Streaming Stochastic Submodular Maximization with On-Demand User Requests

Honglian Wang[1] Sijing Tu[1] Lutz Oettershagen[2] Aristides Gionis[1]

[1]KTH Royal Institute of Technology

[2]University of Liverpool

November 6, 2025

Homogeneous recommendations

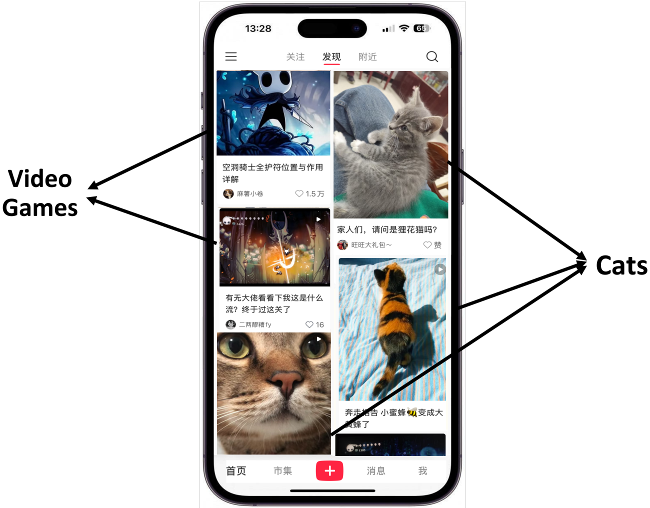


Figure: My Little Red Book first page

Our Problem

- **Streaming setting**: Items arrive in a streaming way
- **Objective**: Maximize the expected number of topics users are exposed to.

				visit 1			*visit 2*			*visit 3*	
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Input	Items stream										...
	Probabilities	0.5	0.1	0.6	0.5	0.8	0.5	0.4	0.5	0.5	
	Visits	0	0	1	0	0	1	0	0	1	
Output ($k = 2$)		$\emptyset$	$\emptyset$	$\{1, 3\}$	$\emptyset$	$\emptyset$	$\{5, 6\}$	$\emptyset$	$\emptyset$	$\{7, 8\}$	

Challenges

- **On-demand**: Users access at any time; system provides instant response
- **Irrevocable**: Recommended content cannot be modified once shown
- **Efficient**: Low memory usage ($\ll \Theta(N)$) and fast response

visit 1 *visit 2* *visit 3*

		1	2	3	4	5	6	7	8	9	
Input	Items stream										…
	Probabilities	0.5	0.1	0.6	0.5	0.8	0.5	0.4	0.5	0.5	
	Visits	0	0	1	0	0	1	0	0	1	
Output ($k = 2$)		$\emptyset$	$\emptyset$	$\{1, 3\}$	$\emptyset$	$\emptyset$	$\{5, 6\}$	$\emptyset$	$\emptyset$	$\{7, 8\}$	

Problem formulation

Assume the user visits for T times, the objective is to find $\mathcal{S}_1, \mathcal{S}_1, \cdots, \mathcal{S}_T$, such that the expected number of covered topics is maximized.

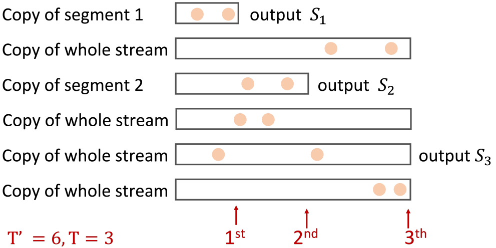
$$
\begin{aligned}
\max_{\mathcal{S}=\mathcal{S}_1\cup\ldots\cup\mathcal{S}_T} \quad & f(\mathcal{S}) = \sum_{j=1}^{d} \left(1 - \prod_{t=1}^{T} \prod_{\substack{V_i \in \mathcal{S}_t \\ V_i \ni c_j}} (1 - p_i) \right) \\
\text{s.t.} \quad & \mathcal{S}_t \subseteq \text{ news items received before the } t\text{-th visit}, \quad \forall t \in [T], \\
& |\mathcal{S}_t| \leq k, \quad \forall t \in [T].
\end{aligned}
$$

where V_i represents item i and p_i its associated probability, c_j denotes topics item j covers.

$f(\mathcal{S})$ is a submodular function.

Our algorithm STORM

- **Assumption:** Assume the user visits for at most T' times.
- **Approach:** We design our algorithm by utilizing the *streaming submodular maximization problem subject to a partition matroid constraint* .
  - Create T' partitions and initialize them as active
  - Copy each incoming item to all active partitions
  - Upon a user's visit, output the result of the active partition that yields the highest marginal gain, and deactivate the selected partition
- **Theoretical guarantee:** STORM achieves a competitive ratio of $\frac{1}{4(T'-T+1)}$



Chekuri, Chandra, Shalmoli Gupta, and Kent Quanrud. "Streaming algorithms for submodular function maximization." International Colloquium on Automata, Languages, and Programming. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

Improved algorithm STORM++

- STORM $++$ makes multiple guesses of T: $\Delta, 2\Delta, \ldots, \lceil \frac{T'}{\Delta} \rceil \Delta$
- STORM $++$ runs one parallel instance of the STORM with each guess as an input
- STORM $++$ greedily selects the best result among all the results returned by the parallel instances, upon each visit.

$\Rightarrow$ STORM $++$ achieves a competitive ratio of $1/(8\Delta)$.

$\mathrm{T} = 3,\ \mathrm{T}' = 6, \Delta = 2.$ STORM++ make guesses of $\mathrm{T} \in \{2,4,6\}$

Guess of T/ partitions	2	4	6	Storm++ output
1	A_2^1	A_4^1	A_6^1	$G_1 = \mathrm{argmax}_j f(A_2^1, A_4^1, A_6^1)$
2	A_2^2	A_4^2	A_6^2	$G_2 = \mathrm{argmax}_j f(A_2^2, A_4^2, A_6^2)$
3		A_4^3	A_6^3	$G_3 = \mathrm{argmax}_j f(A_4^3, A_6^3)$

A_j^i denotes the i-th output of the STORM algorithm with guess $T = j$.

Results

Table: Comparison of our algorithms. $N \in \mathbb{N}$ denotes the stream length, $k \in \mathbb{N}$ number of news items to present per user access, $T \in \mathbb{N}$ and $T' \in \mathbb{N}$ are the exact number and upper bound of user accesses, and $\Delta \in \mathbb{N}$ an approximation parameter.

Algorithm	**Competitive ratio**	**Time**	**Space**	**Response time**
LMGreedy	$1/2$	$\mathcal{O}(NT'k)$	$\Theta(N + kT)$	$\mathcal{O}(Nk)$
Storm	$1/4(T' - T + 1)$	$\mathcal{O}(NT'k)$	$\mathcal{O}(T'k)$	$\mathcal{O}(T')$
Storm++	$1/8\Delta$	$\mathcal{O}(NT^{'2}k/\Delta)$	$\mathcal{O}(T^{'2}k/\Delta)$	$\mathcal{O}(T')$

For more experimental results, please refer to our paper.

Open Problems

1. How do we obtain item click probability p_i?
2. Is it possible to integrate multiple metrics into the current framework? For instance, can we combine completion rate, watch time, engagement rate, dwell time, CTR, and CVR within the same optimization objective?

Thank you