A Appendix

A.1 Background: Packed-Memory Arrays

This section provides background on a common list labeling data structure, ***the packed-memory array (PMA)***, introduced by Bender et al. [4]. The data structure in [4] extends the LLAs of [30, 54, 55] by adding lower-density thresholds. Lower-density thresholds provide the added guarantee that any two elements in the array are $\Theta(1)$ slots apart. Both versions guarantee an amortized cost of $O(\log^2 n)$. For simplicity, we describe the PMA using only upper density thresholds.

Classic PMA. Consider an array with $m = cn$ slots, for a constant $c > 1$. Divide the array into ranges of size $\Theta(\log n)$. These ranges form the ***leaves*** of an implicit binary tree on top of the $\Theta(n/\log n)$ leaves. Each node in this implicit binary tree corresponds to a subarray containing all leaf ranges within its subtree.

For a node in the implicit binary tree, define the ***size(u)*** as number of slots in its subarray and ***density(u)*** as the number of elements in its subarray divided by size(u). Let the ***depth*** of the root node be 0, nodes and leaf nodes be at depth $d = \Theta(\log n/\log\log n)$.

For each node u at depth k, let τ_k be the **density threshold**. Let τ_0 and τ_d be constants such that $0 < \tau_0 < \tau_d < 1$. For a node at depth k, let $\tau_k = \tau_0 + (\tau_d - \tau_0) \cdot \frac{k}{d}$.

A node u is ***within threshold*** if density$(u) \leq \tau_k$. If a node is within threshold, then all its descendants are also within threshold.

To insert an element x, determine the leaf range r it belongs to. If the leaf range r is within threshold, there is always a slot for x. Insert x in its slot and rebalance the r by evenly distributing all elements. If the leaf range is out of threshold, find the first ancestor r_a of the r that is within threshold and rebalance all elements evenly in that range. Now r is within threshold and there is room to insert x.

To see why the amortized cost of insertion is $O(\log^2 n)$, consider the insertion of element x that causes a node u at depth k to be rebalanced. Then, a child v of u must be out of threshold, that is, density$(v) > \tau_{k+1}$. After we rebalance u, both v (and its sibling) have density at most τ_k (density threshold of its parent u). The node u would need to rebalanced again when either v (or its sibling) go out of treshold again. This requires at least $(\tau_{k+1} - \tau_k) \cdot$ size(v) additional insertions. A rebalance of node u costs size(u). Thus, the amortized cost of rebalancing u is:

$$\frac{\text{size}(u)}{(\tau_{k+1} - \tau_k)\text{size}(v)} = \frac{2}{\tau_{k+1} - \tau_k} = \frac{2d}{\tau_d - \tau_0} = O(\log n)$$

Since an insertion can contribute to $O(\log n)$ ancestors being out of balance, the overall amortized cost of an insertion is $O(\log^2 n)$.

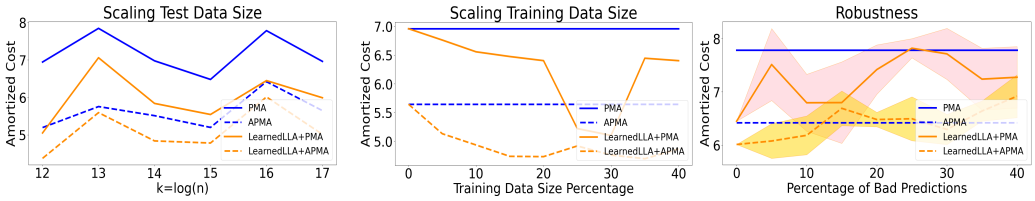
A.2 Additional Experiments

In this section, we further describe the experimental setup and the datasets we use. We also present more experimental results.

Experimental setup. We use a machine with 11th Gen Intel Core i7 CPU 2.80GHz, 32GB of RAM, 128GB NVMe KIOXIA disk drive, and running 64-bit Windows 10 Enterprise to run our experiments. We remark that amortized cost, the average number of element movements, is hardware-independent. We use the following density thresholds for the PMA and APMA: root's lower threshold: 0.2, leaves' lower threshold: 0.1, root's upper threshold: 0.5, leaves' upper threshold: 0.9. We add a $-\infty$ element at the beginning of each experiment. This is to make sure the internal predictor data structure in APMA operates as expected from the beginning of the experiment (see [7]). The datasets we use might include duplicate elements, and we use the same algorithm to insert these elements, even though in Section 2, we assume the elements in the input sequence form a set. This does not affect any of the algorithms and they are still well-defined. The relative order between duplicates can be arbitrary. In LearnedLLA, when we insert an element x, to find the black box LLAs containing the predecessor and successor of x, we use Python Sorted Containers library[5] (note that since we only measure the amortized cost, our results are independent of the function used to find these LLAs). For

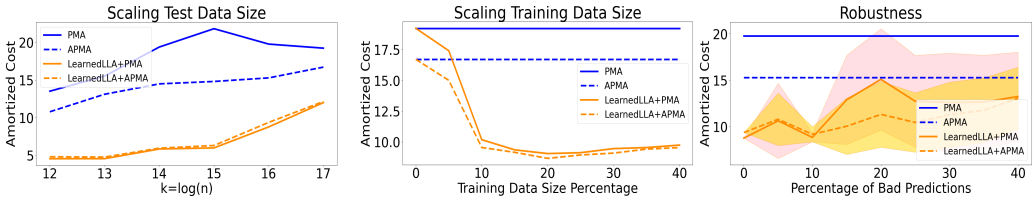
[5] https://grantjenks.com/docs/sortedcontainers/

measuring the amortized cost in the experiments, we do not count the first assignment of a label to an element as a relabel (note that this is in contrast to the theory section of the paper).



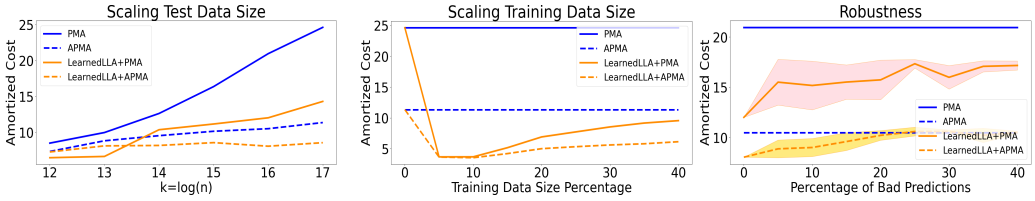
(a) (b) (c)

Figure 2: Gowalla (LocationID)



(a) (b) (c)

Figure 3: MOOC



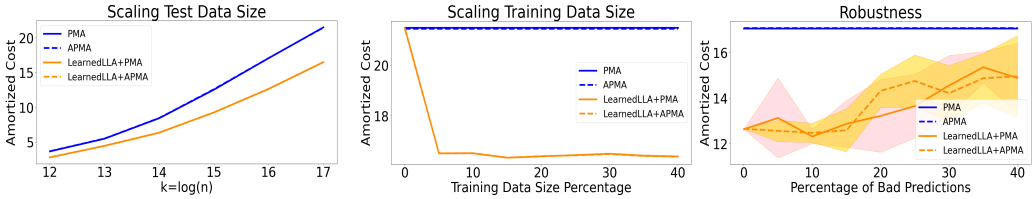
(a) (b) (c)

Figure 4: AskUbuntu

Dataset description. Here we describe the real temporal datasets we use in our experiments. In all cases, we use a prefix of the dataset in temporal order as the input sequence.

- Gowalla[6] [14]: Gowalla is a location-based social networking website where users share their locations by checking in. We use the location ID and latitude of the users that check in.
- MOOC[7] [36]: The MOOC user action dataset represents the actions taken by users on a popular MOOC platform. The actions are represented as a directed, temporal network. The nodes represent users and course activities (targets), and edges represent the actions by users on the targets. We use the user IDs as our input sequence.
- AskUbuntu[8]: This is a temporal network of interactions on the stack exchange web site Ask Ubuntu. There are three different types of interactions represented by a directed edge (u, v, t): i. user u answered user v's question at time t, ii. user u commented on user v's question at time t, and iii. user u commented on user v's answer at time t. We use the IDs of target users in the answers-to-questions network as the input sequence.
- email-Eu-core[9] [44] The network was generated using email data from a large European research institution. The e-mails only represent communication between institution members (the core), and the dataset does not contain incoming messages from or outgoing messages to the rest of the world. A directed edge (u, v, t) means that person u sent an e-mail to

[6] https://snap.stanford.edu/data/loc-Gowalla.html
[7] https://snap.stanford.edu/data/act-mooc.html
[8] https://snap.stanford.edu/data/sx-askubuntu.html
[9] https://snap.stanford.edu/data/email-Eu-core-temporal.html



(a) (b) (c)

Figure 5: email-Eu-core

person v at time t. A separate edge is created for each recipient of the e-mail. We use the IDs of target users as our input sequence.

Results. In Figures 2 to 5, we show plots for other datasets in Table 1. The setup is similar to Figure 1.

Discussion. These results further support our conclusions. Note that in some cases, increasing the size of the training set results in slightly worse performance for LearnedLLA. We believe this is because as we increase the size of the training data, we use older data as training.