Apprentissage en temps réel pour la collecte d'information dans les réseaux sociaux.

Thibault Gisselbrecht, Ludovic Denoyer, Patrick Gallinari, Sylvain Lamprier

Apprentissage en temps réel pour la collecte d'information dans les réseaux sociaux

Thibault Gisselbrecht[*], Ludovic Denoyer[**], Patrick Gallinari[**], Sylvain Lamprier[**]

Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
CNRS, UMR 7606, LIP6, F-75005, Paris, France
[*] *thibault.gisselbrecht@irt-systemx.fr* [**] *prénom.nom@lip6.fr*

RÉSUMÉ. *Dans cet article nous nous intéressons à la collecte d'information dans les réseaux sociaux. Cette tâche, primordiale pour de nombreuses applications, se heurte souvent à diverses contraintes liées aux ressources à disposition ou à des restrictions imposées par les API des médias considérés. Nous formulons cette tâche comme un problème de sélection dynamique de sources, pour lequel nous proposons une méthode d'apprentissage pour orienter la collecte vers les données les plus pertinentes en fonction d'un besoin spécifié. Notre méthode est basée sur une extension d'un algorithme de bandit combinatoire récemment proposé. Nous fournissons des garanties théoriques sur le comportement de l'algorithme, que nous évaluons ensuite sur différents jeux de données Twitter, à la fois hors ligne et en ligne, pour différents besoins de données exprimés.*

ABSTRACT. *We consider the problem of capturing information on social media under bounded resource. The latter may correspond to real time constraints such as response time limitation, limited computing resources, or social media API restrictions. We formulate this problem as a dynamic source selection problem. We then propose a machine learning methodology for dynamically selecting the most relevant information sources for a given information need. This method is based on an extension of a recently proposed combinatorial bandit algorithm. We provide theoretical guarantees on the behavior of the algorithm. We then evaluate the algorithm on different Twitter datasets for both offline and online settings.*

MOTS-CLÉS : *Apprentissage statistique, réseaux sociaux, bandit manchot*

KEYWORDS: *Machine Learning, Social Network, Multi-armed Bandit*

1. Introduction

Depuis leur apparition sur les réseaux il y a une dizaine d'années, les médias sociaux en ligne sont rapidement devenus des sources de données incontournables pour de nombreuses applications et services. La collecte des données qu'ils produisent constitue alors une question clé pour de nombreux acteurs, industriels et académiques. Afin de permettre le suivi de l'activité de leurs utilisateurs sur leur système, la plupart des médias sociaux actuels proposent un service de capture par flux de données (*Streaming*). Ce service permet l'acquisition en temps réel des données produites sur le média social considéré. Néanmoins, l'utilisation d'un tel service peut se heurter à diverses contraintes, aussi bien techniques - ressources de calcul / stockage disponibles limitées - que politiques - restrictions imposées par les médias sociaux pour protéger leurs données. La collecte temps réel de la totalité des données produites sur un média social est alors bien souvent impossible. Une stratégie consiste donc à définir des filtres permettant d'orienter la collecte vers des données correspondant à un besoin particulier. Il s'agit de sélectionner les sources de données à écouter (utilisateurs du réseau, catégories thématiques, etc), les plus susceptibles de produire des données pertinentes pour le besoin défini. Cet échantillonnage des sources, que proposent diverses compagnies de services spécialisées, est cependant difficile à mettre en place manuellement. Dans cet article, nous proposons une stratégie d'échantillonnage automatique, capable de s'adapter à des besoins de données particuliers et prenant en compte les limites opérationnelles des API proposées par les média sociaux.

Prenons l'exemple du célèbre réseau social Twitter qui produit plus de 7000 messages (*tweets*) par seconde. Être à même de consommer une telle quantité de données requiert des capacités de stockage et/ou de calcul très importantes, d'autant plus si l'on souhaite en extraire de l'information utilisable pour une tâche donnée. Par ailleurs, comme la plupart des médias sociaux, Twitter s'est rapidement rendu compte de la richesse des données qu'il possède, et ne permet plus aujourd'hui la capture de la totalité de son activité. Seules les données relatives à un nombre limité d'indicateurs (auteurs ou mots-clé contenus par exemple) peuvent être considérées simultanément, restreignant alors considérablement la connaissance du réseau à un sous-ensemble limité de son activité globale. Dans ce contexte, définir un besoin de données / informations peut s'avérer une tâche complexe : comment définir un ensemble statique d'indicateurs permettant une collecte efficace, alors que l'on ne connaît pas la distribution des données pertinentes sur le réseau ? D'autant plus dans un contexte dynamique ? Si une collecte concernant une thématique particulière peut se faire en définissant une liste de mots-clé spécifiques que doivent contenir les messages à récupérer, les données obtenues via cette méthode sont souvent très bruités ou hors sujet du fait du trop grand nombre de réponses ou d'interférences entre divers évènements. Les entreprises qui vendent des solutions d'accès aux données des réseaux sociaux connaissent bien cette problématique et beaucoup d'entre elles ont recours à l'intervention d'un opérateur humain pour définir et modifier les indicateurs permettant de filtrer les données à collecter, ce qui est onéreux et n'est pas envisageable à grande échelle.

Considérons un système de *streaming* qui, compte tenu d'un ensemble d'utilisa-

teurs sources à écouter, fournit le contenu produit par ces derniers pendant une période de temps spécifique. Étant donnée une fonction permettant d'évaluer la pertinence du contenu délivré par une source pour un besoin particulier, nous proposons une solution à ce problème d'échantillonnage de sources basée sur une méthode d'apprentissage automatique, à savoir une extension des algorithmes de bandit combinatoire. À partir d'un ensemble initial de sources, l'algorithme proposé permet d'explorer, d'évaluer et de redéfinir l'ensemble des sources à considérer. Cela permet d'apprendre progressivement à se concentrer sur les sources d'information les plus pertinentes du réseau, sous les contraintes spécifiées (capacité d'écoute simultanée). Cette méthode a l'avantage de fonctionner pour n'importe quel besoin, sous réserve que ce dernier puisse être exprimé sous la forme d'une fonction de qualité. Elle peut être utilisée par exemple pour collecter des messages d'actualité, identifier des influenceurs thématiques ou capturer des données qui tendent à satisfaire un panel d'utilisateurs finaux donnés.
Les contributions de l'article sont les suivantes :

– Nous proposons une nouvelle tâche d'apprentissage pour la collecte d'information en temps réel sur les réseaux sociaux ;

– Nous formalisons cette tâche comme un problème de bandit combinatoire ;

– Nous proposons une extension de l'algorithme CUCB (Chen *et al.,*, 2013) pour résoudre ce problème et présentons des garanties théoriques sur la convergence de cet algorithme ;

– Nous expérimentons notre méthode sur des données hors ligne et en ligne afin d'en montrer la capacité à orienter automatiquement la collecte vers les sources de données les plus susceptibles de produire des contenus pertinents.

Cet article est organisé comme suit : la section 2 introduit la tâche de collecte de données dynamique sur les réseaux sociaux. La section Section 3 décrit notre algorithme permettant d'optimiser cette collecte de données selon un critère de récompense spécifié. Dans la section 4, nous présentons les résultats des expériences nous permettant d'évaluer la performance de notre approche. Finalement, la section 5 présente les travaux connexes et discute des extensions possibles.

2. Problème d'apprentissage

Le problème de la collecte dynamique de données sur les réseaux sociaux peut donc se voir comme un problème de sélection de sources à écouter : n'ayant pas la capacité de considérer la totalité de l'activité du réseau à chaque instant de la collecte, l'objectif est de définir un sous-ensemble d'utilisateurs susceptibles de produire du contenu pertinent pour le besoin spécifié. Nous proposons de considérer ce problème dans le contexte des réseaux sociaux de grande taille, où le choix des sources ne peut être effectué manuellement en raison du trop grand volume de données produites et du trop grand nombre de sources à envisager. Par exemple, pour nos expérimentations (section 4), bien que notre approche pourrait être appliquée à bien d'autres réseaux sociaux (tous ceux qui offrent un accès temps réel à un sous-ensemble de leurs données),

nous nous intéressons à la collecte de données sur Twitter, où le nombre d'utilisateurs total est supérieur à 240 millions, pour une limite de 5000 utilisateurs pouvant être écoutés simultanément. Dans ce contexte, choisir quel sous ensemble d'utilisateurs suivre à chaque instant est une question complexe, qui requiert la mise en œuvre de techniques d'apprentissage permettant l'exploration de l'ensemble des utilisateurs du réseau pour déterminer de manière efficace les meilleurs candidats à écouter pour la tâche considérée.

Étant donné un processus qui, à chaque instant de la collecte, permet de récupérer les contenus produits par un sous ensemble d'utilisateurs d'un réseau social, et une fonction de qualité permettant de mesurer la pertinence d'un contenu publié par un utilisateur écouté dans un intervalle de temps, il s'agit de définir une stratégie de décision permettant au processus de se concentrer sur les utilisateurs les plus intéressants à mesure que le temps s'écoule. Cette stratégie, apprise de façon incrémentale, est définie de façon à maximiser le score cumulé - évalué via la fonction de qualité - par les utilisateurs écoutés au cours du temps. Dans ce qui suit, cette stratégie de décision est appelée *politique de sélection*. Sur Twitter par exemple, l'information capturée pourrait correspondre à l'ensemble des *tweets* publiés par les utilisateurs écoutés durant la période de temps considérée, et la fonction de qualité pourrait correspondre à un score évaluant le contenu par rapport à une thématique donnée ou bien sa popularité. Diverses fonctions de qualité sont proposées dans la partie 4.

Dans notre contexte, une difficulté majeure provient du fait que l'on ne connaît rien a priori sur les utilisateurs du réseaux, ni des relations qui peuvent les relier : récupérer des profils utilisateurs ou la liste des utilisateurs amis d'un utilisateur donné n'est bien souvent pas envisageable, car cela requiert le questionnement d'une API coûteuse ou restreinte (fréquence des requêtes possibles souvent très limitée) et se heurte parfois à des problèmes de confidentialité des données. Nous considérons donc dans cet article qu'aucune information autre que ce qui est obtenu via l'API de *Streaming* considérée n'est accessible. Cela implique entre autres que nous ne pouvons pas nous baser sur le graphe du média social pour explorer les utilisateurs (ce qui interdit l'emploi de techniques utilisées pour des problèmes connexes de *Crawling*, voir section 5) et que nous ne connaissons pas *a priori* l'ensemble complet des utilisateurs du réseaux.

Ainsi, en partant d'un ensemble d'utilisateurs initiaux, il s'agit de concevoir un processus alimentant l'ensemble des sources possibles, de façon incrémentale au fur et à mesure que de nouveaux utilisateurs sont rencontrés, afin d'orienter la collecte vers des zones non connues du réseau. Par exemple, un nouvel utilisateur peut être référencé dans un message si l'utilisateur à l'origine du message le mentionne dans son contenu. Typiquement, sur Twitter, les utilisateurs peuvent se répondre entre eux ou republier des messages d'autres utilisateurs (voir la section 4), ce qui nous offre la possibilité de découvrir de nouvelles sources sans utilisation de ressources externes. Le processus général, présenté dans la figure 1 peut être décrit de la façon suivante :

1) Sélection d'un sous ensemble d'utilisateurs relativement à la politique choisie ;

2) Écoute de ces utilisateurs sources pendant une fenêtre de temps donné ;

3) Alimentation de l'ensemble des sources possibles en fonction des nouveaux utilisateurs référencés dans les messages enregistrés ;

4) Évaluation des données collectées en fonction de leur pertinence pour la tâche à résoudre ;

5) Mise à jour de la politique de sélection en fonction des scores obtenus.

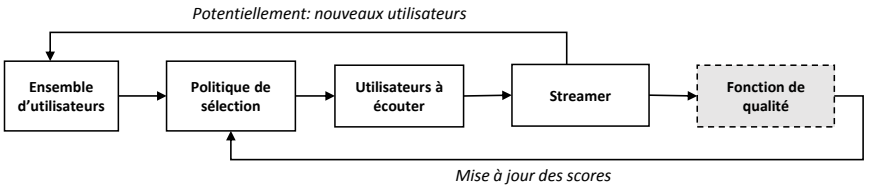

Figure 1. *Processus général de la capture de données*

3. Approche bandit

3.1. *Contexte : Problèmes de bandits*

Le problème du bandit traite du compromis entre exploration et exploitation dans un processus de décision séquentiel où, à chaque pas de temps, un agent doit choisir une action parmi un ensemble d'actions possibles. A l'issue de cette action l'agent reçoit une récompense qui quantifie la qualité de l'action choisie relativement à une fonction de qualité. Le but de l'agent est de maximiser son gain cumulé au cours du temps. Le terme bandit provient de l'expression "bandit-manchot" désignant une machine à sous. Imaginons un joueur en face d'un certain nombre de machines à sous, ayant la possibilité d'en jouer une chaque minute. Son but est évidement de maximiser la somme de ses gains. Afin d'améliorer sa connaissance de l'environnement, un compromis entre l'exploitation des bonnes machines à sous et l'exploration de nouvelles, ou peu connues, doit être défini. Ce compromis représente le point central de tous les algorithmes de bandit. Le problème de bandit est étudié dans (Auer *et al.,*, 2002). Une présentation d'ensemble peut être également trouvée dans (Bubeck *et al.,*, 2012).
Dans le cadre du bandit classique, une seule action est choisie à la fois, puis évaluée. Nous considérons ici le cas où l'agent peut sélectionner **plusieurs actions simultanément** à chaque pas de temps. Ce problème, connu sous le nom *bandit combinatoire*, a récemment été formalisé et étudié dans (Chen *et al.,*, 2013).

Notations : On note $\mathcal{K}$ l'ensemble des K actions et $\omega_{i,t} \in \Omega$ le résultat de l'action i au temps t. On suppose connue la fonction de qualité g, donnant un score $g(\omega_{i,t})$ pour chaque $\omega_{i,t}$. A chaque instant $t = 1, 2..., n$ l'agent doit donc :

– Choisir un sous-ensemble $\mathcal{K}_t \subseteq \mathcal{K}$ de k actions selon la politique π ;

– Observer le résultat $\omega_{i,t}$ $\forall i \in \mathcal{K}_t$ et recevoir la récompense $g(\omega_{i,t})$;

– Améliorer la stratégie de sélection grâce aux nouvelles observations.

L'objectif est donc de trouver la politique de sélection optimale π^* maximisant de gain cumulé dans le temps :

$$\pi^* = \arg\max_{\pi} \sum_{t=1}^{n} \sum_{i \in \mathcal{K}_t} g(\omega_{i,t}) \quad [1]$$

Nous supposons également que nous sommes dans le cas dit stochastique où chaque distribution de récompense suit une loi inconnue ν_i à valeurs dans $[0, 1]$: $g(\omega_{i,.}) \sim \nu_i$ avec une moyenne μ_i.

Algorithme CUCB : L'algorithme *CUCB* récemment proposé dans (Chen *et al.*,, 2013) est une extension de l'algorithme *UCB* original (Auer *et al.*,, 2002) adapté au cas ou plusieurs actions peuvent être choisies simultanément. Nous considérons le cas particulier où la récompense fournie par un ensemble d'actions correspond à la somme des récompenses des actions individuelles. Tout comme l'algorithme *UCB* classique, l'algorithme *CUCB* utilise un score de classement $v_{i,t}$ calculé pour l'action i au temps t. A chaque instant t, l'algorithme *CUCB* sélectionne les k meilleures actions selon ce score, tandis que l'algorithme *UCB* sélectionne uniquement la meilleure. Des garanties de convergence théoriques sont fournies dans (Chen *et al.*,, 2013). Notons $\tau_{i(t)}$ le nombre de fois où l'action i a été choisie pendant les t premiers pas de temps et $g_{i,s}$ la s^{ieme} récompense résultant de l'action i. Le score $v_{i,t}$ associé à l'action i au temps t est calculé en fonction de la moyenne empirique $\widehat{\mu}_{i,\tau_{i(t-1)}}$ des récompenses obtenues avec l'action i dans les $t-1$ premiers pas de temps, où $\widehat{\mu}_{i,x}$ est défini pour tout $i \in \{1..K\}$ et $x \in \{1..\tau_{i(n)}\}$ par :

$$\widehat{\mu}_{i,x} = \frac{1}{x} \sum_{s=1}^{x} g_{i,s} \quad [2]$$

Le score $v_{i,t}$ utilisé pour classer les utilisateurs à chaque instant dans (Chen *et al.*,, 2013) est défini par :

$$v_{i,t} = \widehat{\mu}_{i,\tau_{i(t-1)}} + B_{i,t} \quad [3]$$

où $B_{i,t} = \sqrt{\frac{3\ln(t)}{2\tau_{i(t-1)}}}$. Ce score représente bien un compromis entre l'exploitation (terme $\widehat{\mu}_{i,\tau_{i(t-1)}}$) et l'exploration (terme $B_{i,t}$), puisqu'il s'agit de la somme d'un premier terme estimant la qualité d'une action i et d'un second terme décroissant avec le nombre de fois où l'action i est choisie.

3.2. *Collecte de données ciblée grâce aux bandits*

Formalisation de la tâche comme un problème de bandit combinatoire :
Notons $\mathcal{U}$ l'ensemble des utilisateurs du réseau social étudié et divisons la période de *streaming* en n pas de temps. Le contenu produit par l'utilisateur i pendant la t^{ieme}

étape est noté $\omega_{i,t} \in \Omega$, où Ω est l'ensemble des contenus possibles selon les cas d'utilisations. Sur Twitter par exemple, $\omega_{i,t}$ correspond à l'ensemble des *tweets* postés par l'utilisateur i pendant l'intervalle de temps t. Nous considérons une fonction de qualité g donnant un score a un contenu :

$$g : \begin{cases} \Omega \to [0;1] \\ g(\omega) = \text{ qualité de } \omega \text{ pour la tâche en question} \end{cases} \quad [4]$$

Cette fonction de qualité dépend de la tâche à résoudre et peut par exemple correspondre à un modèle thématique ou un modèle de popularité (c.f. section 4). Comme mentionné précédemment, les API de *streaming* fournies par les réseaux sociaux permettent d'écouter un certain nombre k d'utilisateurs simultanément. Notre problème revient donc à un problème de bandit combinatoire avec k actions simultanées à chaque instant. Étant donnée une période de n pas de temps, un ensemble d'utilisateurs disponibles $\mathcal{K} \subseteq \mathcal{U}$ et une contrainte opérationnelle k, une politique de *streaming* est une fonction $\pi : \{1, ..., n\} \to \mathcal{K}^k$, où $\pi(t)$ définit pour un instant t un sous ensemble de k utilisateurs à écouter. Notre but est donc de trouver la meilleure politique π^*, comme définie dans l'équation 1, permettant de maximiser la somme des récompenses accumulées pendant toute la durée du processus.
Suivant une méthodologie similaire à *CUCB* présenté plus haut, mais appliquée à notre problème où l'ensemble complet des utilisateurs du réseau n'est pas connu a priori, nous proposons l'algorithme 1, qui définit un processus de collecte dynamique à partir d'un ensemble initial d'utilisateurs sources. A chaque itération, notre algorithme calcule un score pour tous les utilisateurs connus, les classe, sélectionne les k premiers, les écoute, enregistre les récompenses associées et alimente l'ensemble $\mathcal{K}$ avec les nouveaux utilisateurs.

Le score $v_{i,t}$ considéré pour chaque utilisateur i à l'instant t est défini par :

$$v_{i,t} = \begin{cases} \widehat{\mu}_{i,\tau_{i(t-1)}} + B_{i,t} & si\ \tau_{i(t-1)} > 0 \\ +\infty & si\ \tau_{i(t-1)} = 0 \end{cases} \quad [5]$$

où $B_{i,t}$ correspond au terme d'exploration. Comme $\mathcal{U}$ n'est pas connu, il est impossible d'initialiser la moyenne empirique de chaque utilisateur en les écoutant une fois au début du processus comme dans un problème de bandit classique. A l'instant t, nous définissons $v_{i,t} = \infty$ pour les i pour lesquels $\tau_{i(t-1)} = 0$. Cela force l'algorithme à écouter les nouveaux utilisateurs au moins une fois dans le but d'initialiser leur moyenne empirique.

***CUCBV* : Prise en compte de la variance** :
Dans notre cas, la récompense de chaque utilisateur est basée sur la pertinence du contenu produit pendant une période finie. Cependant, de fortes variations peuvent être observées sur la fréquence de publication des utilisateurs écoutés. Typiquement, la plupart du temps, les utilisateurs ne produisent aucun contenu. Avec la politique *CUCB* , le score utilisé dans l'équation 3 pénaliserait les utilisateurs produisant peu

de contenu les premières fois qu'ils sont écoutés. De plus, aucune différence ne serait faite entre un utilisateur produisant beaucoup de contenus de qualité moyenne et un utilisateur produisant peu de contenu mais d'une grande qualité. Pour prendre en compte cela, nous proposons un nouvel algorithme de bandit combinatoire appelé **Combinatorial UCBV (CUCBV)**, qui considère la variance des récompenses reçues. L'algorithme *CUCBV* est une extension de l'algorithme *UCBV* proposé dans (Audibert *et al.,*, 2007) au cas combinatoire. Ce dernier utilise la variance dans le terme d'exploration et semble mieux adapté à notre tâche. A notre connaissance, aucun algorithme prenant en compte la variance n'a été proposé dans le contexte du bandit combinatoire.
Le terme d'exploration $B_{i,t}$ de l'algorithme *UCBV* est défini par :

$$B_{i,t} = \sqrt{\frac{2\ln(t)\widehat{\sigma}^2_{i,\tau_{i(t-1)}}}{\tau_{i(t-1)}}} + \frac{3\ln(t)}{\tau_{i(t-1)}} \quad [6]$$

où $\widehat{\sigma}^2_{i,x}$ est la variance empirique de l'utilisateur i après sa x^{ieme} sélection :

$$\widehat{\sigma}^2_{i,x} = \frac{1}{x}\sum_{s=1}^{x}(g_{i,s} - \widehat{\mu}_{i,x})^2 \quad [7]$$

Avec un tel facteur d'exploration, la politique tend à explorer les utilisateurs ayant une grande variance, puisque plus d'informations sont nécessaires pour avoir une bonne estimation de leur qualité. La partie suivante discute des garanties de convergence théoriques pour notre algorithme.

Algorithm 1: Algorithme de streaming

Input: $\mathcal{K}$, k, n
for $t \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** K **do**
    Calculer $v_{i,t}$ avec la formule 5;
  end
  Classer les utilisateurs par ordre décroissant selon $v_{i,t}$;
  Sélectionner les k premiers pour fixer $\mathcal{K}_t$;
  for $i \in \mathcal{K}_t$ **do**
    Écouter i et observer $\omega_{i,t}$;
    Enregistrer la récompense $g(\omega_{i,t})$;
    Alimenter $\mathcal{K}$ avec les nouveaux utilisateurs rencontrés $j, j \notin \mathcal{K}$;
  end
end

Garanties théoriques pour l'algorithme CUCBV :
La performance des algorithmes de bandit est habituellement mesurée par la notion de regret, qui correspond à la perte de récompense qu'un agent est susceptible de subir en choisissant une action i au lieu d'une action optimale i^*. Dans notre contexte, si l'on

note $\mathcal{K}^*$ le sous ensemble de k actions fournissant les plus grandes valeurs moyennes de récompenses cumulées sur une période de n pas de temps, le regret cumulé R_n est défini par :

$$R_n = \sum_{t=1}^{n} \sum_{j \in \mathcal{K}^*} g(\omega_{j,t}) - \sum_{i \in \mathcal{K}_t} g(\omega_{i,t}) \quad [8]$$

Nous sommes donc intéressés par des politiques menant à de faibles valeurs de regret cumulé moyen.

La preuve de la proposition suivante est disponible à l'URL [1].

Proposition 1 *En considérant l'ensemble complet des actions $\mathcal{K}$ connu, nous obtenons, avec notre algorithme CUCBV et $n \in \mathbb{N}^+$:*

$$\mathbb{E}[R_n] \leq \ln(n) \sum_{i \notin \mathcal{K}^*} \left(C + 8(\frac{\sigma_i^2}{\delta_i^2} + \frac{2}{\delta_i}) \right) \Delta_i + D \quad [9]$$

où C et D sont des constantes, Δ_i est la différence entre $\overline{\mu}^*$, la moyenne des moyennes des récompenses dans $\mathcal{K}^*$ et μ_i, la moyenne de i : $\Delta_i = \overline{\mu}^* - \mu_i$, δ_i est la différence entre $\underline{\mu}^*$, la moyenne associée à la moins bonne action de $\mathcal{K}^*$ et μ_i : $\delta_i = \underline{\mu}^* - \mu_i$, σ_i^2 est la variance de l'action i.

A une constante additive près, ce résultat nous permet de garantir une convergence logarithmique. Bien que ce résultat soit valide pour le cas où l'ensemble complet des utilisateurs est connu, il nous permet d'affirmer que lorsque un utilisateur optimal (avec une moyenne parmi les k meilleures) entre dans l'ensemble $\mathcal{K}$, le processus converge de façon logarithmique vers une politique le reconnaissant comme tel.
Notons que nous considérons ici le cas des distributions de récompenses stationnaires. Bien que cette hypothèse ne soit pas toujours vérifiée dans notre contexte, cette preuve de convergence nous indique que si des sources sont bonnes pendant une période de temps suffisamment longue, notre algorithme est en mesure de les détecter.

4. Expériences

4.1. *Définition des fonctions d'utilité*

Nous considérons trois types de fonctions d'utilité caractérisant la "qualité" d'un utilisateur relativement à son activité sur un réseau social. La première considère le contenu des messages publiés, qui correspond au cas où l'on souhaite collecter des

1. `http://www-connex.lip6.fr/~lampriers/CORIA2015-supplementaryMaterial.pdf`

informations sur un sujet prédéfini et orienter la collecte vers les utilisateurs les plus susceptibles de publier sur ce sujet. La seconde fonction d'utilité prend en compte la popularité de l'utilisateur et la troisième est un modèle hybride prenant en compte la popularité d'un utilisateur sur une thématique donnée.

Modèle orienté contenu : Nous utilisons une représentation de type sac de mots. En considérant un dictionnaire de d mots, une requête est définie par un vecteur $\mathcal{Q}$ de taille d notée : $\mathcal{Q} = (w_j^{\mathcal{Q}}, j = 1..d)$, où $w_j^{\mathcal{Q}}$ est le poids du mot j dans la requête $\mathcal{Q}$. Nous modélisons également le contenu de $\omega_{i,t}$ comme un vecteur $\mathcal{D}_{i,t} = (w_j^{i,t}, j = 1..d)$, où $w_j^{i,t}$ est le poids du terme j dans le message posté à cette période. Si un utilisateur poste p messages $\omega_{i,t}^1, ..., \omega_{i,t}^p$, on les concatène en un seul. Ce vecteur est construit en appliquant un processus de *stemming* (Porter Stemmer et retrait des *stopwords*) aux messages collectés, en filtrant les mots absents dans le dictionnaire et en assignant un poids tf aux mots restants ($w_j^{i,t}$ est le nombre d'occurrences du terme j dans le message). La fonction de qualité $g_1(\omega_{i,t})$ est définie comme le cosinus entre le contenu $\mathcal{D}_{i,t}$ de $\omega_{i,t}$ et la requête $\mathcal{Q}$: $g_1(\omega_{i,t}) = \frac{\mathcal{D}_{i,t}.\mathcal{Q}}{\|\mathcal{D}_{i,t}\|\|\mathcal{Q}\|}$

Modèle de popularité : Ce modèle prend en compte les interactions de l'utilisateur suivi avec les autres membres du réseau. En particulier, l'API Twitter nous permet de compter le nombre de *Retweet* et de *Reply* sur un compte écouté, ce qui représente un bon indicateur de son influence, indépendamment du contenu des messages postés. Plus formellement, en notant $n_{i,t}^{Rt,k}$ et $n_{i,t}^{Rp,k}$ respectivement le nombre de *Retweet* et *Reply* sur le message $\omega_{i,t}^k$, la seconde fonction d'utilité est définie par :

$$g_2(\omega_{i,t}) = \tanh(\sum_{k=1}^{p} n_{i,t}^{Rt,k} + n_{i,t}^{Rp,k})$$

Modèle de popularité orienté contenu : Ce modèle hybride regroupe les deux précédents en prenant en compte à la fois le contenu des messages mais aussi leur popularité. Nous souhaitons orienter la collecte vers des personnes influentes sur des thématiques données. La fonction de qualité est définie de la façon suivante :

$$g_3(\omega_{i,t}) = \tanh(\sum_{k=1}^{p} g_1(\omega_{i,t}^k) \times n_{i,t}^{Rt,k})$$

4.2. *Résultats*

4.2.1. *Expériences hors ligne*

Les expériences hors ligne ont d'abord été effectuées sur deux ensembles de données enregistrées, ce qui permet de simuler un processus de *streaming* plusieurs fois et de comparer divers algorithmes. En particulier, cela permet de considérer une politique de référence qui consiste à suivre les utilisateurs optimaux, ce qui ne peut être fait dans une expérience en ligne où ces utilisateurs ne sont pas connus. Nous utilisons deux jeux de données avec des propriétés différentes :

– Le premier jeu de données, appelé *USElections*, correspond à un ensemble de messages récoltés grâce à l'API Twitter en suivant 5000 comptes d'utilisateurs sur une

période de 10 jours précédant les élections américaines de 2012. Nous avons choisi ces comptes en prenant les 5000 premiers à avoir employé les mots-clés "Obama", "Romney"' ou "#USElections". Il contient 2148651 tweets provenant des 5000 utilisateurs suivis ainsi que 1686797 messages correspondant à des *Retweet* ou *Reply*. La taille du pas de temps considéré est de 100 secondes.

– Le second jeu de données, appelé *Libye*, résulte d'une collecte de 3 mois sur le mot-clé "Libya" avec l'API de Twitter. Il contient 1211475 messages de 17341 utilisateurs. Le pas de temps est de 500 secondes (la fréquence des messages étant plus faible que sur le premier jeu de données).

Pour le modèle orienté contenu nous utilisons un dictionnaire spécifique pour chacun des jeux de données, construit en choisissant les 2000 *stems* les plus fréquents parmi les messages du corpus. Étant donné que les résultats peuvent varier selon les requêtes définies, les résultats présentés ci-dessous correspondent à la moyenne obtenue sur 500 requêtes aléatoires.

Resultats

En plus de notre algorithme *CUCBV*, nous considérons l'algorithme *CUCB* et les comparons à deux politiques de référence :

– Une politique *BestSubset*, qui se focalise sur le sous ensemble d'utilisateurs optimaux $\mathcal{K}^*$ (au sens décrit dans la section 3) ;

– Une politique *Random*, qui effectue à chaque pas de temps une sélection aléatoire uniforme de k utilisateurs à écouter.

La figure 2 présente l'évolution du gain cumulé en fonction du temps pour les algorithmes testés. Pour le jeu de données *USElections* les trois modèles de fonction d'utilité sont testés et pour *Libye*, seul le modèle orienté contenu est testé. Pour tous les algorithmes, le nombre d'utilisateurs simultanément suivis est fixé à $k = 100$.
Premièrement, nous remarquons que la courbe la plus élevée correspond à la politique *BestSubset* dans toutes les expériences. Cela confirme que l'hypothèse de stationnarité de notre modèle est pertinente. En effet, choisir un sous ensemble fixe d'utilisateurs à écouter pendant la totalité du processus semble fournir de bons résultats en termes de gain cumulé. Si cette hypothèse était complètement invalide, il n'y aurait aucune raison pour que cette stratégie soit plus performante que la stratégie *Random*. En outre, le fait que la pente de la courbe de la politique *BestSubset* soit plus forte que celle de la politique *Random* dans toutes les expériences et à chaque instant est un autre facteur permettant d'affirmer que certains utilisateurs apportent plus de gains que d'autres pendant toute la durée du processus. Par ailleurs, les deux stratégies proposées offrent de meilleurs résultats que la stratégie *Random*, ce qui confirme que l'application d'algorithmes de bandits pour la capture de données sur un réseau social permet de s'orienter efficacement vers les zones du réseau les plus pertinentes.
Pour toutes les fonctions de qualité et sur le jeu de données *USElections*, l'algorithme *CUCBV* obtient de meilleurs performances que *CUCB*. La prise en compte de la variance empirique dans le terme d'exploration lui permet de réaliser de meilleures es-

timations de l'utilité des sources à forte variabilité. Dans notre cadre, ceci est particulièrement pertinent puisque le gain obtenu dépend fortement de la fréquence de publication d'un utilisateur. Certains d'entre eux peuvent être bien moins actifs que d'autres mais obtenir de meilleurs gains, du fait qu'ils publient essentiellement sur une thématique précise. Considérer la variance pour l'exploration permet de donner plus de chances à ces derniers.
Sur le jeu de données *Libye*, nous observons également que les deux algorithmes sont plus performants que la politique *Random*. On peut noter que les courbes ont une forme particulière aux alentour du pas de temps 8000. Cette période correspond à un pic d'activité relatif à un événement politique en Libye. Même si les algorithmes de bandit sont capables de suivre la tendance, les courbes sont éloignées de la courbe associée à la politique *BestSubset*. De tels pics d'activité sont difficiles à gérer pour ce type d'algorithme car ils ont tendance à se focaliser sur certains comptes à mesure que le temps s'écoule. Cependant, dans une expérience en ligne où les données sont collectées sur des utilisateurs et non des mots-clés comme c'est le cas ici, ces pics sont moins susceptibles d'être observés car les médias sociaux comme Twitter conservent un fort taux de messages postés.

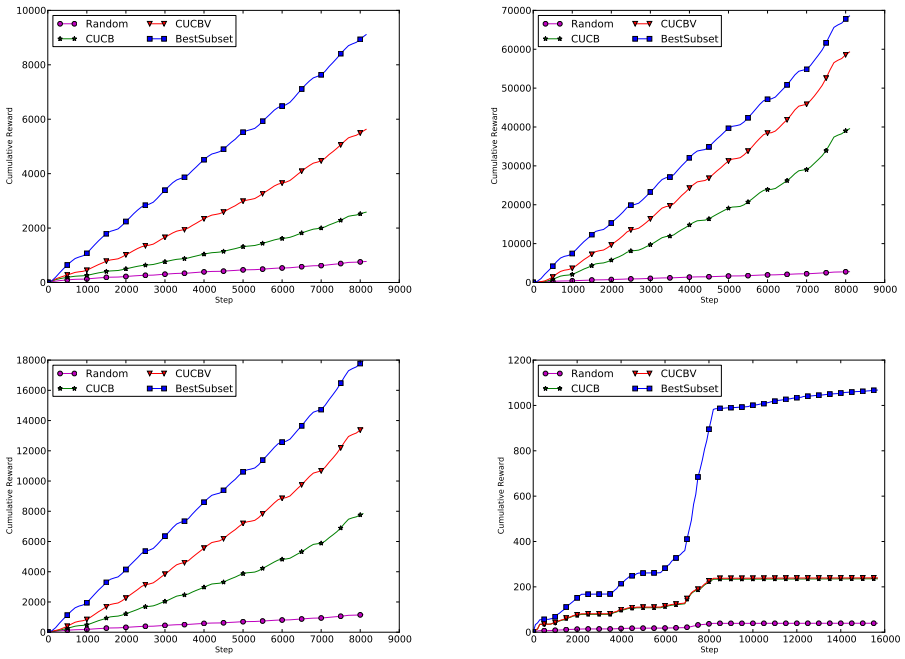

Figure 2. *Gain cumulé en fonction du temps. Haut gauche : USElections / modèle orienté contenu. Haut droite : USElections / modèle de popularité. Bas gauche : USElections / modèle hybride. Bas droite : Libye / modèle orienté contenu.*

La figure 3 de gauche représente l'évolution de le taux de capture sur le jeu de données *USElections* pour le modèle de fonction de qualité hybride. Ce taux est calculée à l'instant t et pour la politique π comme la moyenne des gains obtenus par π pendant la fenêtre de temps $[t-500, t+500]$, normalisée par la moyenne des gains qui auraient été obtenus en suivant la politique optimale sur la même période :

$$C(\pi, t) = \sum_{s=t-500}^{t+500} \sum_{i \in \pi(s)} g(\omega_{i,s}) \; / \; \sum_{s=t-500}^{t+500} \sum_{i \in \mathcal{K}^*} g(\omega_{i,s}) \qquad [10]$$

Les résultats présentés permettent de souligner qu'alors que le taux de capture semble rester constant pour la stratégie *Random* , ce dernier tend à s'accroître pour les deux autres algorithmes pendant la durée du processus. On note qu'à la fin du processus, *CUCBV* atteint un taux de capture très proche de celui de la politique *BestSubset*.

4.2.2. *Expérience en ligne*

Nous proposons maintenant de tester nos modèles sur une expérience grandeur nature en utilisant l'API de Streaming de Twitter. Nous nous focalisons sur la comparaison de *CUCBV* avec une politique *Random*, ce qui nécessite deux comptes Twitter. Les résultats reportés correspondent à une collecte d'une semaine, en prenant comme comptes initiaux *BBC*, *CNN* et *FoxNews* pour les deux politiques. L'ensemble des comptes $\mathcal{K}$ est alimenté à chaque itération t par les utilisateurs ayant *retweeté* ou répondu aux utilisateurs de $\mathcal{K}_t$. Nous considérons le modèle de popularité orienté contenu, où la requête a été créée en prenant les 300 premiers mots employés dans la page Wikipédia de la Libye. Pour nos expériences, le nombre d'utilisateurs écoutés simultanément est fixé à 5000 (maximum autorisé par l'API) et la durée de chaque itération est de 300 secondes.

Resultats

La figure 3 de droite représente le gain cumulé obtenu avec les politiques *Random* et *CUCBV* en fonction du temps. Elle met en valeur les bonnes performances de la méthode proposée car la courbe *CUCBV* augmente significativement plus rapidement que la courbe *Random*. La valeur de 5000 utilisateurs peut prendre un certain temps à être atteinte, c'est pourquoi au début du processus, les deux stratégies suivent les mêmes utilisateurs. A la fin du processus, *CUCBV* obtient un gain cumulé de 50000, tandis que *Random* obtient 20000. En considérant une période de *streaming* plus longue, il est raisonnable de penser que cet écart se creuserait. Finalement, notons que le nombre total d'utilisateurs est beaucoup plus large pour *CUCBV* que pour *Random*, ce qui est cohérent avec le fait que l'on s'oriente mieux vers des utilisateurs populaires, plus souvent *retweetés*.

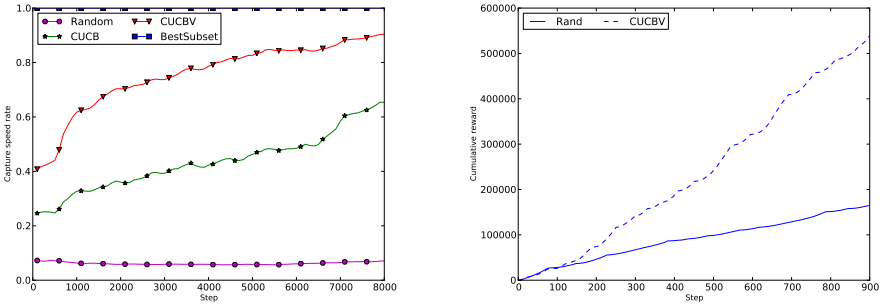

Figure 3. *Gauche : Taux de capture pour le jeu de données USElections et le modèle hybride. Droite : Gain cumulé en fonction du temps pour l'expérience en ligne.*

5. Etat de l'art

5.1. *Collecte de données*

Une très large littérature s'est intéressée à l'identification et le suivi d'événements dans les flux de données. Le domaine de la Topic Detection and Tracking (TDT) propose des modèles pour traiter cette problématique. Par exemple dans (Cataldi *et al.*,, 2010) les auteurs ont proposé une approche traitant en continu des messages provenant de Twitter afin de découvrir des mots-clés correspondant à des sujets en vogue sur le réseau. Cependant, la question de la collecte est laissée de côté dans ce travail, qui utilise simplement le *stream* aléatoire proposé par Twitter. Plus récemment dans (Colbaugh *et al.*,, 2011) la blogosphère est modélisée comme un réseau où chaque nœud correspond à un blog. L'objectif est d'identifier les blogs qui repostent les contenus émergents du moment. En ce sens, l'approche proposée oriente la collecte vers des sources de données utiles, mais cela se fait de manière statique, sur des données d'apprentissage collectées sur tous les nœuds du réseau, ce qui n'est possible que lorsque le réseau est suffisamment petit. Le travail présenté dans (Li *et al.*,, 2013) est probablement l'un des plus proches du notre, car il propose un processus de réorientation de la collecte via l'API *streaming* de Twitter. Cependant, cette approche concerne uniquement la recherche de thématiques émergentes. De plus, à la différence de notre méthode où l'on s'oriente peu à peu vers des zones intéressantes du réseau, les auteurs redéfinissent l'ensemble des mots à suivre à chaque étape, en sélectionnant les termes les plus employés du moment à partir d'observations faites depuis le flux aléatoire proposé par Twitter. Le *crawling* de page Web est un autre domaine de recherche lié à notre travail : des stratégies sont définies pour sélectionner les nœuds du web à visiter à chaque étape du processus de collecte. Le *crawling* ciblé, introduit dans (Chakrabarti *et al.*,, 1999), décide si une page web doit être analysée ou non en fonction de la pertinence de son contexte et de ses liens avec un sujet prédéfini. Ces méthodes nécessitent

une connaissance préalable de la structure des liens entre les sources de données, ce qui n'est généralement pas le cas dans notre contexte où l'extraction de la structure sous-jacente est une tâche très difficile. (Boanjak *et al.,*, 2012) ont proposé une solution distribuée pour effectuer du *crawling* ciblé sur Twitter, mais nécessitant d'interroger l'API de Twitter de façon intensive, depuis de nombreux clients, dans le but d'obtenir le graphe social du réseau. Alors que le nombre de requêtes était déjà limitée par Twitter au moment de ce travail, cette approche ne semble pas envisageable aujourd'hui étant donné les conditions d'utilisation de plus en plus strictes de l' API.

5.2. *Algorithmes de bandits*

Un très grand nombre de publications traitent des algorithmes de bandit et de leurs applications. Ces derniers sont particulièrement adaptés aux problèmes en ligne, telle que la collecte de données, ainsi qu'à l'apprentissage lorsque qu'aucune information a priori n'est disponible. Dans le cadre des réseaux sociaux, les algorithmes de bandit ont été utilisés avec succès dans les domaines de la recommandation dans (Kohli *et al.,*, 2013), du *crawling* dans (Bnaya *et al.,*, 2013), ou encore de la publicité en ligne dans (Buccapatnam *et al.,*, 2014). Cependant, aucun de ces travaux ne traite de prise de décisions simultanées, comme c'est le cas dans notre article. Le problème du bandit combinatoire a été étudié dans (Chen *et al.,*, 2013) où l'algorithme *CUCB* est proposé. Plus récemment dans (Gopalan *et al.,*, 2014), les auteurs généralisent la méthode du *Thompson Sampling*, un autre algorithme de bandit, pour la sélection de k actions parmi un ensemble plus grand de K actions à chaque instant. Dans notre contexte, cet algorithme est difficile à déployer à cause du nombre trop important d'actions, un processus d'échantillonnage complexe et coûteux étant nécessaire à chaque itération.

6. Conclusion

Dans cet article, nous avons abordé le problème de la collecte de données sur les médias sociaux comme un problème d'apprentissage en temps réel. Notre approche est basée sur le formalisme du problème de bandit combinatoire, pour lequel nous avons proposé une extension du récent algorithme *CUCB*. Face à la forte variabilité des récompenses obtenues dans notre contexte, nous avons défini l'algorithme *CUCBV*, prenant en compte la variance dans sa stratégie d'exploration. Après avoir prouvé certaines garanties de convergence, nous l'avons expérimenté dans le contexte des médias sociaux. Les expériences sur données réelles, aussi bien hors ligne qu'en ligne, ont montré la capacité de l'algorithme à s'orienter automatiquement vers des sources pertinentes relativement à une fonction de qualité donnée.
Le travail exposé, qui représente une nouvelle façon de s'intéresser aux médias sociaux, ouvre ainsi plusieurs perspectives. Alors que les modèles utilisés ici sont basés sur l'hypothèse que les distributions sont stationnaires (i.e un utilisateur a un comportement constant au cours du temps), nous sommes actuellement en train d'expérimenter d'autres méthodes pour des distributions non-stationnaires, qui seront mieux adap-

tées pour des collectes sur des durées plus longues dans un environnent complexe. Une autre perspective intéressante serait d'étendre ce travail pour des récompenses combinatoires, où la fonction d'utilité d'un ensemble d'utilisateurs n'est plus une somme des récompenses individuelles, mais pourrait, par exemple prendre en compte la diversité des utilisateurs sélectionnés. Le travail présenté ici peut servir de base à divers problèmes de modélisation en ligne des échanges de contenu sur les réseaux sociaux.

7. Remerciements

Ce travail a été effectué dans le cadre des recherches menées au sein de l'Institut de Recherche Technologique SystemX et a ainsi bénéficié d'une aide de l'Etat au titre du programme d'Investissements d'Avenir.

8. Bibliographie

Audibert J.-Y., Munos R., Szepesvári C., « Tuning Bandit Algorithms in Stochastic Environments », ALT '07, p. 150-165, 2007.

Auer P., Cesa-Bianchi N., Fischer P., « Finite-time analysis of the multiarmed bandit problem », *Machine Learning*, vol. 47, n. 2-3, p. 235-256, 2002.

Bnaya Z., Puzis R., Stern R., Felner A., « Bandit Algorithms for Social Network Queries », *Social Computing (SocialCom), 2013 International Conference on*, p. 148-153, Sept, 2013.

Boanjak M., Oliveira E., Martins J., Mendes Rodrigues E., Sarmento L., « TwitterEcho : A Distributed Focused Crawler to Support Open Research with Twitter Data », WWW '12 Companion, ACM, NY, USA, p. 1233-1240, 2012.

Bubeck S., Cesa-Bianchi N., « Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems », *Foundations and Trends in ML*, vol. 5, n. 1, p. 1-122, 2012.

Buccapatnam S., Eryilmaz A., Shroff N. B., « Stochastic Bandits with Side Observations on Networks », SIGMETRICS '14, ACM, NY, USA, p. 289-300, 2014.

Cataldi M., Di Caro L., Schifanella C., « Emerging Topic Detection on Twitter Based on Temporal and Social Terms Evaluation », MDMKDD '10, ACM, NY, USA, p. 4 :1-4 :10, 2010.

Chakrabarti S., van den Berg M., Dom B., « Focused Crawling : A New Approach to Topic-specific Web Resource Discovery », WWW '99, Inc., NY, USA, p. 1623-1640, 1999.

Chen W., Wang Y., Yuan Y., « Combinatorial Multi-Armed Bandit : General Framework and Applications », vol. 28, JMLR Workshop and Conference Proceedings, p. 151-159, 2013.

Colbaugh R., Glass K., « Emerging Topic Detection for Business Intelligence Via Predictive Analysis of 'Meme' Dynamics », AAAI, 2011.

Gopalan A., Mannor S., Mansour Y., « Thompson Sampling for Complex Online Problems », *ICML*, vol. 32 of *JMLR Proceedings*, JMLR.org, p. 100-108, 2014.

Kohli P., Salek M., Stoddard G., « A Fast Bandit Algorithm for Recommendation to Users With Heterogenous Tastes. », *in* , M. desJardins, , M. L. Littman (eds), *AAAI*, AAAI Press, 2013.

Li R., Wang S., Chang K. C.-C., « Towards Social Data Platform : Automatic Topic-focused Monitor for Twitter Stream », *Proc. VLDB Endow.*, vol. 6, n. 14, p. 1966-1977, 2013.