is equivalent to the accuracy of that dataset on the buyers model.

Market-based pricing: From surveying existing commercial DMs we have obtained a distribution for image dataset prices following two distinct log-normal distributions. The two classes represent "general purpose" datasets which are generally larger and less task dependent and "custom" datasets which are generally smaller and more tailored to a specific task. We simulate this case by adding noise to the larger partitions in our split and assigning those partitions the price distribution of the "general purpose" datasets, whereas the smaller cleaner datasets are assigned the price distribution of "custom" datasets.

4 Preliminary Results

To compare our algorithms, we have implemented five other algorithms present in the existing exploration vs exploitation research literature [8, 9, 10].

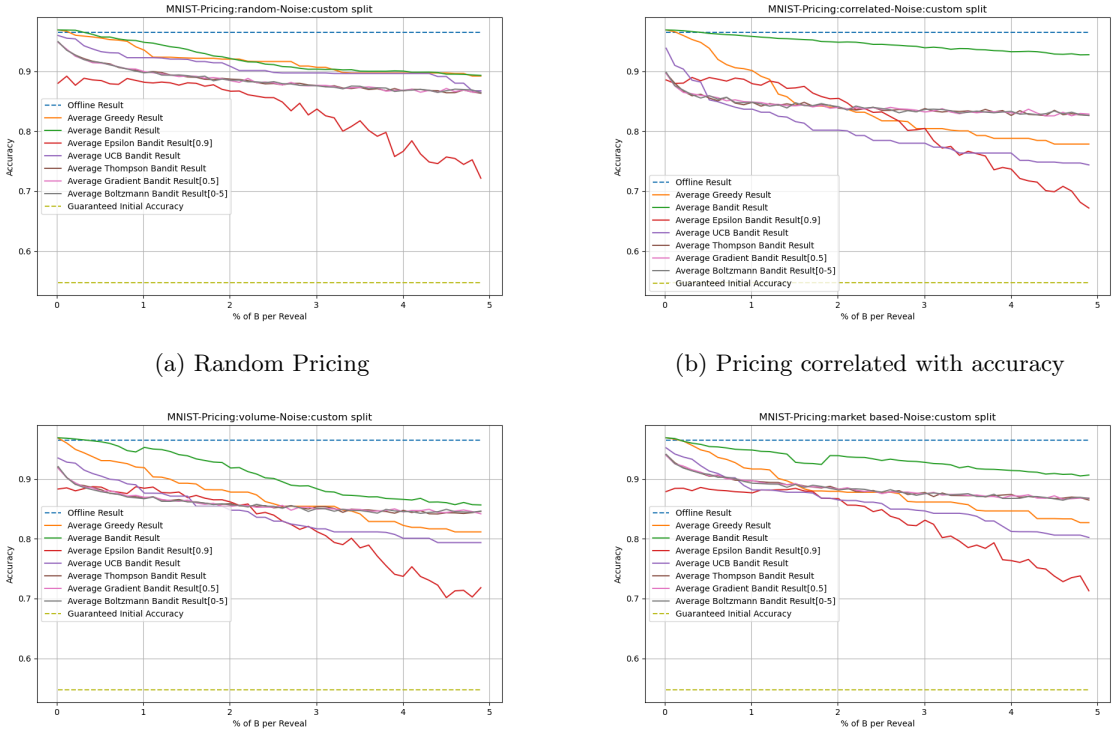
Epsilon Greedy Bandit: In the Epsilon-Greedy strategy, with probability epsilon, you explore by revealing the accuracy of a dataset in $\mathcal{U}$, and with probability 1 - epsilon, you exploit by choosing the best dataset found so far that is affordable with the remaining budget. The value of epsilon determines the balance between exploration and exploitation – a higher epsilon meaning more exploration.

Upper Confidence Bound (UCB) Bandit: UCB involves selecting the dataset that has the highest upper confidence bound, balancing between the mean accuracy and the uncertainty associated with it.

Thompson Sampling Bandit: Thompson sampling in our context would involve maintaining a probability distribution for the accuracy of each dataset in $\mathcal{U}$. As you reveal more datasets, you update these distributions.

Softmax (Boltzmann) Exploration: Softmax - aka Boltzmann - Exploration is a strategy where the probability of selecting an option is based on the relative estimated values of the options, using a softmax function. This approach is softer than the hard max decision rule used in UCB and can be more explorative.

Gradient bandit Algorithm: Gradient Bandit Algorithms use a numerical preference for each option. The preferences are updated based on received rewards, and the probability of selecting each option is determined using a softmax function over these preferences.



(a) Random Pricing

(b) Pricing correlated with accuracy

(c) Volume-based Pricing

(d) Market-based Pricing

Figure 3: Performance Comparison of our algorithms on MNIST.

5 Conclusion and Next Steps

As shown in Figure 3, our Bandit algorithm seems to outperform the greedy and the other algorithms from exploration vs exploitation literature in all four of our case scenarios. It proves to be more robust as it deals with even unfavourable pricing schemes such as random pricing while still maintaining its strength in the correlated and volume-based pricing scenarios.

Next Steps: We are progressing with this task in the following directions:

1. Scanning for optimal parameters both for the bandit and the existing literature solutions to properly show that our algorithm outperforms them,

2. Introducing VFs and PVMs and testing them to find out about the accuracy vs cost trade-off that is of interest to us in the scope of this work, and finally

3. Moving on to a second scenario to validate our purchasing strategies performance in that case and explore the effect and trade-off introduced by the use of VFs and PVMs. In this case, we will evaluate and select data of individual taxis for the task of forecasting taxi-ride demand per district in Chicago.

References

[1] S. Andrés Azcoitia and N. Laoutaris. A survey of data marketplaces and their business models. 2022.

[2] Santiago Andrés Azcoitia, Marius Paraschiv, and Nikolaos Laoutaris. Computing the relative value of spatio-temporal data in wholesale and retail data marketplaces, 2020.

[3] Anish Agarwal, Munther Dahleh, and Tuhin Sarkar. A marketplace for data: An algorithmic solution. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 701–726, 06 2019.

[4] Lingjiao Chen, Paraschos Koutris, and Arun Kumar. Towards model-based pricing for machine learning in a data marketplace. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, page 1535–1552, New York, NY, USA, 2019. Association for Computing Machinery.

[5] Santiago Andrés Azcoitia and Nikolaos Laoutaris. Try before you buy: A practical data purchasing algorithm for real-world data marketplaces, 2020.

[6] Xinlei Xu, Awni Hannun, and Laurens Van Der Maaten. Data appraisal without data sharing. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 11422–11437. PMLR, 28–30 Mar 2022.

[7] City of Chicago. Taxi trips: City of chicago: Data portal, Dec 2023.

[8] WILLIAM R THOMPSON. ON THE LIKELIHOOD THAT ONE UNKNOWN PROBABILITY EXCEEDS ANOTHER IN VIEW OF THE EVIDENCE OF TWO SAMPLES. *Biometrika*, 25(3-4):285–294, 12 1933.

[9] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

[10] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.