*ECAI 2024*
*U. Endriss et al. (Eds.)*


*doi:10.3233/FAIA241065*

# Federated Online Learning for Heavy Hitter Detection

**Paula Silva**[a,b,*], **João Vinagre**[c] **and João Gama**[a,d]

[a]INESC TEC, Porto, Portugal
[b]FEUP - University of Porto, Porto, Portugal
[c]Joint Research Centre - European Commission, Seville, Spain
[d]FEP - University of Porto, Porto, Portugal
ORCID (Paula Silva): https://orcid.org/0000-0002-3144-3600, ORCID (João Vinagre): https://orcid.org/0000-0001-6219-3977, ORCID (João Gama): https://orcid.org/0000-0003-3357-1195

**Abstract.** Effective anomaly detection in telecommunication networks is essential for securing digital transactions and supporting the sustainability of our global information ecosystem. However, the volume of data in such high-speed distributed environments imposes strict latency and scalability requirements on anomaly detection systems. This study focuses on distributed heavy hitter detection in telephone networks – a critical component of network traffic analysis and fraud detection. We propose a federated version of the Lossy Counting algorithm and compare it to its centralized version. Our experimental results reveal that the federated approach can detect considerably more unique heavy hitters than the centralized method while enhancing privacy. Furthermore, Federated Lossy Counting does not need a large amount of centralized processing power since it can leverage the networked infrastructure with minimal impact on bandwidth and computing power.

## 1 Introduction

The fast-growing volume and data speed in network environments creates substantial challenges for conventional data processing architectures. Centralized systems often struggle with scalability, latency, and privacy issues, especially when handling sensitive information. To address these challenges, Federated Learning (FL) has emerged as a distributed machine learning paradigm designed to address the increasing demands for data privacy and efficient data processing across decentralized sources [3]. Unlike traditional centralized learning, FL enables multiple clients, such as smartphones, IoT devices, or various institutions, to train a global model collaboratively without sharing their raw data. This is achieved by exchanging model updates rather than the actual data, thereby preserving the privacy of each client's dataset [4]. Given their distributed nature and the high volume and speed of traffic, FL applications have a high potential for solving problems related to network diagnosis and operation.

Interconnect bypass fraud, referred to as toll bypass fraud, interconnect fraud, GSM Gateway fraud, or SIM Boxing, involves unauthorized traffic rerouting through another carrier's network to avoid termination fees. This fraud typically leverages low-cost IP connections to forward international calls, resulting in substantial financial losses within the telecommunications sector. Moreover, fraudsters employ increasingly sophisticated strategies to circumvent traditional detection mechanisms. Typically, interconnect bypass fraud manifests as a sudden surge in call volume, complicating detection and mitigation efforts [17]. Effective detection and mitigation strategies are crucial to counter these fraudulent activities. Two powerful analytical approaches that can be leveraged to address this issue are frequent itemset mining and heavy hitter detection. These methodologies can uncover patterns and anomalies in data that are not easily detectable through conventional means.

Heavy hitter detection focuses on identifying the few items that account for some significant fraction of the activity in a dataset. In telecommunications, this method can effectively spot the number of connections that generate unusually high traffic, indicative of interconnect bypass fraud. For example, a phone number frequently appearing in call logs with a high volume of international connections made over short periods might be flagged as a heavy hitter. The implementation of heavy hitter detection algorithms, such as the Lossy Counting algorithms [16], allows for real-time analysis of streaming data, making it possible to detect and respond to potential fraud cases as they occur.

This manuscript delves into applying the Lossy Counting algorithm within a federated framework to detect heavy hitters in network traffic—key elements or patterns frequently occurring within a data stream. This method leverages federated learning to enhance the detection of anomalous traffic patterns across various network nodes while preserving the privacy and security of the data involved. In essence, our approach consists of counting heavy hitters locally at each participating node in the network and then combining the contributions of each node in a central node that estimates the network's heavy hitters.

This process does not require sharing raw data, potentially improving privacy and reducing bandwidth requirements. Furthermore, since the system is distributed, computation and communication overhead is more straightforward to accommodate within the existing network resources, with potential savings in computational power, network bandwidth, maintenance actions, and economic and environmental benefits.

The main contributions of this manuscript are as follows:

- Aggregation Method: We propose an aggregation method focused on ranking aggregation within discrete bucket windows. This technique is particularly adept at summarizing and synthesizing information over specified intervals, which is crucial for timely decision-making in dynamic environments.

* Corresponding Author. Email: paula.r.silva@inesctec.pt

- Comparative Experimental Setup: Our experimental framework directly compares the performance of centralized and federated learning approaches. This setup demonstrates the efficacy of our integrated solutions and provides valuable insights into their practical applicability and advantages in various scenarios.

## 2 Related Works

This section delves into the intricacies of frequent itemset mining within online and federated learning realms. Each subsection addresses theoretical frameworks and discusses practical implementations and challenges, providing a comprehensive overview of how frequent itemset mining is adapted to these learning paradigms.

### *2.1 Online Learning and Frequent Itemsets*

Frequent Itemset Mining (FIM) is a fundamental task in larger scale data stream mining, aimed at discovering sets of items that appear frequently together in a given dataset [7, 2]. The body of research on Frequent Itemset Mining (FIM) includes algorithms designed to uncover the hidden patterns of frequent itemsets more efficiently, aiming for reduced runtime and lower memory usage as data volumes grow [9, 22].

The Lossy Counting algorithm addresses the need for efficient, scalable, and accurate streaming data processing to extract meaningful patterns with reduced computational overhead in environments characterized by high velocity and volume. The primary application of this algorithm is finding heavy hitters in network traffic, which addresses the need for effective network management and monitoring [8]. The algorithm provides a way to approximate the frequencies of different *itemsets* without keeping the entire data history, thus conserving memory and processing resources. The authors discuss enhancements to the traditional Lossy Counting method, offering probabilistic guarantees on accuracy and reducing false positives and memory consumption in network traffic analysis.

Veloso et al. [15] explore using advanced heavy-hitter algorithms to identify interconnect bypass fraud in telecommunications. The researchers employ the Lossy Counting algorithm, demonstrating its ability to manage and analyze large data volumes effectively, thus accurately identifying fraudulent activities. Further developing this approach, Veloso et al. [17] integrates a fast forgetting technique with Lossy Counting to better detect unusual patterns, such as sudden spikes in calls from particular numbers. This innovation enhances the algorithm's ability to uncover fraud patterns characterized by repeated and mirrored behaviours across various destination numbers. The authors highlight that their method not only supports but also enhances the performance of existing telecom fraud detection processes regarding speed, memory efficiency, and sensitivity. The *River* framework [13], a specialized machine-learning library designed for real-time, online learning, supports the implementation of these methodologies.

Even though the Lossy Counting algorithm is not a new addition to the field of data stream mining, it continues to hold significant relevance and is actively used to address contemporary challenges in online learning. Researchers and practitioners find it an indispensable tool in the era of big data, where the demand for real-time analytics and efficient data processing mechanisms is ever-growing. Thus, while it may not be a new tool, Lossy Counting remains a cornerstone in the toolkit of algorithms for online learning and data stream processing [21, 18, 14].

### *2.2 Frequent Itemsets in Federated Learning*

Frequent itemsets in federated learning focus on identifying patterns or item combinations that commonly appear across decentralized data sources while adhering to privacy principles and maintaining data locality. The main challenge is modifying traditional itemset mining algorithms for a federated context where data does not leave its local environment. These algorithms must efficiently aggregate results from each node to reflect the global training accurately.

Li et al. [11] introduce a novel algorithm, *FIML*, which mines frequent itemsets within local differential privacy standards. *FIML* combines a padding-and-sampling-based frequent oracle with an interactive method, achieving high accuracy while preserving privacy. However, its complexity and the need for precise parameter tuning may limit its practicality.

A novel framework for frequent pattern mining (FPM) tasks that preserve user privacy while maintaining high data utility was introduced in [19]. *FedFPM* operates under a federated analytics framework, which employs an interactive query-response approach, leveraging the *Apriori* property and *Hoeffding's* inequality to refine queries and aggregate data across clients. The framework performs better than existing methods but requires careful parameter configuration and assumes stable network conditions.

Hei et al. [10] introduces an intrusion detection framework that integrates blockchain with Federated Learning (FL) to secure Internet of Things (IoT) devices while preserving user privacy. This approach, however, may encounter issues such as increased computational and storage demands on IoT devices and potential latency introduced by blockchain, which could impede real-time detection capabilities. Similarly, Wu et al. [20] presents an FL framework tailored for industrial IoT, employing a "pre-large concept" to reduce data scans and bolster privacy. Despite its advancements, this framework struggles with synchronising local models and aggregation of parameters, particularly in dynamic settings where real-time responses are essential.

The paper [6] propose the framework *FedFIM* that leverages FL combined with local differential privacy and use of a *non-Apriori* algorithm for pattern discovery. The computational demands and the complexity of managing cryptographic keys could challenge the practical deployment. Additionally, the increased complexity of the system could affect scalability and real-time performance, particularly in environments with stringent response time requirements.

The paper [1] discusses a method to enhance data privacy in cyber-physical systems (CPS) using federated learning for frequent itemset mining. The complexity and computational demands of maintaining an attention-based model and FL system could be significant, especially in systems with constrained resources. Additionally, the reliance on local data processing and the need for effective synchronization across distributed networks pose challenges in ensuring consistency and reliability of the learning process.

## 3 Frequent Itemsets with Lossy Counting

Manku and Motwani [12] present the Lossy Counting algorithm, a one-pass algorithm for computing frequency counts that exceed a user-specified threshold on data streams. While the results are approximate, the algorithm ensures that the error does not surpass a parameter specified by the user. Lossy Counting requires two user-specified parameters: a support threshold $s \in [0, 1]$, and an error parameter $\epsilon \in [0, 1]$ such that $\epsilon \ll s$. The algorithm can produce a list of item sets and their estimated frequencies at any time.

Let $N$ represent the current length of the stream. The results generated will adhere to the following assurances:

- All items whose true frequency exceeds $s \times N$ are output. There are no false negatives;
- No item whose true frequency is less than $(s - \epsilon) \times N$ is output;
- Estimated frequencies are less than the true frequencies by at most $\epsilon \times N$.

The pseudo-code is presented in Algorithm 1 [12]. The process begins by segmenting the incoming data stream into discrete units, or buckets, each containing $w = \lceil \frac{N}{\epsilon} \rceil$ transactions. These buckets are sequentially assigned unique identifiers, bucket *ids*. Let $b_{current}$ represent the identifier for the current bucket. As each element is processed, it checks whether it is already in the data structure. If it is, increment its count. If not, add it with an initial count and a $\delta$ value to account for potential errors. For any given element $e$ within the stream, $f_e$ represents its true frequency up to the current point in the stream. At the end of each bucket (a stream segment defined by $w$), the algorithm reviews each item in the data structure to determine if it should be removed based on its count and $\delta$ value. After processing the stream, the data structure $D$ contains the elements and their approximate counts.

**Algorithm 1** Lossy Counting Algorithm

**Input:** Stream of elements $S$, error parameter $\epsilon$, support $s$
**Output:** Approximate frequencies of items
$b\_current \leftarrow 1$ ▷ Current bucket number
$n \leftarrow 0$ ▷ Total elements processed
$w \leftarrow \lceil 1/\epsilon \rceil$ ▷ Width of a bucket
Initialize empty data structure $D$
**for** each element $e$ in stream $S$ **do**
    $n \leftarrow n + 1$
    **if** $e$ is in $D$ **then**
        $D[e].count \leftarrow D[e].count + 1$
    **else**
        $D[e].count \leftarrow 1$
        $D[e].\delta \leftarrow b\_current - 1$
    **end if**
    **if** $n \mod w == 0$ **then**
        **for** each item $i$ in $D$ **do**
            **if** $D[i].count + D[i].\delta \leq b\_current \times s$ **then**
                Remove $i$ from $D$
            **end if**
        **end for**
        $b\_current \leftarrow b\_current + 1$
    **end if**
**end for**
**return** $D$

## 4 Federated Lossy Counting

This manuscript presents an enhanced implementation of the Lossy Counting algorithm within a federated learning context. Our approach adopts a Horizontal Federated Learning architecture, wherein each client node independently executes the probabilistic counting process. The server's role is to aggregate these counts from client nodes. Notably, the server does not disseminate any updated weights or parameters back to the clients in this implementation.

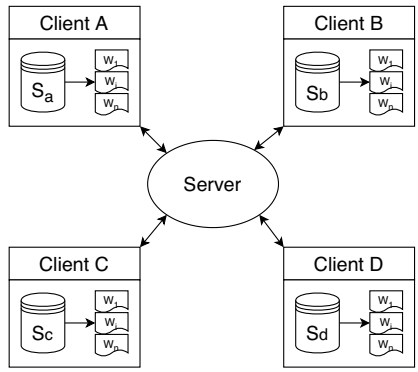


**Figure 1.** Horizontal FL architecture

Lossy Counting inherently operates in a one-way pass mode, focusing solely on aggregation without iterative model updates.

Figure 1 illustrates the architecture of a horizontal federated learning environment comprising one server and multiple clients (Client A, Client B, Client C, Client D). Each client holds its subset of data, denoted as $S_a$, $S_b$, $S_c$, and $S_d$ for Clients A, B, C, and D respectively. These clients independently process their data stream segmented into buckets represented by $w_1$ to $w_n$, each containing $w = \lceil \frac{N}{\epsilon} \rceil$ transactions. The server aggregates these summaries from all clients to update the global counting for each data bucket.

Let $D$ be a dataset represented where each row corresponds to an observation with values of $Id$ and $DateTime$. Each client starts the process by receiving the first bucket of streaming data. The bucket is computed according to the $\epsilon$ parameter, and the number of rows corresponds to $\lceil 1/\epsilon \rceil$. For example, if $\epsilon = 0.0005$, the bucket size will be 1000 rows. Then, each client runs the Algorithm 1 and returns the top-k counting ranking. The server aggregates the rankings by the sum of probabilistic counting according to the time interval.

**Algorithm 2** Aggregate Bucket Ranking

**Input:** Dataset $D$ consisting of tuples $(id_d, v_d, t_d)$
**Input:** Time interval $[t_{\text{begin}}, t_{\text{end}}]$, and integer $k$
**Output:** Top $k$ entries from $D$ based on summed values $v_d$
Initialize an empty list $ValidEntries$
Initialize $Agg$ to store sums of $v_d$ based on $code_d$
**if** $D$ is not empty **then**
    **for all** $(id_d, v_d, t_d) \in D$ **do**
        **if** $t_{\text{begin}} \leq t_d \leq t_{\text{end}}$ **then**
            **if** $id_d$ in $Agg$ **then**
                $Agg[id_d] \leftarrow Agg[id_d] + v_d$
            **else**
                $Agg[id_d] \leftarrow v_d$
            **end if**
        **end if**
    **end for**
    **for all** $id, total\_value$ in $Agg$ **do**
        Append $(id, total\_value)$ to $ValidEntries$
    **end for**
    Sort $ValidEntries$ in descending order based on $total\_value$
    $R \leftarrow$ the first $k$ elements of $ValidEntries$
**else**
    $R \leftarrow$ an empty list
**end if**
**return** $R$

Algorithm 2 processes a dataset $D$ to find and rank entries based on aggregated values within a specified time interval. The purpose of this algorithm is to aggregate values (denoted as $v_d$) from a dataset $D$ based on a standard id ($id_d$) within a specified time range ($t_{\text{begin}}$ to $t_{\text{end}}$). The algorithm sums these values and ranks the entries based on these summed values, finally returning the top $k$ entries. Initialize an empty list $ValidEntries$ to store the resulting tuples of id and their aggregated values. Initialize a data structure $Agg$ to store the summed values for each unique id. The algorithm iterates over each entry in the dataset $D$. If an entry's timestamp $t_d$ is within the specified time interval, it checks if the id $id_d$ already exists in the array $Agg$. If it does, it updates the existing sum. If not, it initializes a new sum for that code. After processing all entries, the algorithm iterates over the array $Agg$ and appends each code and its corresponding total value to the list $ValidEntries$. Sort $ValidEntries$ in descending order based on the total value. The algorithm extracts the first $k$ entries from the sorted list to form the output $R$. If the dataset $D$ is empty, the algorithm returns an empty list.

The for-loop (Lines 8-13) processes each entry in $D$, yielding a complexity of $O(n)$, with $n$ denoting the number of entries in $D$. Inserting and updating the $Agg$ has an average-case time complexity of $O(1)$ per operation. Sorting the list $ValidEntries$ has a time complexity of $O(m \log m)$, where $m$ is the number of unique codes in $D$. The predominant time complexity is $O(n) + O(m \log m)$, where $m$ is less than or equal to $n$. The space complexity is $O(m + n)$ due to the storage requirements for the dictionary $Agg$ and the list $ValidEntries$. Regarding scalability, as the size of the dataset $D$ increases, the memory overhead of storing $Agg$ and $ValidEntries$ could become significant, especially with many unique codes.

## 5 Interconnect Bypass Fraud Detection

Interconnect Bypass Fraud (IBF) detection algorithms typically consume a stream $S$ of events, where $S$ contains information about the origin number $A\_Number$, the destination number $B\_Number$, the associated timestamp, and the status of the call. The expected output is a set of potentially fraudulent $A\_Numbers$. The telecom operator can then further inspect this set to confirm or dismiss fraudulent activity.

In IBF, we can observe three different types of abnormal behaviours: (i) the burst of calls, which are $A\_Numbers$ that produce enormous quantities of $\#calls$ (above the $\overline{\#calls}$ of all $A\_Numbers$) during a specific time window $W$. The size of this time window is typically tiny; (ii) repetitions, which are the repetition of some pattern ($\#calls$) produced by a $A\_Number$ during consecutive time windows $W$; and (iii) mirror behaviours, which are two distinct $A\_Numbers$ (typically these $A\_Numbers$ are from the same country) that produce the same pattern of calls ($\#calls$) during a time window $W$. The utilization of Lossy Counting provides a sophisticated approach to rapidly identify and analyze abnormal behaviours in telecommunications, thereby enhancing the detection and prevention of interconnected bypass fraud. In this experiment, we apply a single-pass heavy-hitter algorithm to identify calls to numerous destinations, from a single number to multiple receiving numbers ($B\_number$).

The experimental setup evaluates the differences between the centralized and federated versions of heavy hitter detection. As our baseline algorithm, we used the approach proposed by [12].

### 5.1 Data

We evaluated our proposal with two anonymized telecommunication data sets [5], which provides information about traffic flow (calls) into/out of gateways. Each call contains the following attributes:

- A-Number: origin number
- B-Number: destination number
- Date-Time: date and time of the call

The A- and B-numbers are unique identifiers of respectively the origin and destination of each call. Dataset A encompasses data collected from July 24, 2018, to October 21, 2018, comprising 83366367 calls. Meanwhile, Dataset B, which includes 32879670 calls, was gathered within one month from June 1, 2019, to June 30, 2019.

### 5.2 Centralized Approach

The centralized application of the Lossy Counting algorithm was employed on Dataset A and Dataset B. We applied a similar methodology proposed by Veloso et al. [17] using the two datasets and the Lossy Counting implementation available in River [13]. In this implementation, in addition to the support and epsilon parameters, a fading factor can impose a decay on the statistics from window to window. We set a $fading\_factor = 1$ in our experiments, meaning we did not retain statistics between buckets. The Lossy Counting algorithm was configured in three different setups to identify the most frequently occurring phone numbers, referred to as heavy hitters, within these datasets.

The three different parameter configurations were chosen based on the size of the bucket window ($1/\epsilon$) to make the centralised approach easily comparable with the federated approach regarding the number of examples by the window. Are as follows:

- Configuration 1: $epsilon = 0.0005$ (bucket size 2000), $support = 0.0005$
- Configuration 2: $epsilon = 0.00025$ (bucket size 4000), $support = 0.00025$
- Configuration 3: $epsilon = 0.000125$ (bucket size 8000), $support = 0.000125$

We applied the centralized Lossy Counting algorithm to each dataset under the three configurations mentioned above. The progressive reduction in the support and $\epsilon$ parameters lowers the frequency threshold required for a phone number to qualify as a heavy hitter. The adjustments in these configurations were designed to align with the federated approach, where a centralized bucket size of 2000 corresponds to two clients, each handling buckets of 1000, thereby facilitating the comparison between the two approaches.

Despite applying three distinct configurations of Lossy Counting, results regarding the identification of heavy hitters and the count of a-numbers have shown remarkable consistency across different settings. Table 1 highlights how slight variations in bucket size, from 2000 to 8000, primarily affect Dataset A, with a minimal impact on the number of heavy hitters detected, thus underscoring the algorithm's stability.

Figure 2 displays the activity patterns of the top-5 *A-numbers* in *Dataset A* over three months. This chart is crucial for identifying trends, anomalies, or consistent behaviours among the most active *A-numbers*, which could indicate heavy usage or potential misuse. Observing these patterns helps understand the dynamic nature of

**Table 1.** Number of A-numbers identified as heavy hitters with Centralized Lossy Counting

| Dataset | Bucket Size | | |
|---|---|---|---|
| | 2000 | 4000 | 8000 |
| A - 3 months | 20 | 19 | 19 |
| B - 1 month | 22 | 20 | 20 |

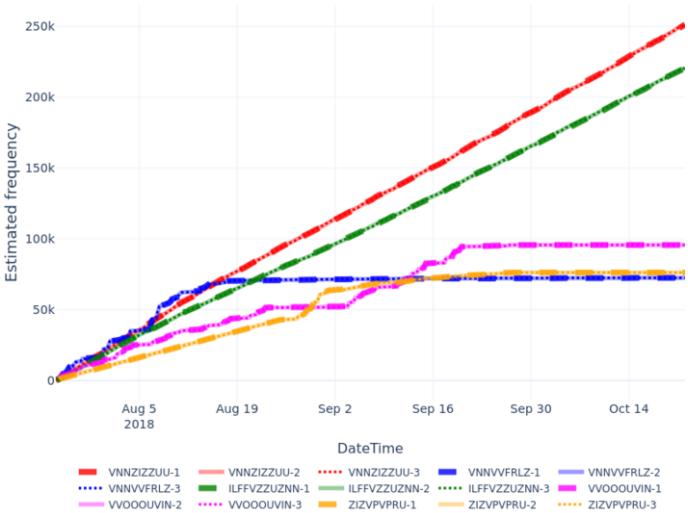


**Figure 2.** Estimated absolute frequencies of the 5 most active A-numbers over time for Dataset A. A-numbers are suffixed with *-1* for configuration 1 (dash lines), *-2* for configuration 2 (solid lines), and *-3* for configuration 3 (dot lines)

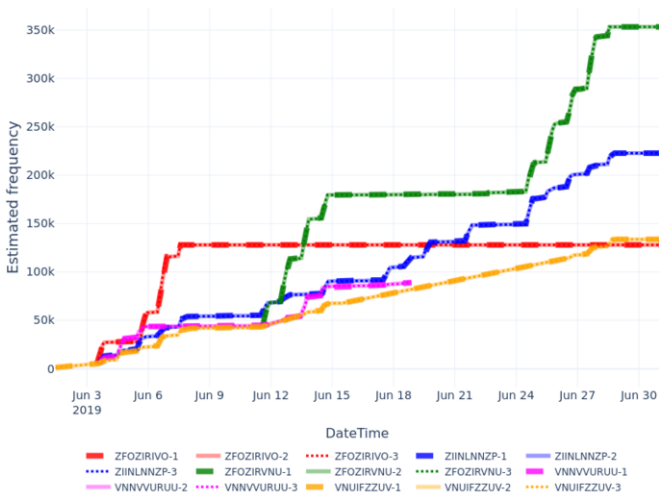


**Figure 3.** Estimated absolute frequencies of the 5 most active A-numbers over time for Dataset B. A-numbers are suffixed with *-1* for configuration 1 (dash lines), *-2* for configuration 2 (solid lines), and *-3* for configuration 3 (dot lines)

telecommunications traffic and can be vital for anomaly detection systems designed to flag fraudulent activities.

Similarly, Figure 3 presents the activity of the top-5 *A-numbers* in *Dataset B*. Given that *Dataset B* covers a shorter period (one month), the visualization can provide insights into more condensed activity patterns. Comparing these patterns against those observed in *Dataset A* might reveal significant differences in usage or highlight specific numbers that consistently exhibit high activity, which could be subject to further investigation for fraud or other anomalous behaviours.

Both figures are instrumental in assessing the efficacy of the heavy hitter detection system in real-world applications, especially in identifying potential fraudulent activities within the telecommunications sector. The visualization helps stakeholders quickly grasp significant trends and anomalies without delving into more complex data analysis, facilitating faster decision-making and response times to potential threats.

## 5.3 *Federated Approach*

In the federated approach, Datasets A and B were partitioned to simulate a multi-client environment. We used random distribution, whereby each row from the complete dataset is randomly allocated to a specific client.

Following the implementation of the multi-client environment in Flower, we implemented the federated Lossy Counting approach as detailed in Section 4. Each client executes the Lossy Counting algorithm, and the server aggregates the rankings from each client. We also apply three parameter configurations:

- Configuration 1: $epsilon = 0.001$, $support = 0.0005$
- Configuration 2: $epsilon = 0.001$, $support = 0.00025$
- Configuration 3: $epsilon = 0.001$, $support = 0.000125$

In all configurations, we consistently used a bucket size of 1000 examples, adjusting the minimum support to match the corresponding settings of the centralized approach. This standardization allows for a direct comparison between the centralized and federated methodologies within a controlled experimental framework.

When attempting to achieve similar counting results between centralized and federated approaches, it is important to consider the parameters' aggregate impact across all federated system nodes. Here are some considerations for aligning the parameters:

- Bucket Size and $\epsilon$: The bucket size is inversely proportional to epsilon. If the federated approach sums results from two clients, to approximate the bucket size used in the centralized approach, we would need to adjust the epsilon in the federated approach to be half of that in the centralized system.
- Support: The support threshold determines the minimum frequency an item must have to be considered a heavy hitter. In a federated system, if the data is evenly split and we are summing the results, we might keep the support value the same as the centralized approach because the summed frequencies from the federated nodes would theoretically equal the total frequency in the centralized system.

Given these considerations, we wanted that the federated approach to have a similar sensitivity as the centralized approach with an initial bucket size of 2000 (assuming $\epsilon = 0.0005$), so we adjusted the parameters of the federated system so that when combined, they mirror the centralized system's settings. For each of the two clients, we used $epsilon = 0.001$ – i.e. twice the centralized $\epsilon$ since each client sees half the data – and $support = 0.0005$ – same as centralized because the frequencies summed will be equivalent to the centralized dataset. Additionally, the data distribution, while statistically similar over time, may not be identical in each partition, leading to differences in the heavy hitters identified by the federated approach.

As illustrated in Table 2, the federated approach identified a number of distinct numbers very similar to the centralized method. The

**Table 2.** Number of A-numbers identified as heavy hitters with Federated Lossy Counting

| Dataset | Number of clients | | |
|---|---|---|---|
| | 2 | 4 | 8 |
| A - 3 months | 23 | 22 | 22 |
| B - 1 month | 22 | 20 | 20 |

percentage of common numbers in Dataset A is, on average, 86.36%, while the percentage of common numbers in Dataset B is, on average, 100%. These slight variations might include unique heavy hitters specific to the data subsets but do not appear as significant when the entire dataset is considered collectively in the centralized method. The federated approach operates on smaller samples in each client, potentially making it more sensitive to minor fluctuations in data. These fluctuations might be averaged out or obscured in the larger dataset used in the centralized approach. This dilution effect can cause the number of detected heavy hitters to decrease as the anomaly or heavy hitter's signal strength is not strong enough across smaller datasets.

Figure 4 shows the comparison between the frequencies of phone calls of a centralized approach (denoted as -C) and a federated approach (denoted as -F) of *Dataset A*, and Figure 5 illustrates the same for *Dataset B*. In both plots, we constructed a subset of the results for the *Configuration 1* of both approaches, comprising the top-5 *A-numbers* identified by the centralized version over buckets. The main observation is that, the federated version estimates the same frequencies as the centralized version for the same top-k *A-numbers*.

In Figure 4, the results reveal notable differences between the two implementations regarding frequency estimations. The centralized version, depicted by solid lines, consistently shows higher and more stable frequency estimations. In contrast, the federated version, represented by dashed lines, exhibits variations in growth patterns and sometimes lower frequency counts. These discrepancies suggest that the federated Lossy Counting implementation may exhibit different behaviour in frequency tracking compared to the centralized

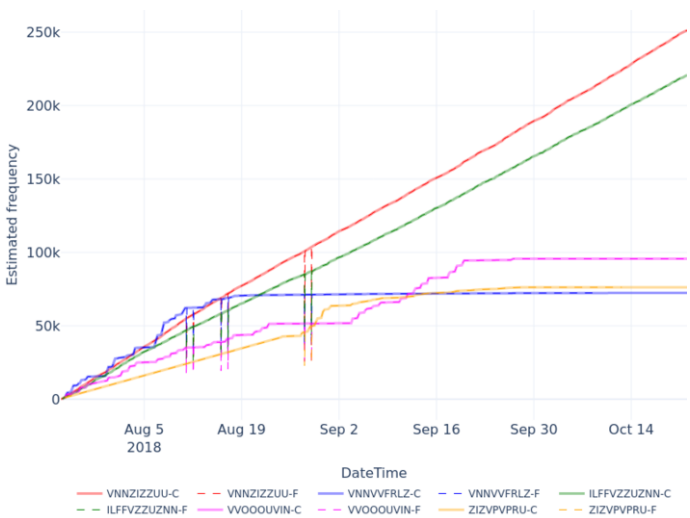


**Figure 4.** Estimated frequencies of the top-5 A-numbers (as identified by the centralized version) in Dataset A. Frequencies of A-numbers are suffixed with *-C* (solid lines) for the centralized version, and suffixed with -F (dash lines)for the federated version. Note that since we pick the numbers identified by the centralized Lossy Counting, all 5 A-numbers are not always within those identified by the federated version.

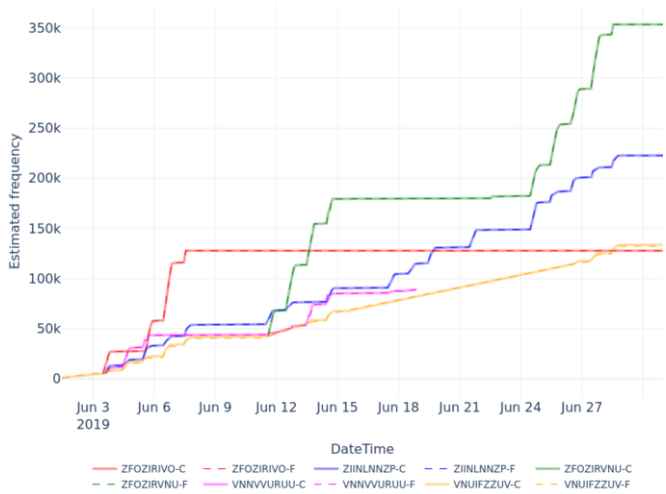


**Figure 5.** Estimated frequencies of the top-5 A-numbers (as identified by the centralized version) in Dataset B. Frequencies of A-numbers are suffixed with *-C* (solid lines) for the centralized version, and suffixed with -F (dash lines) for the federated version. Note that since we pick the numbers identified by the centralized Lossy Counting, all 5 A-numbers are not always within those identified by the federated version.

approach, particularly when capturing the most frequent items. This comparison highlights the potential variations in frequency estimation outcomes between federated and centralized implementations of the Lossy Counting algorithm, emphasizing decentralization's impact on identifying high-frequency elements. It is important to note that this consistent behaviour is observed not only in Configuration 1 but also in Configurations 2 and 3.

In Figure 5, one notable aspect of the results is that the centralized and federated approaches achieve the same frequency estimations for the top five A-numbers in Dataset B. The solid lines representing the centralized version and the dashed lines representing the federated version overlap precisely, indicating that both approaches yield identical results in estimating the frequencies of these A-numbers. Across all configurations, the centralized and federated implementations demonstrate the same frequency estimations, underscoring the robustness of the federated approach in *Dataset B*.

Given the characteristics of Lossy Counting and the context of the centralized versus federated approaches, several factors could explain why the federated version identified top-k numbers with lower frequencies in *Dataset A* but achieved the identical frequency estimations as the centralized version in the *Dataset B*:

- Partitioned Data: Since the federated approach divides the dataset into $n$ parts, each data node operates on a smaller subset of the total data. This partitioning could identify heavy hitters that might not appear as frequently in the combined dataset but are significant within their local subsets.
- Data Distribution Sensitivity: Even though data distribution is theoretically the same – given the random distribution –, in practice, slight changes in each node's distinct subset of data could lead to identifying different top-k items. The Lossy Counting algorithm is sensitive to the data distribution in each subset, which may cause certain items to appear as heavy hitters locally, even if they are not in the overall dataset.
- Error Bound and Frequency Threshold: The error in the counts from the Lossy Counting algorithm is within a certain threshold

related to the window size. In a federated setup, each node operates with its local data, potentially resulting in different error margins that can affect the detection of top-k items. Smaller datasets (as in each federated node) could have a smaller error bound, allowing the detection of items with lower global frequencies but which are significant on a local level.
- Combining Results: In the federated approach, when counts from individual nodes are aggregated, it is possible for items that are less frequent globally but frequent in a partition to make it to the combined top-k list, especially if they appear just below the frequency threshold.

The nature of the federated approach could capture a more diverse set of top-k items because it accounts for local variations that are not visible in the centralized dataset. The federated approach's capacity to detect local patterns of significance that may not emerge in a centralized analysis can be advantageous in distributed data environments. It allows for a finer-grained understanding of the data, revealing patterns specific to subsets within the whole.

The code is available in the repository [1].

## 6 Conclusions and Future Works

This manuscript has explored the application of the Lossy Counting algorithm within both centralized and federated learning frameworks for detecting heavy hitters in telecommunications data. Our investigation revealed that while both approaches effectively identify significant patterns within data streams, the federated approach demonstrates notable strengths in scenarios requiring heightened privacy and data security. Our findings indicate that the federated system identifies a broader array of heavy hitters thanks to its localized data processing. This approach captures data variations that could be overlooked or diminished in a centralized system.

The federated version of Lossy Counting, by processing data locally at each node and only sharing aggregated information, potentially enhances data privacy and security, making it suitable for sensitive data environments. While beneficial, the federated approach presents challenges, such as the complexity of managing and synchronizing multiple clients and ensuring consistent detection thresholds across diverse data segments. The detection sensitivity varied significantly across different configurations and setups, indicating that parameter tuning is critical for optimizing performance in diverse operational scenarios.

Future research could focus on refining the Lossy Counting algorithm's parameterization within a federated framework to enhance consistency and reliability of heavy hitter detection across varied network conditions. Extending the application of federated heavy hitter detection to other domains such as finance, healthcare, and e-commerce could provide deeper insights into its versatility and efficiency. Developing more sophisticated data aggregation methods could mitigate the challenges of variability and enhance the overall accuracy of the federated learning models.

## Acknowledgements

This research supported by the Fundação para a Ciência e Tecnologia (FCT), Portugal for the PhD Grant 2022.12896.BD.

[1] https://github.com/paularaissa/FederatedHeavyHitterDetection

## References

[1] U. Ahmed, G. Srivastava, and J. C. Lin. A federated learning approach to frequent itemset mining in cyber-physical systems. *J. Netw. Syst. Manag.*, 29(4):42, 2021.
[2] S. Bagui and R. P. Stanley. Mining frequent itemsets from streaming transaction data using genetic algorithms. *J. Big Data*, 7(1):54, 2020.
[3] N. R. Barroso, D. Jiménez-López, M. V. Luzón, F. Herrera, and E. Martínez-Cámara. Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges. *Inf. Fusion*, 90:148–173, 2023.
[4] E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, and A. H. Celdrán. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Commun. Surv. Tutorials*, 25(4):2983–3013, 2023.
[5] V. Bruno, M. Carlos, E. Raphael, S. Paula Raissa, A. Raul, and G. João. Anonymised Phone Call Dataset for Anomaly Detection, Aug. 2024. URL https://doi.org/10.5281/zenodo.13254389.
[6] Y. Chen, W. Gan, Y. Wu, and P. S. Yu. Privacy-preserving federated mining of frequent itemsets. *Inf. Sci.*, 625:504–520, 2023.
[7] G. Cormode and M. Hadjieleftheriou. Finding the frequent items in streams of data. *Commun. ACM*, 52(10):97–105, 2009.
[8] X. A. Dimitropoulos, P. Hurley, and A. Kind. Probabilistic lossy counting: an efficient algorithm for finding heavy hitters. *Comput. Commun. Rev.*, 38(1):5, 2008.
[9] P. Goyal, J. S. Challa, S. Shrivastava, and N. Goyal. Anytime frequent itemset mining of transactional data streams. *Big Data Res.*, 21:100146, 2020.
[10] X. Hei, X. Yin, Y. Wang, J. Ren, and L. Zhu. A trusted feature aggregator federated learning for distributed malicious attack detection. *Comput. Secur.*, 99:102033, 2020.
[11] J. Li, W. Gan, Y. Gui, Y. Wu, and P. S. Yu. Frequent itemset mining with local differential privacy. In *CIKM*, pages 1146–1155. ACM, 2022.
[12] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. *Proc. VLDB Endow.*, 5(12):1699, 2012.
[13] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, and A. Bifet. River: machine learning for streaming data in python. *Journal of Machine Learning Research*, 22(110):1–8, 2021. URL http://jmlr.org/papers/v22/20-1380.html.
[14] A. Moon, S. W. Son, H. Kim, and M. Kim. Lossy predictive models for accurate classification algorithms. In *IEEE Big Data*, pages 4576–4582. IEEE, 2022.
[15] B. Veloso, J. Gama, C. Martins, R. Espanha, and R. Azevedo. A case study on using heavy-hitters in interconnect bypass fraud. *ACM SIGAPP Applied Computing Review*, 20(3):47–57, 2020.
[16] B. Veloso, C. Martins, R. Espanha, R. Azevedo, and J. Gama. Fraud detection using heavy hitters: a case study. *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020. doi: 10.1145/3341105.3373842.
[17] B. Veloso, S. Tabassum, C. Martins, R. Espanha, R. Azevedo, and J. Gama. Interconnect bypass fraud detection: a case study. *Ann. des Télécommunications*, 75(9-10):583–596, 2020.
[18] S. Wang, G. Zhang, P. C. Sheu, M. Hayakawa, H. Shigematsu, and A. Kitazawa. Lossy graph data reduction. *Int. J. Semantic Comput.*, 12(3):425–456, 2018.
[19] Z. Wang, Y. Zhu, D. Wang, and Z. Han. Fedfpm: A unified federated analytics framework for collaborative frequent pattern mining. In *INFOCOM*, pages 61–70. IEEE, 2022.
[20] J. M. Wu, Q. Teng, S. Huda, Y. Chen, and C. Chen. A privacy frequent itemsets mining framework for collaboration in iot using federated learning. *ACM Trans. Sens. Networks*, 19(2):27:1–27:15, 2023.
[21] O. Wu, Y. S. Koh, G. Dobbie, and T. Lacombe. PEARL: probabilistic exact adaptive random forest with lossy counting for data streams. In *PAKDD (2)*, volume 12085 of *Lecture Notes in Computer Science*, pages 17–30. Springer, 2020.
[22] W. Xiao and J. Hu. Sweclat: a frequent itemset mining algorithm over streaming data using spark streaming. *J. Supercomput.*, 76(10):7619–7634, 2020.