Transfert entre campagnes de Real-Time Bidding

Charles Monzani *, Martial Hue**
Patrick Gallinari*

*Sorbonne Université, CNRS, LIP6, 4 place Jussieu, 75005 Paris
**Tradelab, 10 rue Henner, 75009 Paris

Résumé. La réussite d'une campagne publicitaire sur internet passe par une modélisation efficace des internautes : pour cela de nombreuses méthodes sont aujourd'hui utilisées, notamment des réseaux de neurones récurrents, qui permettent de prendre en compte les séquences d'action des internautes. Cependant il est nécessaire d'avoir déjà acquis un conséquent volume de données avant qu'ils ne soient vraiment efficaces. Ainsi au début de chaque campagne il y a une phase de pure collecte de données : les dépenses publicitaires sont effectuées de manière aléatoire afin d'acquérir des données non biaisées, de plus on ne sait pas encore quel type d'internautes viser : on parle de problème de Cold-Start.

Pour éviter ce problème, nous proposons de réutiliser dans données acquises sur des campagnes antérieures, en reprenant un réseau de neurones déjà entraîné.

1 Introduction

Afficher la bonne publicité, au bon internaute, au bon moment, est la clef de la réussite d'une campagne publicitaire en *Real-Time Bidding*, cette technologie qui permet la mise aux enchères en temps-réel d'un espace publicitaire sur une page (défini par l'adresse de la page, l'emplacement et le format de la bannière, ainsi que l'identifiant anonymisé de l'internaute). Deux métriques sont utilisées a posteriori pour évaluer une campagne : le coût par clic (CPC), qui correspond à la somme totale dépensée en achat de bannières divisée par le nombre de clics obtenus sur ces bannières ; et le coût par acquisition (CPA), qui correspond à la somme totale dépensée divisée par le nombre d'achats réalisés par des internautes ayant vu une bannière publicitaire de la marque (sans qu'ils aient nécessairement cliqué dessus). On parle ainsi de taux de clic (CTR) et de taux de conversion (CVR) pour, respectivement, la probabilité qu'un internaute clique sur la bannière ; et la probabilité qu'il réalise une conversion.

Pour éviter d'afficher une bannière à des internautes à faible probabilité de conversion, il est indispensable d'avoir une méthode efficace de représentation des internautes ainsi qu'une méthode efficace de calcul du taux de conversion. En raison du volume faible de conversion pour une campagne, il est généralement plus simple de calculer le taux de clic. L'association d'un *feature-engineering* poussé et d'une régression logistique a longtemps été la méthode de choix pour le calcul du taux de clic (Agarwal

et al., 2010), et ce aussi bien pour des raisons de performance finale que d'optimisation du temps de calcul, puisque la régression logistique est facilement parallélisable comme le montre Chapelle et al. (2014). Avec les techniques d'apprentissage profond, de nouvelles méthodes sont apparues, Zolna et Romanski (2017) montre ainsi l'intérêt d'utiliser un réseau de neurones récurrents, fondée sur des cellules LSTM (Hochreiter et Schmidhuber, 1997), pour analyser la séquence d'actions de l'internaute.

Quelle que soit la méthode choisie, il est nécessaire pour entraîner ces modèles d'accumuler assez de données, notamment d'exemples positifs car ces événements sont rares. Ainsi au début d'une campagne publicitaire, il faut attendre avant d'avoir des modèles aux performances satisfaisantes, on parle de problème de *cold-start* (Schein et al., 2002). Ce manque de données apparaît dans d'autres domaines et une solution a été trouvée : l'apprentissage par transfert (Pan et al., 2010). Il consiste à utiliser des données déjà acquises pour une tâche, sur un nouveau problème. On peut utiliser sur de nouvelles données des réseaux pré-entraînés, et cela fonctionne notamment avec des réseaux à convolution en reconnaissance d'images (Yosinski et al., 2014). Laptev et al. (2018) applique ce fonctionnement à des réseaux récurrents pour la prédiction de séries temporelles et Aggarwal et al. (2019) utilise l'adaptation de domaine pour le transfert entre campagnes de prospection RTB, mais dans le cas où des données de Retargeting sur l'annonceur cible ont déjà été récoltées.

Dans cet article, nous proposons une méthode de représentation de l'internaute fondée sur des réseaux de neurones récurrents. Dans un premier temps nous comparons cette méthode à une autre, intuitive, fondée sur des heuristiques. Puis dans un second temps nous proposons une méthode d'utilisation de données déjà accumulées sur des campagnes antérieures, sur une nouvelle campagne.

2 Notre approche

Le but de notre étude est d'utiliser le parcours de chaque internaute sur le site marchand de l'annonceur (souvent un site e-commerce) pour prédire si l'internaute va réaliser un achat sur ce même site dans un délai proche. Selon sa probabilité de conversion, on pourra dans un second temps décider du prix maximum auquel on est prêt à lui montrer une bannière. On a donc un problème de prédiction comme expliqué en figure 1. Chaque modèle est quotidien, c'est-à-dire qu'on souhaite prédire uniquement les conversions du lendemain [1].

Notre modélisation de l'internaute est fondée sur son parcours de navigation : on a donc à notre disposition uniquement ses actions sur le site du client, ce qui représente moins d'informations que dans les autres travaux sur le calcul du CTR qui prennent en compte un panel plus large de données : l'historique, l'âge de l'internaute, sa catégorie socio-professionnelle, ses achats récents sur des sites partenaires, etc. Ce choix est dicté par les réglementations récentes comme la RGPD [2].

Pour l'analyse du parcours de navigation, on utilise un réseau de neurones récurrents avec des cellules LSTM. Cette méthode permet de gérer des séquences de navigation

1. Cela est dû au fait qu'il est facile de mettre à jour, chaque nuit, pour chaque internaute, le prix à payer.
2. `www.cnil.fr`

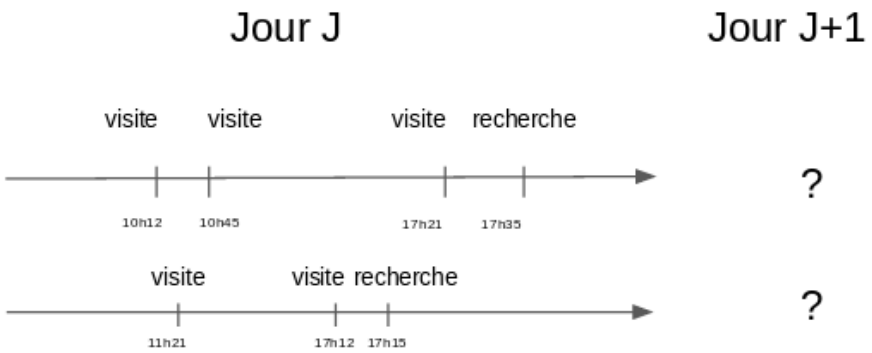

FIG. 1 – *Le problème de prédiction.*

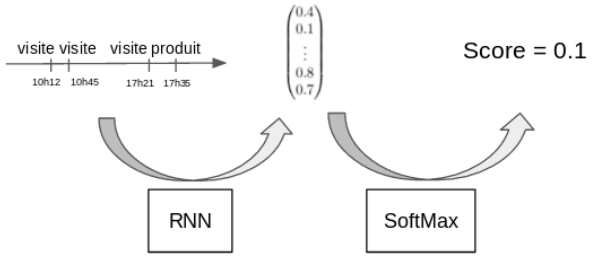

FIG. 2 – *Pipeline : de la séquence de navigation au score.*

de taille variables (comme c'est le cas dans nos données : tous les internautes n'ont pas le même nombre d'événements sur un site marchand), et permet de conserver l'ordre entre les événements en donnant en plus un poids plus important aux plus récents. Cependant elles ne sont plus adéquates pour traiter des séquences trop longues, ce n'est pas un problème ici car, compte tenu de la durée de vie d'un identifiant, les séquences ne contiennent généralement pas plus de quelques dizaines d'événements, ce qui exclut aussi la possibilité d'enrichir son réseau avec de l'apprentissage multi-tâche (comme de prévoir la probabilité de retour sur le site ou la probabilité d'achat sous 30 jours).

En sortie réseau de neurones récurrents, on a ainsi un vecteur représentant l'internaute, auquel on applique une régression logistique (ou softmax) pour obtenir sa probabilité d'achat (figure 2).

Problème de Cold-Start Pour ne pas perdre de temps lors du lancement d'une nouvelle campagne, on va réentraîner nos modèles uniquement sur les événements en commun entre les anciennes campagnes (où nous avons des données) et la nouvelle campagne (où nous n'avons pas encore de données). En effet, de nombreux événements

sont communs entre deux campagnes, comme par exemple l'événement *arrivée sur le site en provenance d'un réseau social* : cependant selon la manière dont la récolte de données sur le site de la marque a été organisée, il y a des événements différents d'une campagne à l'autre (voir tableau 1). L'idée sous-jacente est que les schémas de navigation sont assez proches d'un site e-commerce à un autre.

	# total d'événements	# d'évenements en commun
Annonceur A	28	12
Annonceur B	42	14
Annonceur C	39	14

TAB. 1 – *Recouvrement des jeux de données*

3 Expériences

Pour évaluer les performances de nos modèles, nous avons réalisé des tests rétroactifs de validité sur les données de l'entreprise Tradelab, des mois d'avril et mai 2019 pour 3 annonceurs : les modèles sont entraînés sur le mois d'Avril et testés sur les données de Mai, la répartition moyenne entre séquences positives et négatives pour chaque annonceur est indiqué en tableau 2. Tous les modèles sont entraînés avec les paramètres suivants : 1 couche cachée de dimension 64, pour 50 itérations, avec des batchs équilibrés entre positifs et négatifs. L'entraînement d'un modèle quotidien prend entre 30 minutes (pour le modèle de base) et 2 heures (pour les modèles en transfert).

	# moyen de séquences	# moyen de positifs
Annonceur A	860800	40
Annonceur B	101000	2130
Annonceur C	2095000	320

TAB. 2 – *Nombre moyen de séquences par jour*

3.1 Réseau de neurones récurrents face à un modèle simple de comptage

On souhaite d'abord valider notre modélisation, pour cela on teste les performances du modèle avec notre représentation face à un simple modèle de comptage : pour chaque parcours de navigation, on regarde combien de fois chaque événement a eu lieu, et on forme ainsi le vecteur de représentation de l'internaute (qui a donc pour dimension le nombre d'événements différents). On a ainsi pour chaque annonceur et chaque modélisation (RNN ou comptage), sept modèles, chacun correspondant à un jour de la semaine, afin de tester sur le jour correspondant. Cela se justifie par la différence observée de comportement des internautes entre la semaine et le week-end.

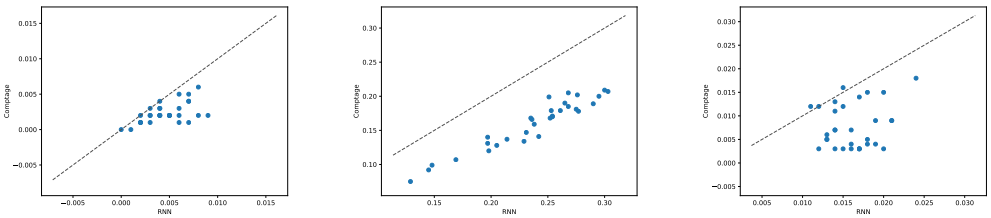

FIG. 3 – *RNN vs Comptage : Nuages de points pour l'annonceur A, B et C. En abscisse la performance du modèle avec le RNN, en ordonnée celle du modèle avec comptage.*

3.2 Cold-Start

On souhaite vérifier que notre approche permet d'éviter le problème de Cold-start. Pour cela on entraîne un premier modèle sur les données d'avril des annonceurs A et B, et on teste sur les données de mai de l'annonceur C. On prend pour comparaison un modèle RNN entraîné sur les données d'avril testé sur les mêmes données de mai. L'idée est de montrer que les performances sont proches. Comme pour la comparaison entre RNN et comptage, on a un modèle par jour de la semainee. On réalise cette expériences pour les 3 couples d'annonceurs A-B, B-C et A-C testés respectivement sur l'annonceur C, A et B.

4 Résultats

Pour l'évaluation de chaque modèle : on classe chaque séquence de l'ensemble de test par probabilité d'achat décroissante, et on fait évoluer le seuil au dessus duquel on prédit les séquences comme positives. On obtient ainsi des courbes précision-rappel, plus adéquats pour des problèmes à faible taux de positifs que les courbes ROC (Davis et Goadrich, 2006).

De plus, on obtient pour chaque jour de test la précision moyenne de chaque modèle, qui nous permet d'obtenir une courbe en nuage de points.

4.1 Réseau de neurones récurrents face à un modèle simple de comptage

	RNN	Comptage
Annonceur A	$4.6\ 10^{-3}$	$2.2\ 10^{-3}$
Annonceur B	0.24	0.16
Annonceur C	0.016	$7.8\ 10^{-3}$

TAB. 3 – *RNN vs Comptage : Précision moyenne*

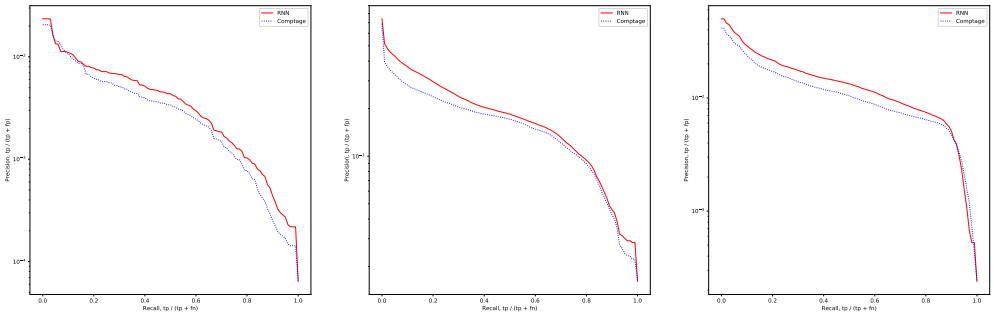

FIG. 4 – *RNN vs Comptage : Courbes Précision-Rappel pour l'annonceur A, B et C. En rouge la performance du modèle avec RNN, en bleu celle du modèle avec comptage*

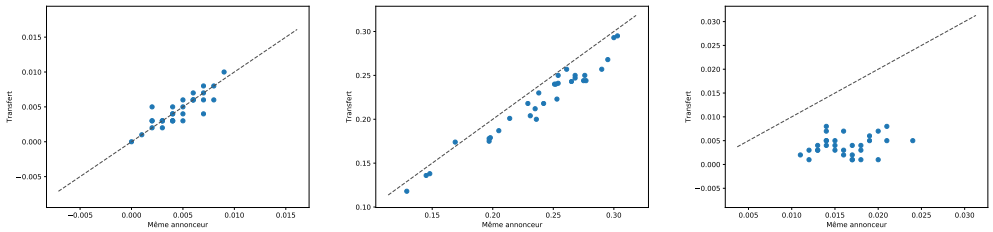

FIG. 5 – *Cold-start : Nuages de points pour les tests sur les annonceurs A, B et C. En abscisse la performance du modèle avec le RNN du même annonceur, en ordonnée celle du modèle en transfert.*

On observe sur les figures 3 et 4 ainsi que sur le tableau 3 les résultats de notre première expériences : dans une grande majorité de cas (tous les jours sauf 3 pour l'annonceur C), la représentation par un RNN obtient une meilleure précision que celle par comptage, cela valide ainsi notre approche.

4.2 Cold-Start

On observe sur les figures 5 et 6 les résultats de notre deuxième expérience. Qualitativement, on voit que sur l'annonceur A et sur l'annonceur B, le modèle de transfert est proche de la performance de base, ce qui est validé de manière quantitative par la précision moyenne (tableau 4).

Cependant pour l'annonceur C, le modèle en transfert reste très éloigné du modèle de base. Cela peut s'expliquer par la différence de volume entre ces annonceurs. En effet il y a en moyenne deux fois plus de séquences pour l'annonceur C que pour les annonceurs A et B, on compare ainsi un modèle qui a appris sur deux fois plus de données et sur plus d'événements différents. Il serait intéressant de tester sur l'annonceur C un modèle entraîné conjointement sur les annonceurs A,B et C.

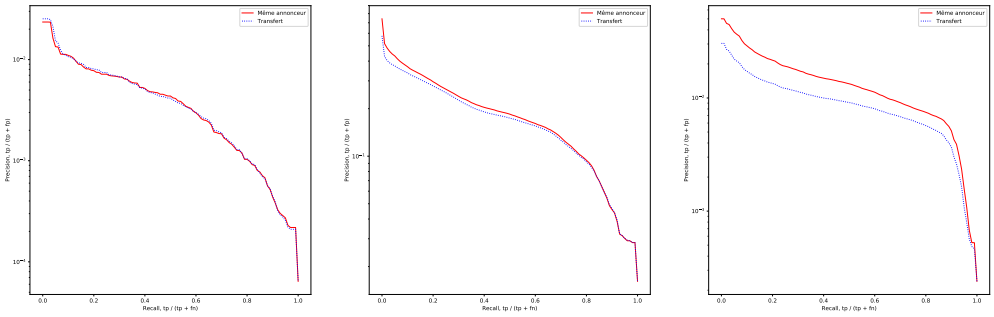

FIG. 6 – *Cold-start : Courbes Précision-Rappel pour les tests sur les annonceurs A, B et C. En rouge la performance du modèle avec le RNN du même annonceur, en bleu celle du modèle en transfert*

	Même annonceur	Transfert
Annonceur A	$4.6\ 10^{-3}$	$4.5\ 10^{-3}$
Annonceur B	0.24	0.22
Annonceur C	0.016	$3.9\ 10^{-3}$

TAB. 4 – *Cold-start : Précision moyenne*

5 Conclusion

Nous avons montré que pour une modélisation des internautes à partir de leur simple séquence d'action sur le site marchand, l'adoption d'un réseau de neurones récurrents est adéquate. De plus, on peut, dans certains cas, réutiliser un réseau déjà entraîné sur un premier annonceur, pour prédire les conversions d'un second annonceur. Cependant, cette méthode n'est pas à toute épreuve et il reste difficile de prédire a priori les cas où elle fonctionne. Pour aller plus loin, on pourrait penser à une représentation complète d'un internaute à partir de ses actions sur les sites de tous les annonceurs, au lieu d'avoir une représentation par internaute par annonceur.

Références

Agarwal, D., R. Agrawal, R. Khanna, et N. Kota (2010). Estimating rates of rare events with multiple hierarchies through scalable log-linear models. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 213–222. ACM.

Aggarwal, K., P. Yadav, et S. S. Keerthi (2019). Domain adaptation in display advertising : an application for partner cold-start. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pp. 178–186. ACM.

Chapelle, O., E. Manavoglu, et R. Rosales (2014). Simple and scalable response prediction for display advertising. *ACM Trans. Intell. Syst. Technol. 5*(4), 61 :1–61 :34.

Davis, J. et M. Goadrich (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240. ACM.

Hochreiter, S. et J. Schmidhuber (1997). Long short-term memory. *Neural Comput. 9*(8), 1735–1780.

Laptev, N., J. Yu, et R. Rajagopal (2018). Reconstruction and regression loss for time-series transfer learning. *SIGKDD MiLeTS–2018–8* .

Pan, S. J., Q. Yang, et al. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering 22*(10), 1345–1359.

Schein, A. I., A. Popescul, L. H. Ungar, et D. M. Pennock (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 253–260. ACM.

Yosinski, J., J. Clune, Y. Bengio, et H. Lipson (2014). How transferable are features in deep neural networks ? In *Advances in neural information processing systems*, pp. 3320–3328.

Zolna, K. et B. Romanski (2017). User modeling using lstm networks. In *AAAI*, pp. 5025–5027.

Summary

An efficient way of modelling web users is needed in the optimisation of an advertising campaign: a lot of different methods have been used, including recurrent neural networks. Using those networks, it is possible to take into account the order of the events in a sequence of browsing. Nevertheless, in order for those to have good enough performance, a consequent amount of data is needed. Because of that, it seems impossible to use them at the beginning of an ad campaign, when not enough data about browsing behaviour have been gathered: and the ad budget is spent randomly in order to gather unbiased data: this is the cold-start problem.

To avoid that problem, we suggest leveraging data from previous campaigns, from other announcers, reusing an already trained neural network.