Finding Top-k Most Frequent Items in Distributed Streams in the Time-Sliding Window Model

Emmanuelle Anceaume, Yann Busnel, Vasile Cazacu

Finding Top-k Most Frequent Items in Distributed Streams in the Time-Sliding Window Model

Emmanuelle Anceaume
CNRS, IRISA
Rennes, France
Email: emmanuelle.anceaume@irisa.fr

Yann Busnel
IMT Atlantique, IRISA
Cesson-Sévigné, France
Email: yann.busnel@imt-atlantique.fr

Vasile Cazacu
CNRS, IRISA
Rennes, France
Email: vasile.cazacu@irisa.fr

I. Introduction and problem statement

From marketing over social networks to prevention of distributed denial of service (DDoS) attacks, the need to analyze in real time large-scale and distributed data streams has recently became tremendous. Among several important problems raised in this context, the need to dynamically detect heavy-hitters (or *hot*) items during the most recent time window is essential but highly challenging. The problem of finding the most frequent items has been heavily studied during the last decades with both exact and probabilistic solutions [1], [2]. Nevertheless, answering this issue over a sliding time window is still an active research field [3]. Very recently, Song *et al.* [4] formalized this problem as the Windowed Top-k Frequent Items (WTK) problem and proposed an efficient and very elegant solution, named the Floating Top-K (FTK) method, to solve WTK. We improve upon their solution by providing a new algorithm, that we call FTK$_{CE}$, which is based on a deterministic counting of the most over-represented items in the data streams, which are themselves identified probabilistically. Performances (both in accuracy and memory cost) are astonishingly good, despite an adversary whose objective is to manipulate the order in which items are received at the nodes.

System model. We consider a set of $\mathcal{N}$ nodes $S_1, \ldots, S_{\mathcal{N}}$ such that each node S_i receives a large sequence σ_{S_i} of data items or symbols. We assume that streams $\sigma_{S_1}, \ldots, \sigma_{S_{\mathcal{N}}}$ do not necessarily have the same size, *i.e.*, some of the items present in one stream do not necessarily appear in others or their occurrence number may differ from one stream to another one. We also suppose that node S_i ($1 \leq i \leq \mathcal{N}$) does not know the length of its input stream. Items arrive regularly and quickly, and due to memory constraints (*i.e.*, nodes can locally store only a small amount of information with respect to the size of their input stream and perform simple operations on them) need to be processed sequentially and in an online manner. Let $\sigma = a_1, a_2, a_3, \ldots, a_n$ be a stream of data items that arrive sequentially. Each data item i is drawn from the universe $\Omega = \{1, 2, \ldots, N\}$, where N is very large. A natural approach to study a data stream σ of length n is to model it as a fingerprint vector over the universe Ω, given by $X = (x_1, x_2, \ldots, x_N)$ where x_i represents the number of occurrences of data item i in σ.

Problem statement. The windowed Top-k frequent items definition is borrowed from [4]. Specifically,

Definition 1 (Windowed Top-k Frequent Items [4]): Let t be the current time unit. The *Windowed Top-k Frequent Item problem* consists in returning the k most frequent items for any given time window from time unit $t - w$ to time unit t, denoted as $[t - w, t]$, with $w \leq W$, W being a user-defined upper bound of w. More formally, we seek the set $Top_k = \{i \in \Omega \mid x_i \geq x_j$ where x_j is the k-th greatest value in $X\}$. This query may be asked at any time unit t.

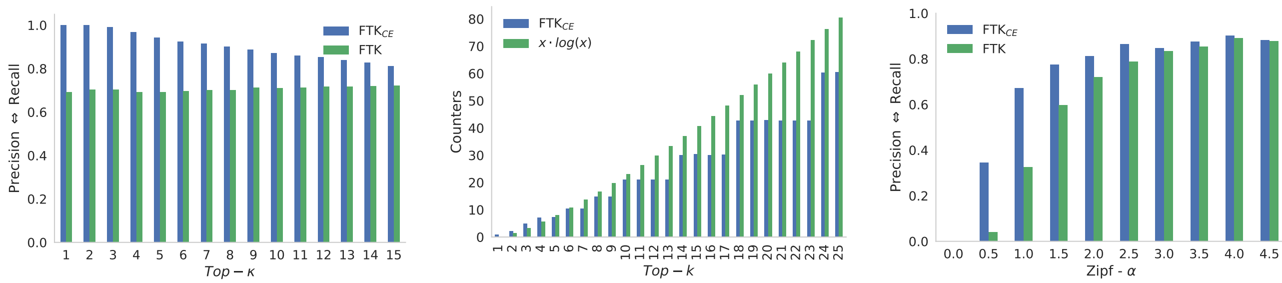
Moreover, the time domain is partitioned into *time units*, which define the granularity of window sliding ("jumping"). Indeed, the time unit acts like a micro-batch for the WTK problem. The top-k most frequent items are generated for each time unit, after what the top-κ most frequent items over the whole past window is returned. Moreover, window's length $w \leq W$ and number $\kappa \leq k$ of items requested are freely chosen by the user at execution time.

II. Algorithm description

Our algorithm FTK$_{CE}$ is based on two steps allowing respectively *(i)* to select the relevant candidates to integrate the Top_k set and *(ii)* to rank these candidates to answer the problem whatever the value of $\kappa \leq k$ provided.

Step 1: Probabilistic selection is inspired by the use of the primitive RandomLevel proposed in [4]. This primitive associates to any item received a random *level* value computed on the fly, as follow. At each call of RandomLevel, the algorithm starts by throwing a coin, and with probability p gets "head", and with probability $1-p$ gets "tail". This process continues as long as "head" are returned. Once a "tail" is returned, the primitive returns the number of "head" obtained for this item. The random variable *level* follows a geometric distribution. Given a predetermined threshold θ, this item is then considered as a potential candidate if $level \geq \theta$. Else, it is simply ignored and do not enter to the next step.

Step 2: The second step of our algorithm is the counting phase. It uses a buffer memory Γ_t whose size is dynamic. For each candidate item $i \in \Omega$ selected in the first step, if item i already has a c_i counter in Γ_t, this one is simply incremented. Otherwise, a new $(i, 1)$ counter is added to Γ_t. The set of all tuples of a unit of time t are grouped into a data structure



(a) (b) (c)

Fig. 1. Simulation results obtained with 10^6 executions of both FTK$_{CE}$ and FTK fed with input streams of $n = 10^6$ items made of $N = 10^4$ distinct items.

Γ. This structure brings together subsets from Γ_t to Γ_{t-W} arranged in reverse chronological order.

Intuitively, if an item is very frequent, its probability of exceeding θ increases due to the large number of calls to RANDOMLEVEL primitive. In comparison, the chance for a rare item to be a candidate is very low. Thus, the expectation for item i of being a candidate is statistically proportional to its frequency x_i. Counting how many times an item is candidate allows us to obtain a global ranking which is, in expectation, accurate with the weight of the heavy hitters in X.

Query: When requesting the Top_κ set (with $\kappa \leq k$) on a window of size $w \leq W$, the algorithm just sums, for each item i of Γ, all the tuples corresponding to i in the different sub-structures Γ_τ (for $\tau \in \{t-w, \ldots, t\}$). The set of items with the κ highest global counter value is then returned.

The challenging aspect of FTK$_{CE}$ is to properly choose θ. θ must minimize the number of candidates while maximizing the probability for the heavy hitters to be retained. We rely on a former analysis of the coupon collector problem [5, Formula (5)] and an analysis of a leader election problem [6, Th. 4.1] to compute θ. Anceaume *et al.* [5] analysis allows us to compute the expected number of items to be drawn in order to collect the Top_k items we are interested in. We apply next a leader election result from [6], which allows us to determine θ as the expected number of rounds to elect the expected number of items to be drawn.

III. PERFORMANCE EVALUATION

Our theoretical analysis shows us that FTK$_{CE}$ returns the same Top_k items when applied directly over the whole time window or over per time units. This is of utmost importance when considering a malicious environment. Briefly, any ordering manipulation of the input data stream during any time window has no impact on the returned top-k items. The only feature that influences the probability of error of our algorithm is the total items frequency during the time window. This comes from the random and independent level attribution schema, which by construction selects each item independently of each other, and thus independently of the order in which items are received.

In the remaining of the paper, we present a summary of the experiments we have conducted to compare performances of our algorithm with the one of Song *et al*. [4], which is so far, and to the best of our knowledge, the most impressive solution to solve the windowed top-k problem. Note that these experiments illustrate the nice theoretical properties we are still conducting on our algorithm. Performances of our algorithm are denoted by "FTK$_{CE}$" on the graphs, and the ones of Song *et al.* [4] are denoted by "FTK" for "Floating Top-k". Figure 1a compares the precision of both FTK$_{CE}$ and FTK algorithms, fed with a Zipf-α distribution of items, with $\alpha = 2$, when queried to provide the top-κ, with $k = 15$. By precision, we mean the number of top-i, with $1 \leq i \leq k$, correctly detected by the algorithms divided by the total number of detected items. A particularity of the Top-k problem is that precision and recall are equivalent in this case. Indeed, the number of false positives in the returned Top_κ set corresponds exactly to the number of false negatives not returned by the algorithm. Figure 1b represents the total number of counters used in both algorithms. Note that we have plotted the $x \cdot log(x)$ function as Song *et al.* [4] argue in their paper that this should represent a lower memory bound of their algorithm. Figure 1c compares the precision of both solutions as a function of different Zipf-α distributions of items (from a uniform one, *i.e.*, $\alpha = 0$, to a strongly skewed one, *i.e.*, $\alpha = 4.5$). FTK$_{CE}$ is capable of detecting frequent items even for very flat item distributions (*i.e.*, Zipf-0.5).

To conclude, we have briefly shown the impressive behavior of our algorithm. We are still analyzing the theoretical behavior of our algorithm (*i.e.*, (ϵ, δ)-approximation and bounds on the memory cost), and we conducting extensive simulations in a particularly adversarial environment.

REFERENCES

[1] A. Metwally, D. Agrawal, and A. El Abbadi, "Efficient computation of frequent and top-k elements in data streams," in *Proceedings of the 10th International Conference on Database Theory (ICDT)*, 2005.
[2] E. Anceaume, Y. Busnel, N. Rivetti, and B. Sericola, "Identifying global icebergs in distributed streams," in *Proceedings of the 34th IEEE Symposium on Reliable Distributed Systems (SRDS)*, 2015, pp. 266–275.
[3] Z. Wei, G. Luo, K. Yi, X. Du, and J.-R. Wen, "Persistent data sketching," in *Proceedings of the 2015 ACM International Conference on Management of Data (SIGMOD)*, 2015.
[4] C. Song, X. Liu, and T. Ge, "Top-k frequent items and item frequency tracking over sliding windows of any sizes," in *Proceedings of the 2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 2017.
[5] E. Anceaume, Y. Busnel, and B. Sericola, "New results on a generalized coupon collector problem using markov chains," *Journal of Applied Probability*, vol. 52, no. 2, pp. 405–418, 2015.
[6] R. Kalpathy, H. M.Mahmoud, and W. Rosenkrantz, "Survivors in leader election algorithms," *Statistics & Probability Letters*, vol. 83, no. 12, pp. 2743 – 2749, 2013.