L'art d'extraire des éléments du top-k en temps réel sur des fenêtres glissantes réparties

Emmanuelle Anceaume, Yann Busnel, Vasile Cazacu

L'art d'extraire des éléments du top-k en temps réel sur des fenêtres glissantes réparties

Emmanuelle Anceaume[1], Yann Busnel[2] et Vasile Cazacu[1]

[1]*CNRS, IRISA, F-35042 Rennes, France*
[2]*IMT Atlantique, IRISA, F-35700 Rennes, France*

Avec la prolifération des appareils connectés (smartphones, capteurs, *etc.*), de plus en plus de sources de flux de données émettent des informations en temps réel avec des fluctuations du débit d'entrée et de la distribution des valeurs au fil du temps. Le traitement de ces flux tout en respectant certaines contraintes de qualité de service (QoS) soulève des problèmes de Big Data (variété et rapidité) dans un contexte temps réel.

Le besoin de collecter l'information a des fins de supervision est désormais devenu essentiel. Dans un objectif de passage a l'échelle, l'échantillonnage avisé semble être une solution pertinente pour fournir une brique de base de ces applications. Nous considérons dans cet exposé le problème de l'échantillonnage des données dans les systèmes à grande échelle en présence d'un adversaire puissant, en temps réel, dans le contexte de fenêtres glissantes.

Nous proposons dans un cet article un algorithme probabiliste permettant d'extraire les récents éléments fréquents du *top-k*, dans des flux de données distribués. Cet algorithme améliore significativement la fiabilité et la précision des résultats de la littérature, en réduisant fortement l'empreinte mémoire nécessaire sur chacun des nœuds participants à la résolution du problème.

Travaux co-financés par le projet BigClin du Labex CominLabs (ANR-10-LABX-07-01).

1 Introduction et présentation du problème

Du marketing aux réseaux sociaux en passant par la prévention des attaques par déni de service distribué (DDoS), la nécessité d'analyser en temps réel des flux de données distribués et à grande échelle est récemment devenue essentielle. Parmi plusieurs problèmes importants soulevés dans ce contexte, la nécessité de détecter dynamiquement les objets les plus fréquents au cours de la période la plus récente est essentielle mais très difficile. Ce problème a été fortement étudié au cours des dernières décennies avec des solutions à la fois exactes et probabilistes [MAEA05, ABRS15]. Néanmoins, répondre à cette question dans le cadre d'une fenêtre glissante est toujours un domaine de recherche actif [WLY+15]. Très récemment, Song, Liu & Ge *et al.* [SLG17] ont formalisé ce problème sous le nom de problème des éléments fréquents en fenêtres glissantes (*Windowed Top-k Frequent Items – FTK*) et ont proposé une solution efficace et très élégante, appelée la méthode FTK (*Floating Top-k*), pour résoudre ce problème. Notre solution améliore cette dernière en fournissant un nouvel algorithme, dénommé FTK$_{CE}$. Celui-ci est fondé sur un comptage déterministe des éléments les plus sur-représentés dans les flux de données, lesquels sont identifiés de manière probabiliste. Les performances (tant en précision qu'en coût mémoire) sont particulièrement bonnes, malgré la présence d'un adversaire dont l'objectif est de manipuler l'ordre dans lequel les éléments sont reçus.

Modèle du système. Nous considérons un ensemble de $\mathcal{N}$ nœuds $S_1, \ldots, S_{\mathcal{N}}$, tels que chaque nœud S_i reçoive une grande séquence σ_{S_i} de données ou de symboles. Nous supposons que les flux $\sigma_{S_1}, \ldots, \sigma_{S_{\mathcal{N}}}$ n'ont pas forcément les mêmes caractéristiques, c'est-à-dire, certains des éléments présents dans un flux n'apparaissent pas nécessairement dans d'autres ou leur nombre d'occurrences peut différer d'un flux à l'autre. Nous supposons également que le nœuds S_i ($1 \leq i \leq \mathcal{N}$) ne connaissent pas la longueur totale de leur flux d'entrée. Les éléments arrivent régulièrement et rapidement, et en raison de contraintes de mémoire (*i.e.*, les nœuds ne peuvent stocker localement qu'une petite quantité d'informations par rapport à la taille

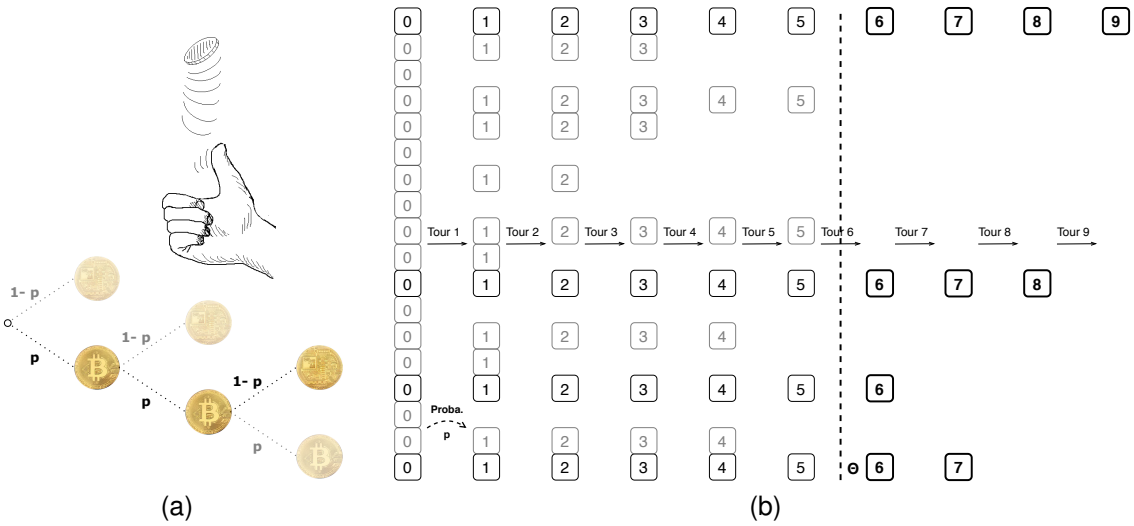

FIGURE 1: Illustration des deux étapes de notre algorithme FTK$_{CE}$.

de leur flux d'entrée et effectuer des opérations simples sur celles-ci), ils doivent être traités de manière séquentielle et en ligne.

Soit $\sigma =< a_1, a_2, a_3, \ldots, a_n >$ un flux de données qui arrivent séquentiellement. Chaque donnée i est tirée de l'univers $\Omega = \{1, 2, \ldots, N\}$, où N est potentiellement très grand. Une approche naturelle pour étudier un flux de données σ de longueur n est de le modéliser comme un vecteur de fréquence sur l'univers Ω, défini par $X = (x_1, x_2, \ldots, x_N)$ où x_i représente le nombre d'occurrences de la donnée i dans σ.

Énoncé du problème. La définition des éléments top-k les plus fréquents sur fenêtre glissante est empruntée à [SLG17]. Plus précisément :

Définition (Windowed Top-k Frequent Items (WTK) [SLG17])**.** *Soit t l'instant courant. Le problème WTK consiste à renvoyer les k éléments les plus fréquents sur une fenêtre temporelle donnée, s'étendant du temps t au temps $t-w$, dénoté $[t-w, t]$, avec $w \leq W$, W étant une borne supérieure de w, définie par l'utilisateur. Plus formellement, nous recherchons l'ensemble*

$$Top_k = \{i \in \Omega \mid x_i \geq x_j \text{ où } x_j \text{ est la } k\text{-ème plus grande valeur de } X\}.$$

Le domaine temporel est divisé en unités de temps, ce qui définit la granularité de glissement des fenêtres. En effet, l'unité de temps agit comme un micro-batch pour le problème WTK. Les éléments du top-k les plus fréquents sont générés pour chaque unité de temps, après quoi les éléments du top-k les plus fréquents sur l'ensemble de la fenêtre glissante sont retournés. La longueur de la fenêtre $w \leq W$ et le nombre $\kappa \leq k$ des éléments demandés sont librement choisis par l'utilisateur au moment de l'exécution. Cette requête peut être demandée à tout instant t.

2 Description de l'algorithme

Notre algorithme FTK$_{CE}$ se décompose en deux étapes permettant respectivement *(i)* de sélectionner les candidats pertinents pour intégrer l'ensemble top-k et *(ii)* de classer ces candidats pour répondre au problème quelque soit la valeur de $\kappa \leq k$ fournie.

Étape 1 : La sélection probabiliste s'inspire de l'utilisation de la primitive RANDOMLEVEL proposée dans [SLG17]. Cette primitive associe à n'importe quel objet reçu une valeur de *niveau* aléatoire, calculée à la volée, comme suit. À chaque appel de RANDOMLEVEL, l'algorithme effectue un tirage de *pile-ou-face*, lequel obtient *face* avec une probabilité p, et *pile* avec une probabilité $1-p$. Ce processus se poursuit aussi longtemps que l'algorithme obtient *face*. Lorsque le premier *pile* est retourné, la primitive retourne

le nombre de *face* obtenu pour cet élément. La variable aléatoire *niveau* suit une distribution géométrique. Compte tenu d'un seuil prédéterminé θ, cet élément est alors considéré comme un candidat potentiel si *niveau* $\geq \theta$. Sinon, il est simplement ignoré et l'étape suivante ne sera pas exécutée pour celui-ci. La figure 1a illustre le tirage effectué pour un élément dont la valeur *niveau* retournée vaut 2.

Étape 2 : La deuxième étape de notre algorithme est la phase de comptage. Il utilise une mémoire tampon Γ_t dont la taille est dynamique. Pour chaque item candidat $i \in \Omega$ sélectionné dans la première étape, si l'élément i a déjà un compteur c_i dans Γ_t, celui-ci est simplement incrémenté. Sinon, un nouveau compteur $(i,1)$ est ajouté à Γ_t. L'ensemble de tous les tuples d'une unité de temps t sont regroupés dans une structure de données Γ. Cette structure rassemble les sous-ensembles de Γ_t à Γ_{t-W} classés dans l'ordre chronologique inverse. La figure 1b représente les 4 éléments sélectionnés (potentiellement redondants) grâce au seuil θ, fixé à 6 dans cet exemple.

Intuitivement, si un élément est très fréquent, sa probabilité de dépasser θ augmente en raison du grand nombre d'appels à la primitive RANDOMLEVEL. En comparaison, la probabilité qu'un objet rare soit candidat est très faible. Par conséquent, l'espérance pour l'item i d'être un candidat est statistiquement proportionnelle à sa fréquence x_i. Compter le nombre de fois qu'un objet est candidat nous permet d'obtenir un classement global fiable (estimateur non biaisé), considérant le poids des éléments fréquents dans X.

Requête : Lors de la demande de l'ensemble top-κ (avec $\kappa \leq k$) sur une fenêtre de taille $w \leq W$, l'algorithme additionne simplement, pour chaque élément i de Γ, tous les tuples correspondant à i dans les différentes sous-structures Γ_τ (pour $\tau \in \{t-w,\ldots,t\}$). L'ensemble des κ éléments ayant les valeurs de compteurs ainsi calculées les plus élevées est alors retourné.

Le défi majeur de FTK$_{CE}$ est de bien déterminer θ. θ doit minimiser le nombre de candidats tout en maximisant la probabilité que les éléments fréquents soient retenus. Nous nous appuyons sur une analyse antérieure du problème des collectionneurs de coupons [ABS15, Equation (5)] et une analyse d'un problème d'élection d'un leader [KMR13, Théorème 4.1] pour calculer θ. L'analyse de Anceaume, Busnel et Sericola [ABS15] nous permet de calculer le nombre minimum d'éléments à récolter afin de recueillir l'ensemble des items top-k qui nous intéressent. Nous appliquons ensuite le résultat de l'élection de leader de Kalpathy, Mahmoud & Rosenkrantz [KMR13], ce qui nous permet de déterminer θ comme étant le nombre prévu de tours nécessaire à l'élection du nombre d'éléments à tirer, estimé précédemment. Le lecteur intéressé par les preuves formelles de notre algorithme pourra se référer à la version complète de cet article [ABC18].

3 Évaluation des performances

Notre analyse théorique nous démontre que FTK$_{CE}$ retourne les mêmes éléments top-k lorsqu'il est appliqué soit directement sur toute la fenêtre temporelle, soit sur des unités de temps séparément. C'est d'une importance capitale lorsque l'on considère un environnement malveillant. En bref, toute manipulation du flux d'entrée pendant n'importe quelle fenêtre temporelle n'aura pas d'impact sur les éléments retournés dans l'ensemble top-k. La seule caractéristique qui influence la probabilité d'erreur de notre algorithme est la fréquence totale des items pendant la fenêtre temporelle. Cela provient du schéma d'attribution aléatoire et indépendant qui, par construction, sélectionne chaque élément indépendamment les uns des autres, et donc indépendamment de l'ordre dans lequel les éléments sont reçus.

Dans le reste de l'article, nous présentons un résumé des expérimentations que nous avons menées pour comparer les performances de notre algorithme avec celles de Song *et al.* [SLG17], qui est jusqu'à présent, et à notre connaissance, la meilleure solution pour résoudre le problème WTK. Il est à noter que ces résultats d'expérimentation illustrent les propriétés théoriques intéressantes que nous poursuivons sur notre algorithme.

Les performances de notre solution sont identifiés par "FTK$_{CE}$" sur les figures, et celles de Song *et al.* [SLG17] par "FTK". La figure 2a compare la précision des algorithmes FTK$_{CE}$ et FTK alimentés par une distribution Zipf, de paramètre $\alpha = 2$, lors d'une requête des top-κ, avec $\kappa \leq k = 15$. Par précision, nous entendons le nombre d'éléments de top-κ, avec $1 \leq \kappa \leq k$, correctement détectés par les algorithmes divisé par le nombre total d'items détectés. Une particularité du problème top-k est que la précision et le rappel

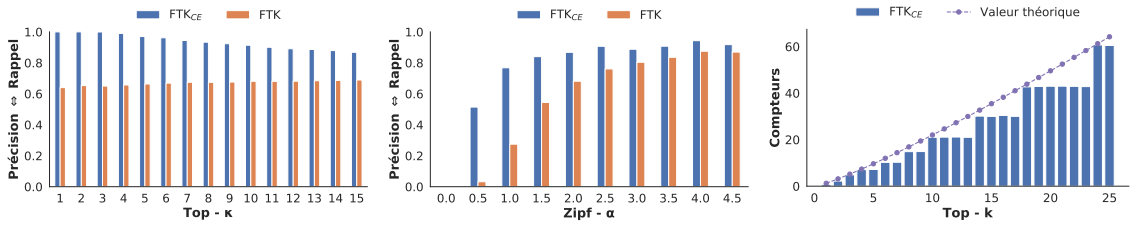

FIGURE 2: Résultats des simulations obtenus avec 10^3 exécutions de FTK$_{CE}$ et de FTK, sur des flux de données de taille $n = 10^6$ composés de $N = 10^4$ éléments distincts.

sont équivalents dans ce cas. En effet, le nombre de faux positifs dans l'ensemble top-k retourné correspond exactement au nombre de faux négatifs non retournés par l'algorithme.

La figure 2b compare la précision des deux solutions en fonction du paramètre α de la distribution Zipf (d'une distribution uniforme, *i.e.* $\alpha = 0$, à une distribution fortement biaisée, *i.e.* $\alpha = 4,5$). FTK$_{CE}$ est capable de détecter des éléments fréquents même pour des distributions faiblement biaisées (*cf.*, Zipf-$0,5$).

La figure 2c représente le nombre total de compteurs utilisés pour identifier les top-k éléments les plus fréquents avec, $k \leq 25$. Le nombre de compteurs correspond au nombre de candidats au top-k retenus lors de la première étape de notre algorithme. La forme en escaliers est due à l'arrondi à l'entier inférieur du seuil θ. Nous avons donc représenté également le nombre de compteurs avec la vrai valeur théorique de θ.

4 Conclusion

En conclusion, nous avons brièvement montré le comportement impressionnant de notre algorithme. Une version étendue de cet article est disponible pour le lecteur intéressé [ABC18]. L'analyse théorique complète de notre algorithme est toujours en cours (*i.e.*, (ε, δ)-approximation et les bornes asymptotiques du coût mémoire), et nous effectuons des simulations approfondies dans un environnement particulièrement conflictuel.

Références

[ABC18] E. Anceaume, Y. Busnel, and V. Cazacu. On the fly detection of the top-k items in the distributed sliding window model. In *Proc. of the 17th IEEE International Symposium on Network Computing and Applications (NCA 2018)*, pages 1–8, 2018.

[ABRS15] E. Anceaume, Y. Busnel, N. Rivetti, and B. Sericola. Identifying global icebergs in distributed streams. In *Proc. of the 34th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 266–275, 2015.

[ABS15] E. Anceaume, Y. Busnel, and B. Sericola. New results on a generalized coupon collector problem using markov chains. *Journal of Applied Probability*, 52(2) :405–418, 2015.

[KMR13] Ravi Kalpathy, Hosam M.Mahmoud, and Walter Rosenkrantz. Survivors in leader election algorithms. *Statistics & Probability Letters*, 83(12) :2743 – 2749, 2013.

[MAEA05] A. Metwally, D. Agrawal, and A. El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Proc. of the 10th International Conference on Database Theory (ICDT)*, 2005.

[SLG17] C. Song, X. Liu, and T. Ge. Top-k frequent items and item frequency tracking over sliding windows of any sizes. In *Proc. of the 2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 2017.

[WLY+15] Zhewei Wei, Ge Luo, Ke Yi, Xiaoyong Du, and Ji-Rong Wen. Persistent data sketching. In *Proc. of the 2015 ACM International Conference on Management of Data (SIGMOD)*, 2015.