DML-LBF

Project presentation!

aditya - heer - lavanya - pratham

Dynamic Multi-Level Learned Bloom Filter with Entropy Checking

- Real-world data is non-uniform and changes over time, demanding adaptive filtering and dynamic change.
- Entropy measures data skewness, guiding resource allocation to overloaded partitions and precise pathways.
- Neural models and entropy work together to minimize mispredictions and false positives for the membership question.
- Cascading Bloom filters enable fast, multi-stage rejection of non-members reducing the query time significantly.
- Dynamic scaling adjusts filter sizes and thresholds as data evolves.
- DML-LBF unifies learning, entropy, and cascading for robust, efficient membership testing.
- We aim that this holistic approach should achieve ultra-low FPR and memory efficiency in dynamic environments.

Standard Bloom Filters:

Traditional Bloom filters efficiently check membership but suffer from high false positive rates and static configurations

Prationed Bloom Filters:

Partitioned Bloom Filters split the bit array among hash functions to reduce false positives but struggle with dynamic datasets.

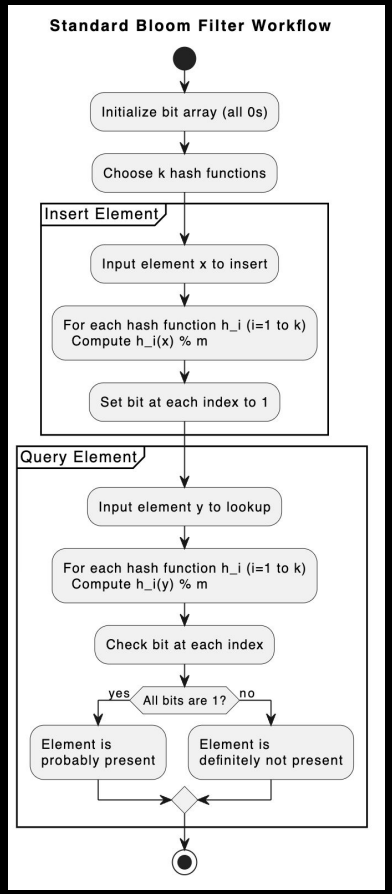
Cascading Bloom Filters:

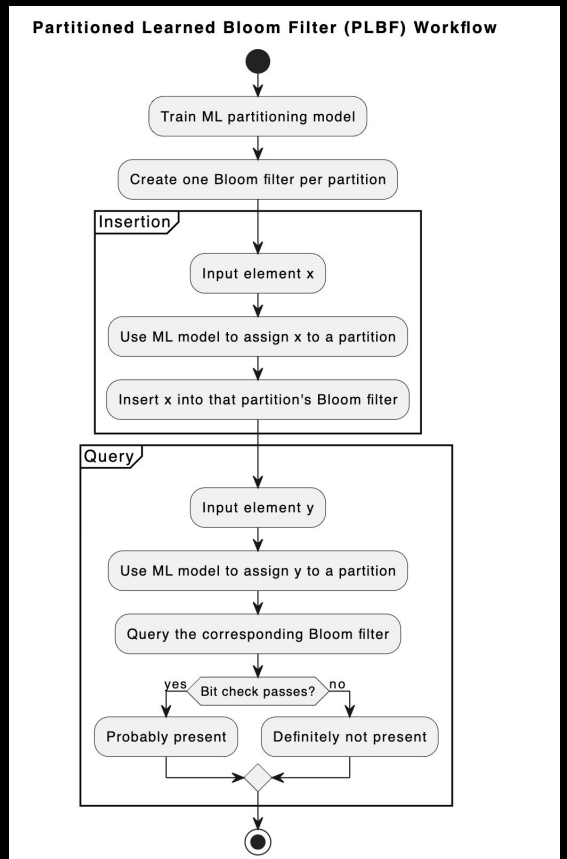
Cascading Bloom Filters use multiple filters to reduce false positives but increase query time and require careful tuning.

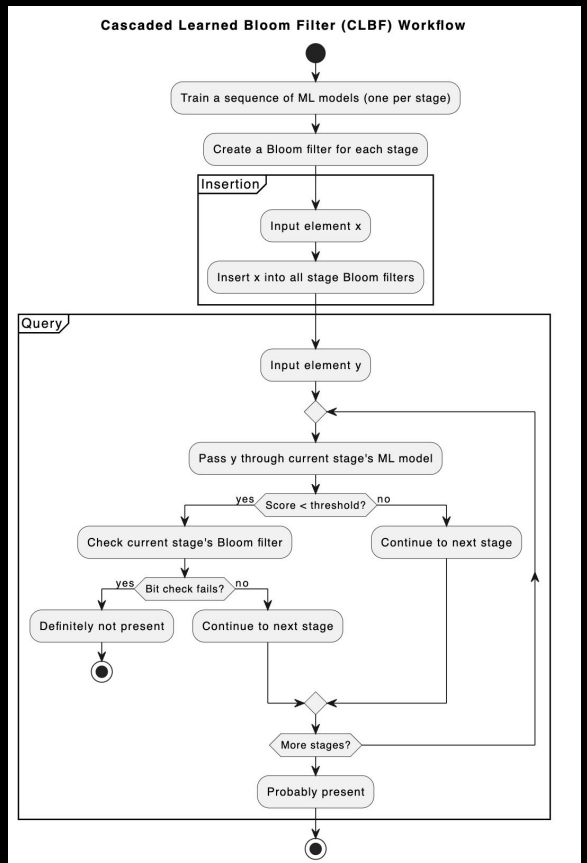
Learned Bloom Filters:

Learned Bloom Filters (LBFs) improve memory efficiency but struggle with skewed data distributions and lack adaptability.

DML-LBF/**system-architecture**





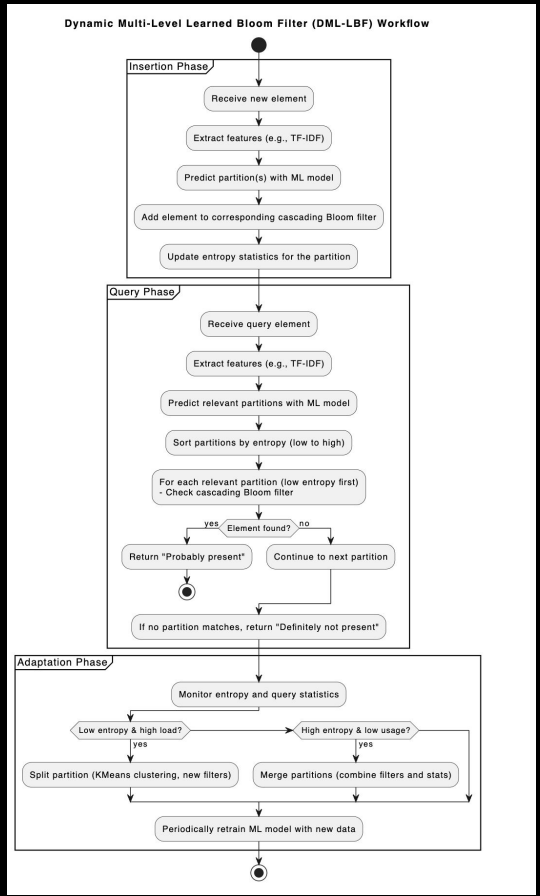


The proposed system, Dynamic Multi-Level Learned Bloom Filter with Entropy Checking (DML-LBF), combines learned models, partitioning, cascading Bloom filters, adaptivity, and entropy checking into a single, cohesive system.

Novel Features of the approach:

- **Entropy-aware optimization:** Uses entropy to check skewed part that needs precise checks while uniform partitions requiring minimal resources.
- **Adaptive learned model:** The model is adaptive to the upcoming queries and updates the threshold as and when partitioned are changed.
- **Dynamic scaling (real-time partitioning/merging):** Dynamically merges or splits the partitions and hash functions based on entropy response.
- **Integrated cascading and partitioning:** Fuses partitioning and cascading filters in the pipeline for coarse-to-fine check.

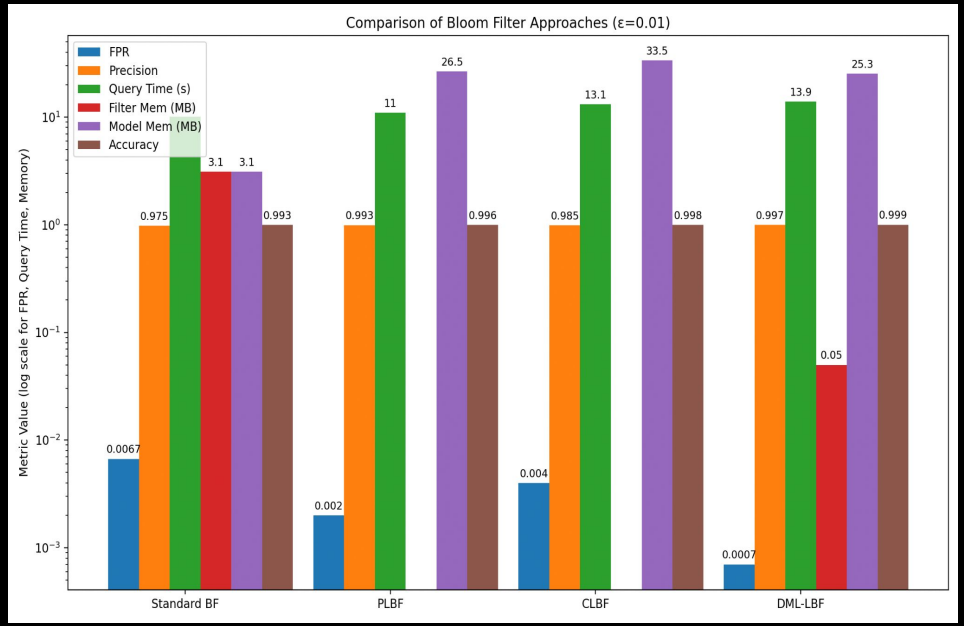
DML-LBF/**system-architecture**



- **Dataset Used:** Malicious URL Dataset (Kaggle Link)
- **Dataset Information:** 651,191 URLs, out of which 428103 benign or safe URLs, 96457 defacement URLs, 94111 phishing URLs, and 32520 malware URLs.
- **Train-Test Split:** 60% positive + 100% negative data into Insertions(Train) and 40% positive + negative data for Query(Test).
- **Evaluation Metrics Used:**
  - False Positive Rate (FPR)
  - Memory Usage (Model + Filter)
  - Accuracy
  - Precision
  - Query Latency
- **Baseline for Comparison:**
  - Standard BF
  - Partition LBF
  - Cascaded LBF
- **Other important threshold:**
  Adaptive Partition Threshold: 0.2 and 0.4 for split and merge
- **Model used:**
  - Sklearn’s Logistic Regressor and Random Forest (when on CPU)
  - PyTorch’s Logistic Regressor (when GPU enabled via Kaggle)

Results (for error rate = 0.01)

MetricP	Standard BF	PLBF	CLBF	DML-LBF
FPR	0.0067	0.0020	0.0040	0.0007
Accuracy	0.9933	0.9980	0.9960	0.9993
Precision	0.9751	0.9925	0.9851	0.9972
Query Time	10.10s	10.97s	13.13s	13.94s
Filter Memory	3.10 kB	~0.00 MB	~0.01 MB	0.05 MB
Total Memory	3.10 kB	26.48 MB	33.49 MB	25.37 MB



Current Limitations:

- Results are analysed on partial data (10,000 URLs out of ~6L) due to lack of Computational Resources
- The implementation of advanced Bloom filter variants such as Partitioned Learned Bloom Filter (PLBF) and Cascaded Learned Bloom Filter (CLBF) presents significant challenges due to their limited availability in standard libraries. While basic Bloom filters are commonly available in various programming languages, these more sophisticated learned variants require custom implementation.

Learnings:

- Entropy + learned models enhance filtering accuracy in skewed datasets
- Dynamic scaling is effective for adapting to evolving data
- Manual implementation of advanced Bloom filters deepened understanding
- Even limited data can reveal meaningful performance insights
- Resource constraints highlight the need for lightweight, efficient solutions

ThankYou

Questions?

[1] Liu, X., Zhang, Y., & Wang, Z. (2018). A Novel Cascading Bloom Filter Approach for Efficient Query Processing. Information Sciences.

[2] Chen, L., Kumar, S., & Li, H. (2023). Design and Evaluation of Adaptive Bloom Filters for High-Speed Networks. IEEE Transactions on Networking.

[3] Cormode, G., & Muthukrishnan, S. (2010). Cascading Bloom Filters: A Hierarchical Approach for Efficient Storage. IEEE Transactions on Knowledge and Data Engineering.

[4] Su, C.-C. (n.d.). Learned Sketch. Retrieved [Month Day, Year]

[5] Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018). The Case for Learned Index Structures. In Advances in Neural Information Processing Systems

[6] Rae, J., et al. (2019). Learned Bloom Filters: An Adaptive Approach to Membership Testing.
In Proceedings of the 36th International Conference on Machine Learning (ICML).