BRec the Bank : encodeur auto-attentif sensible au contexte pour la recommandation de produits bancaires

Davide Liu[*,**], George Philippe Farajalla[*], Alexandre Boulenger[*,**]

[*] Genify.ai
{davide, george.farajalla, alex}@genify.ai
[**] Tsinghua University

Résumé. Cartes de crédit, dépôts, prêts, fonds de pension, fonds communs de placement – lesquels de ces produits sont pertinents pour les clients d'une banque et à quel moment dans leur parcours bancaire ? Nous proposons un cadre de modélisation pour la recommandation de produits à l'aide d'un encodeur auto-attentif multi-têtes et d'une représentation novatrice des données d'entrée, sensible au contexte temporel de l'acquisition des produits et aux métadonnées de l'utilisateur. Évalué sur un vaste jeu de données public de la banque Santander, notre modèle atteint une précision top-1 et top-5 de 98.9% et 40.2%, respectivement, surpassant ainsi un nombre de modèles de recommandation de pointe. Nous évaluons aussi les métriques de sérendipité, de nouveauté et de couverture. Le plongement des utilisateurs appris par le modèle a le potentiel d'informer des décisions d'ampleur plus importante que ladite recommandation de produits.

1 Introduction

L'objectif d'un système de recommandation de produits est de fournir aux utilisateurs des recommandations de produits sensées et basées sur leur comportement passé, leur similitude avec d'autres utilisateurs et la compréhension d'un certain contexte. Les moteurs de recommandation peuvent identifier les informations pertinentes à partir des préférences et des intérêts des utilisateurs, ce qui est essentiel pour recommander des produits. Les systèmes de recommandation se sont considérablement développés à la fin du XXe siècle et ont été principalement appliqués dans la vente en ligne : Amazon, eBay, CDNOW (Schafer et al., 1999). Aujourd'hui, les moteurs de recommandation sont omniprésents dans la vente en ligne ; Amazon et Alibaba utilisent des systèmes pour recommander de nouveaux articles à leurs clients, réduisant ainsi le champ de recherche de l'utilisateur et le surprenant en lui proposant de nouveaux produits qu'il pourrait désirer en fonction de ses données actuelles et passées. Des moteurs de recommandation ont également été employés dans la gestion de patrimoine et ont récemment gagné du terrain dans la banque de détail. Plusieurs compétitions Kaggle organisées par Santander soulignent le besoin croissant de recommander des produits bancaires pertinents aux clients.

Les avancées récentes dans le traitement du langage naturel, alimentées par l'architecture Transformer (Vaswani et al., 2017), ont conduit au développement de systèmes de recommandation basés sur l'auto-attention. Les avantages de ce type de modèles sont les représentations d'entrée et de sortie apprises lors de l'apprentissage. Celles-ci constituent un plongement

continu des utilisateurs, qui peut s'avérer utile pour des applications en aval, souvent sans rapport direct avec la tâche de recommandation de produits, mais à forte valeur ajoutée.

Le présent article est un résumé de l'article publié dans la conférence IJCNN 2022 (Liu et al., 2022). Ses principaux apports sont les suivants :

— un cadre de modélisation basé sur l'auto-attention et une représentation novatrice des données d'entrée, temporelle et flexible (non spécifique au secteur bancaire) ;
— une illustration de l'efficacité de ce modèle, qui surpasse quatre modèles de pointe sur un jeu de données de recommandation de produits de banque de détail ;
— une analyse poussée de la performance via des métriques outre la pertinence ;
— le code source en libre accès afin de faciliter la reproduction des résultats. [1]

2 Travaux reliés

L'apprentissage profond a récemment influencé le développement des systèmes de recommandation, en particulier ceux basés sur le filtrage collaboratif. Cependant, ces méthodes ne tiennent pas compte du comportement séquentiel des utilisateurs ; par exemple, un utilisateur est plus susceptible d'acheter une souris s'il possède déjà un ordinateur. Les réseaux de type RNN, GRU et LSTM ont pallié ce manque, et les résultats expérimentaux suggèrent que les dépendances implicites entre événements successifs peuvent être exploitées.

Compte tenu de l'historique des interactions entre un utilisateur et les articles, et du contexte spécifique à l'utilisateur, un moteur de recommandation moderne se doit d'apprendre les schémas temporels, séquentiels et contextuels. Parmi les approches possibles, le Transformer (Vaswani et al., 2017) a donné lieu à des gains de performance à la tâche de recommandation.

SASRec (Kang et McAuley, 2018), un modèle de recommandation séquentiel récent basé sur l'auto-attention, a également été conçu pour tenir compte de manière adaptative des articles déjà acquis. TiSASRec (Li et al., 2020) a encore amélioré les performances, proposant un modèle auto-attentif, sensible à l'intervalle de temps, tenant compte à la fois des intervalles relatifs et de la position absolue dans la séquence d'articles afin de prédire les interactions futures. De même, BERT4Rec (Sun et al., 2019) emploie un mécanisme d'attention bidirectionnelle et une fonction objectif Cloze pour modéliser les séquences de comportement des utilisateurs. MEANTIME (Cho et al., 2020) propose une amélioration à TiSASRec (Li et al., 2020) en introduisant plusieurs plongements temporels.

A l'inverse de l'approche que nous proposons, ces modèles, à l'exception de MEANTIME, représentent les séquences d'articles sans notion forte de temps (***e.g.***, BERT4Rec considère uniquement la position relative des éléments), et ils n'encodent pas les articles non acquis.

3 Formulation du problème

Nous appelons *données* l'union des *produits* et des *métadonnées*. Considérons un ensemble d'utilisateurs C, leurs métadonnées M et un ensemble de produits B. $B_c \subset B$ dénote les produits déjà acquis par $c \in C$, M_c ses métadonnées, et $H_c = B_c \cup M_c$ ses données.

Nous modélisons la tâche de recommandation comme un problème de classification multi-classes où le but est d'apprendre une fonction $f : (c, H_c) \rightarrow B \times [0, 1]$ qui associe un utilisateur

1. https://github.com/genifyai/brec_ijcnn2022

$c \in C$ et ses données H_c à une distribution de probabilité P_c^B sur l'ensemble des articles B, probabilité que c les acquière lors de la prochaine période. Il est naturel de considérer les variables M_c et B_c comme des séquences, variant à travers le temps. Étant donné une séquence d'articles B_c^t et une séquence de métadonnées M_c^t où $t = [0, ..., T]$, T est la longueur de la séquence, le but du modèle est d'apprendre une fonction $f : (c, H_c^0, ..., H_c^T)$ qui capture les schémas séquentiels et temporels et retourne la probabilité d'acquisition des articles de B en période $T + 1$ (figure 1).

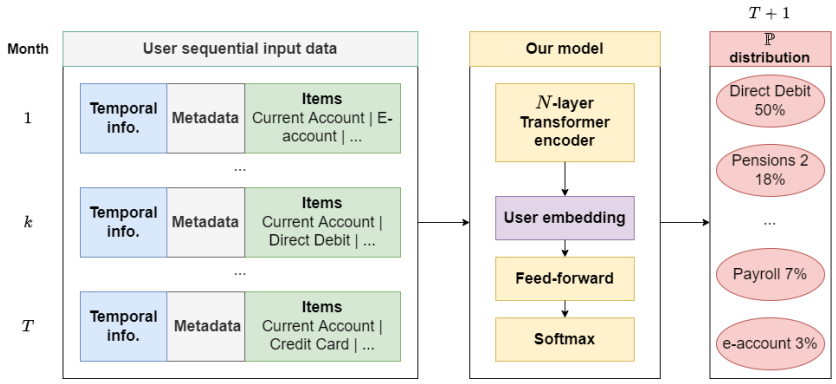

FIG. 1 – *Entrée et sortie du modèle, ainsi que ses différentes composantes.*

4 Méthodes

4.1 Représentation séquentielle des données d'entrée

Étant donné les similitudes avec la tâche de modélisation du langage, nous représentons les périodes de temps sous forme de mots et les données d'une période sous forme de lettres. L'entrée du modèle peut être vue comme une phrase de T mots composés chacun d'un nombre fixe de lettres, correspondant à l'historique de longueur T. La figure 2 montre comment un mot d'une phrase est représenté, dans le cas présent.

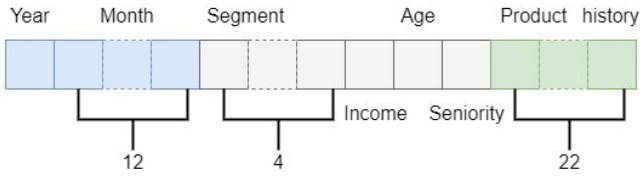

FIG. 2 – *Représentation d'un mois de la séquence d'entrée sous forme de mot d'une phrase.*

Cette représentation temporelle et flexible des données d'entrée peut être employée avec des problèmes et des jeux de données ayant des données temporelles, métadonnées ou historiques d'articles. Elle n'est pas spécifique à, mais est particulièrement utile pour le problème de recommandation en présence d'univers de quelques centaines de produits mais un historique

de plusieurs décennies (matrices creuses), que l'on trouve dans des secteurs tels que la banque, l'assurance, la téléphonie, l'immobilier ou encore le commerce spécialisé en ligne.

4.2 Modèle

D'après l'idée de BERT, nous enlevons le décodeur du Transformer et ne conservons que son encodeur. L'encodeur apprend un plongement des données d'entrée, qu'un réseau de neurones à propagation avant emploie ensuite pour produire la distribution de probabilité sur l'ensemble des articles. Le modèle génère une probabilité $P_b >= 0$ pour chaque article $b \in B$.

L'encodeur est composé de six couches d'attention multi-têtes, avec sept têtes d'attention. Chacune de ces couches est suivie d'une couche de propagation avant de taille 2048. Avant et après la couche de propagation avant demeure une couche *batch-normalization*, et après chaque couche d'encodeur demeure une couche *dropout* avec probabilité 0.5, afin de réduire le sur-ajustement. Les couches d'encodeur sont suivies d'une couche de propagation avant, elle-même suivie d'une couche *softmax*. Cette dernière est utilisée pour produire la probabilité d'acquisition de chaque article à la période suivante, $T + 1$. Puisque notre objectif est de prédire quels nouveaux articles un utilisateur acquerra en plus des articles qu'il possède déjà, nous masquons de P_c^B les articles déjà acquis avant $T + 1$.

5 Jeu de données

5.1 Aperçu du jeu de données

Nous utilisons l'ensemble de données de la compétition Kaggle Santander Product Recommendation [2]–le plus vaste jeu de données accessible au public pour la recommandation de produits de banque de détail. Il comporte 17 mois de données sur un ensemble de clients bancaires (utilisateurs). Les séquences débutent le 2015–01–28 et s'étendent jusqu'au 2016–05–28. Chaque utilisateur dispose d'un relevé mensuel des produits bancaires qu'il détient parmi les 22 de l'univers de produits, produits tels que la carte de crédit ou le compte d'épargne, ainsi que de ses informations personnelles telles que son niveau de salaire, son âge et son sexe.

Les relevés comportent 48 variables pour chaque utilisateur. Parmi celles-ci, 22 correspondent aux produits possédés au cours d'un mois donné (articles), tandis que toutes les autres variables sont des caractéristiques de l'utilisateur (métadonnées). Pour simplifier l'étude et les analyses à suivre, nous conservons seulement quatre variables des métadonnées : `Âge`, `Revenu`, `Ancienneté` (avec la banque) et `Segment`.

5.2 Préparation des données

Le jeu de données couvre 17 mois. Les 16 premiers mois sont utilisés pour l'apprentissage, et le modèle prédit quels nouveaux articles un utilisateur acquerra lors du dix-septième mois. `Segment` est la seule variable des métadonnées qui est encodée en multiples variables binaires. Les trois autres sont des variables continues et sont normalisées entre $[0, 1]$. L'encodage positionnel de la séquence est effectué en attachant la représentation binaire de l'année à un vecteur de longueur 12 représentant en binaire le mois d'un échantillon. Comme le montre

2. `https://www.kaggle.com/c/santander-product-recommendation/`

la figure 2, nous construisons pour chaque utilisateur 16 séquences de longueur 42 correspondant aux mois du 2015–01–28 au 2016–04–28. Il est souhaitable d'inclure les métadonnées à chaque période puisqu'un changement dans les métadonnées peut refléter un changement d'intention de l'utilisateur.

6 Expériences

Nous comparons la performance de notre modèle à celle de systèmes de recommandation de pointe basés sur l'auto-attention, les modèles SASRec, TiSASRec, BERT4Rec et MEANTIME, entraînés grâce aux implémentations mises à disposition avec ce dernier [3].

6.1 Entraînement

Le modèle apprend en utilisant une fonction objectif d'entropie croisée binaire, et l'optimisateur Adam. Pour le taux d'apprentissage, nous adoptons le programme d'échauffement RAdam (Liu et al., 2020) avec un pic après 5 epochs et une baisse jusqu'à la vingtième epoch, quand l'entraînement se conclue. Ce programme d'échauffement assure un taux d'apprentissage initial élevé, lorsque les paramètres sont encore loin de l'optimum, suivi d'une diminution progressive du taux pour un apprentissage stable.

Lors de l'apprentissage, les utilisateurs sont groupés par lots de $k = 32$ utilisateurs. L'entrée de l'encodeur est de taille $k \times dim(T) \times dim(H)$, la sortie (espace vectoriel) $k \times dim(D)$ et la distribution de probabilité finale $k \times dim(B)$.

Nous optimisons le nombre de têtes d'attention, le nombre de couches d'encodeurs, la taille de la couche cachée, le nombre d'epochs d'échauffement et la taille des lots. Nous n'ajustons pas le taux d'apprentissage initial (fixé à 10^{-4}), mais ajustons plutôt le nombre d'epochs d'échauffement. Pour notre modèle, la taille de fenêtre (longueur des séquences qu'un modèle reçoit en entrée) et la longueur maximale de la séquence ont toutes deux été fixées 70. Le temps d'apprentissage de notre modèle s'élève à 3-4 minutes et la complexité est comparable à celle de BERT, ayant tous deux la même architecture d'encodeur attentif.

Les modèles de pointe sont entraînés sur le même jeu de données pendant 10 à 20 epochs, tant que la performance ne cesse d'augmenter. La longueur maximale de la séquence est fixée à 70 afin d'inclure l'historique complet des interactions d'un utilisateur avec les articles. Pour ces modèles, les hyperparamètres influençant directement l'apprentissage sont ajustés, mais les valeurs par défaut demeurent inchangées pour ceux liés par exemple à la structure des modèles.

Toutes les expériences sont réalisées en Python avec NumPy version 1.18.5, PyTorch version 1.8.1 et un GPU GeForce GTX TITAN X avec 12212 Mo de mémoire.

6.2 Comparaison de la performance

Pour mesurer la performance de notre modèle, nous considérons Pre@k, la précision top-k, Rec@k, le *recall* (ou rappel) top-k, avec $k = 1, 5, 10$ ainsi que le rang réciproque moyen (MRR) et le gain cumulé actualisé normalisé (NDCG). Les métriques sont rapportées dans le tableau 1. De manière intuitive, Pre@1 doit être inférieure à Pre@5, mais si par exemple nous n'avons que trois articles *ground truth*, Pre@5 est au plus 3/5 mais Pre@1 peut être 1.

3. https://github.com/SungMinCho/MEANTIME

Modèle	Pre@1	Pre@5	Pre@10	Rec@1	Rec@5	Rec@10	MRR	NDCG
Nôtre	**0.9891**	**0.4022**	**0.2157**	**0.6975**	**0.9764**	0.9979	**0.9937**	**0.9941**
No M.	0.9813	0.3995	0.2157	0.6911	0.9726	0.9979	0.9891	0.9895
BERT4.	0.9693	0.3881	0.2125	0.6823	0.9581	0.9912	0.9830	0.9796
SASR.	0.9600	0.3936	0.2154	0.6736	0.9665	0.9981	0.9781	0.9782
TiSAS.	0.9599	0.3937	0.2155	0.6735	0.9671	**0.9985**	0.9781	0.9784
MEAN.	0.9631	0.3811	0.2125	0.6792	0.9485	0.9912	0.9791	0.9724

TAB. 1 – *Performance des modèles. No M. : Nôtre, en excluant les méta-données de l'entrée.*

Métrique	Toutes entrées	Pas de méta.	Métadonnées seules	Historique court
Pre@1	**0.9891**	0.9813	0.5826	0.7931
Pre@5	**0.4022**	0.3995	0.2451	0.2463
Pre@10	**0.2157**	**0.2157**	0.1653	0.1514
Rec@1	**0.6975**	0.6911	0.4216	0.6326
Rec@5	**0.9764**	0.9726	0.7451	0.7428
Rec@10	**0.9979**	**0.9979**	0.9137	0.8330
MRR	**0.9937**	0.9891	0.7452	0.8492
NDCG	**0.9941**	0.9895	0.7711	0.8510

TAB. 2 – *Sensibilité de la performance du modèle à différentes combinaisons d'entrées.*

Notre modèle surpasse les modèles de pointe basés sur l'auto-attention d'après toutes les métriques sauf Rec@10, d'une marge modeste ($< 2\%$). La représentation des données d'entrée est plus adaptée à un nombre relativement faible de périodes, chacune associée à plusieurs articles, et accepte des métadonnées temporelles (ou statiques) en entrée, ce qui améliore la performance, notamment les métriques Pre@1 et Rec@1. Les modèles concurrents, cependant, ne peuvent structurellement pas utiliser les métadonnées. Dans le tableau 1, TISAS est le seul modèle surpassant le nôtre, mais d'après une seule métrique, Rec@10. Son plongement temporel semble également être efficace en présence de nombreux articles à chaque période.

Lorsque notre modèle ne reçoit que l'historique des produits en entrée, excluant ainsi les métadonnées, il surpasse toujours les modèles de pointe de la même manière (d'après toutes les métriques sauf une, Rec@10). Fait intéressant, nous observons dans le tableau 2 que la seule présence de métadonnées en entrée, sans l'historique des articles, suffit à produire des recommandations, mais non sans perte de précision.

7 Outre les métriques de pertinence

Le tableau 3 présente une évaluation, outre les métriques de pertinence courantes, se concentrant sur l'utilisateur : la sérendipité, la nouveauté et la couverture (Kotkov et al., 2016).

Bien que notre modèle soit supérieur en termes de pertinence des recommandations, il n'en va pas de même pour ces métriques alternatives : les autres modèles surpassent le nôtre d'après certaines métriques, mais aucun modèle ne figure clairement en tête. TiSASRec est le seul modèle à surpasser le nôtre d'après une métrique de pertinence, mais il produit la pire nouveauté de tous, dénotant un dilemme entre nouveauté et précision.

...@k, $k = 1$	Sérendipité	Nouveauté	Couverture
Notre modèle	0.198	0.731	0.818
BERT4Rec	0.184	0.814	**0.909**
SASRec	0.199	0.794	0.681
TiSASRec	**0.224**	0.775	0.727
MEANTIME	0.171	**0.847**	**0.909**
$k = 5$			
Notre modèle	0.075	0.558	1.000
BERT4Rec	0.072	**0.638**	1.000
SASRec	0.078	0.593	0.954
TiSASRec	**0.080**	0.557	1.000
MEANTIME	0.075	0.609	1.000

TAB. 3 – *Performance de notre modèle et des modèles de pointe, au-delà de la pertinence.*

8 Conclusion

Dans cette étude, nous modélisons le problème de recommandation comme une tâche de séquence-à-article et proposons un modèle auto-attentif associé à une représentation novatrice des données d'entrée séquentielles et du contexte de l'utilisateur. Par analogie à la modélisation du langage, pour chaque utilisateur, nous représentons les articles qu'il détient et ses métadonnées sous forme de « lettres » et les périodes sous forme de « mots » pour former des « phrases » tenant compte du contexte temporel de l'historique des articles et des métadonnées.

Notre modèle surpasse les modèles de pointe basés sur l'auto-attention tels que BERT4Rec, à la tâche de recommandation de produits de banque de détail (jeu de données Santander), pour toutes les métriques sauf une. Bien que la capacité à prendre en entrée des métadonnées temporelles distingue notre modèle, il surpasse toujours les autres lorsque privé de ces dernières. Cependant, la prise en compte de métriques outre la pertinence, telles que la sérendipité, brosse un tableau plus nuancé : aucun modèle ne domine d'après toutes les métriques.

Un tel système de recommandation de produits bancaires soulève un certain nombre de considérations éthiques. L'opacité des recommandations peut entraîner une perte de confiance envers le moteur de recommandation ou l'établissement bancaire, ou pire, un usage inapproprié des produits bancaires peut mener à des pertes financières. L'équité des recommandations envers tous les groupes doit être assurée, et les potentiels effets néfastes sur la société, tels que l'homogénéisation des comportements, doivent être surveillés et mitigés.

Les pistes d'amélioration de notre modèle incluent l'ajout d'un terme « au-delà de la pertinence » à la fonction objectif lors de l'apprentissage, ou la modification de l'architecture pour tenir compte du retour des utilisateurs ou de contraintes supplémentaires telles que la valeur monétaire ou l'impact social des produits recommandés.

Le modèle proposé produit un plongement riche et continu des utilisateurs grâce à son encodeur. Cette représentation peut faciliter la segmentation et le profilage de la clientèle, ou être utilisée en amont d'autres applications à forte valeur ajoutée.

Références

Cho, S. M., E. Park, et S. Yoo (2020). MEANTIME : Mixture of attention mechanisms with multi-temporal embeddings for sequential recommendation. In *Fourteenth ACM Conference on Recommender Systems*. ACM.

Kang, W.-C. et J. McAuley (2018). Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206.

Kotkov, D., S. Wang, et J. Veijalainen (2016). A survey of serendipity in recommender systems. *Knowledge-Based Systems 111*, 180–192.

Li, J., Y. Wang, et J. McAuley (2020). Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, New York, NY, USA, pp. 322–330. Association for Computing Machinery.

Liu, D., G. P. Farajalla, et A. Boulenger (2022). Brec the bank : Context-aware self-attentive encoder for banking products recommendation. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8.

Liu, L., H. Jiang, P. He, W. Chen, X. Liu, J. Gao, et J. Han (2020). On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*.

Schafer, J. B., J. Konstan, et J. Riedl (1999). Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pp. 158–166.

Sun, F., J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, et P. Jiang (2019). Bert4rec : Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1441–1450.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, et I. Polosukhin (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Volume 30. Curran Associates, Inc.

Summary

Credit cards, deposits, loans, pension funds, mutual funds–which of these products are relevant to a bank's clients, and at what time in their banking journey? We propose a modeling framework for item recommendation using a multi-head self-attentive encoder and a novel sequential input data representation accounting for the temporal context of both item ownership and user metadata. We evaluate our model on a large public dataset from Santander, and achieve a top-1 and top-5 precision of 98.9% and 40.2%, respectively, thereby improving upon a number of state-of-the-art models. Further, we consider serendipity, novelty and coverage to exhibit a trade-off with recommendation relevance. The continuous user representation learned by our model may inform decisions far more impactful than the recommendations themselves.