Apprentissage de trajectoires temporelles pour la recommandation dans les communautés d'utilisateurs

Elie Guàrdia-Sebaoun[1], Vincent Guigue[1], and Patrick Gallinari[1]

[1]Laboratoire d'informatique de Paris 6 (LIP6), CNRS UMR 7606 , Sorbonne-Universités, UPMC, Paris 6

Résumé

Les systèmes de recommandation sont devenus centraux, non seulement dans le e-commerce mais dans la façon même dont on accède à l'information. Des problèmes de CRM aux réseaux sociaux, la personnalisation est omniprésente et propose de nouveaux défis scientifiques aussi bien en terme d'efficacité que de passage à l'échelle. Les méthodes à variable latentes tendent à exploiter des informations contextuelles de plus en plus variées (information textuelle, interactions sociales, temps) et permettent la définition de profils utilisateur de plus en plus riches. Cet article se concentre sur les aspects temporels de la contextualisation et, plus précisément, les dynamiques utilisateur. En caractérisant les transition des utilisateurs entre les items, nous proposons un modèle de recommandation personnalisée capable d'expliquer ses propositions. En prédisant le prochain produit qu'un utilisateur va voir et s'il va ou non l'aimer, notre système devient proactif.

1 Introduction

Dans cet article, nous nous intéressons à la modélisation des dynamiques utilisateur dans la recommandation personnalisée. Au cours des dix dernières années, de nombreuses méthodes à facteurs latents pour la modélisation des profils item et utilisateur ont vu le jour [33, 24]. A l'aide des ces espaces vectoriels et de critères d'apprentissage adaptés, il est possible de caractériser efficacement les utilisateurs comme les items. De plus, une fois ces modèles appris, leurs étapes d'inférence sont rapides et passent facilement à l'échelle, permettant de s'atteler à des problèmes issus du monde réel. Dans ce domaine, les travaux les plus récents tendent à utiliser l'information textuelle [17, 25, 31] ou encore la dimension temporelle [23, 26] pour la définition d'un contexte à la recommandation.

Le temps est une donnée particulièrement intéressante car il existe bien des façons de l'appréhender. On pourra par exemple utiliser des modèles de dérive conceptuelle et ainsi définir des problèmes de mode comme l'évolution au cours du temps de la perception qu'ont les utilisateurs (dans leur ensemble) des item. Dans [23], l'auteur a prouvé la grande influence du temps sur les revues de film et a donc proposé un modèle à facteurs latents qui encode directement les principales tendances. Dans [39], les auteurs ont choisi de différencier les biais temporels à court et à long terme dans leur système de recommandation. Dans [26], les auteurs proposent une approche plus centrée sur l'utilisateur en optant pour un système à niveaux d'expérience. Ainsi, le temps n'est plus une variable continue et le problème est maintenant de savoir quand un utilisateur est censé gagner un niveau. Enfin, dans [40], les auteurs proposent de voir cette évolution comme une trajectoire dans l'espace latent. Leur système repose sur la modélisation des transitions entre les items (ou des catégories d'items) à l'aide d'un système bayésien. Ils préconisent aussi l'utilisation du rappel@K comme mesure d'évaluation, considérant la tâche de recommandation comme la capacité du système à prédire les prochains items dans la trace de l'utilisateur.

Dans cet article, nous proposons de pousser plus loin cette philosophie en considérant la tâche de recommandation dans sa globalité. Nous considérons qu'en plus de fournir une liste d'items pertinents, un système de recommandation doit être en mesure d'expliquer ses choix à l'utilisateur. Concrètement, le système doit être capable de prédire vers quel item un utilisateur se dirige, s'il l'aimera ou non et, enfin, agir en conséquence. Un réponse type serait alors *"Je pense que vous allez consulter XXX, or vous risquez de ne pas le trouver à votre goût, essayez plutôt YYY ou ZZZ"*.

Tout d'abord, nous considérons le premier aspect du problème, la prédiction d'items. Nous proposons d'évaluer diverses stratégies de modélisation des utilisateurs. Elles ont cependant toutes un point commun : le concept de temps est remplacé par celui de séquence et l'utilisateur est modélisé comme un mouvement au sein de l'espace de représentation des items. Si une régression globale sur l'ensemble des items vus précédemment semble peu performante, une modélisation de la transition entre deux items par translation semble prometteuse. Nous proposons aussi une extension à ce modèle permettant la prise en compte d'information communautaire. Pour ce faire, nous segmentons la population d'utilisateurs en plusieurs communautés. Ainsi, nous pouvons apprendre un endomorphisme de l'espace de représentation des items, nous permettant de mieux tirer parti des préférences de chaque communauté.

Dans un second temps, nous nous intéresserons à la prédiction de notes. Nous proposons d'utiliser les modélisations apprises précédemment pour améliorer les résultats de la factorisation matricielle. En pratique, nous ajoutons aux représentations latentes des items et des utilisateurs des dimensions dédiées à la prédiction de notes. Ces nouveaux coefficients sont optimisés par rapport au critère d'erreur quadratique moyenne.

La prédiction d'items est évaluée suivant le critère du rappel@k. et la prédiction de note est évaluée à l'aide du critère d'erreur quadratique moyenne [23, 26].

La suite de cet article est structurée de la façon suivante : la section 2 est consacrée aux travaux connexes. Les modèles et algorithmes d'apprentissages sont décrits dans les Sections 3 et 4. Dans la Section 5, nous décrivons les datasets et les protocoles expérimentaux. Enfin, nos résultats sont présentés dans la Section 6 et nos conclusions dans la Section 7.

2 Travaux Connexes

Il existe deux façons de définir la tâche de recommandation : la suggestion d'un ensemble d'items pertinents pour un utilisateur [35, 8, 27] ou la prédiction de notes[33, 7, 22, 2].

Quelque soit la méthode d'évaluation, le but principal d'un système de recommandation est d'extraire autant d'information que possible pour la caractérisation des similarités entre les utilisateurs et les items [24, 9]. Or les préférences d'un utilisateur ne sont pas statiques et peuvent beaucoup varier suivant le contexte (*humeur, heure, météo, localisation*) [1]. C'est pourquoi au cours de la dernière décennie, la compréhension de ce contexte [31, 21, 30] et plus particulièrement, celle du temps est devenue un sujet de recherche majeur dans la recommandation. Les données temporelles présentent un grand avantage par rapport aux autres dimensions de contextualisation : elles sont faciles à récupérer (des horodatages sont souvent fournis avec les jeux de données). De plus, le temps peut être naturellement divisé suivant de nombreuses granularités différentes (*heure, jour de la semaine ou du mois, mois, saison, année, etc.*) et ses interprétations sont variées. On pourra par exemple détecter une évolution dans les préférences utilisateur [39, 23], ou encore des phénomènes de périodicité [11].

Il faut toutefois souligner que l'appréhension des dynamiques temporelles dans la recommandation n'est pas chose aisée. Par exemple, l'utilisation d'une décroissance exponentielle peut aussi bien améliorer [15] que dégrader [23] les performances du système. pour affiner les profils utilisateur, les auteurs proposent dans [5] de partitionner les données à l'aide de fenêtres temporelles . Cependant, lorsqu'on s'attendrait à obtenir des résultats optimaux pour des coupes suivant des cycles naturels (*jour/nuit, semaine/week-end, etc.*), il s'avère que les meilleurs scores étaient atteints pour des coupes aléatoires.

Dans [10], les auteurs proposent une étude sur les différentes façons d'utiliser le temps dans la recommandation. Ces méthodes peuvent être divisées en deux grandes familles : les méthodes à mémoire [15, 36] et celles à modélisation [23, 26, 40]. Alors que la première utilise directement les exemples disponible dans l'ensemble d'apprentissage pour calculer une prédiction, les méthodes à modélisation utilisent ces exemples pour apprendre des représentations. Ainsi, on pourra citer

deux contributions majeures : dans [40], les auteurs utilisent des filtres de Kalman pour représenter les transitions a cours du temps entre différentes représentations latentes de l'utilisateur. Dans [26], les auteurs décident de discrétiser le temps en niveaux d'expérience. Les représentations des items et des utilisateurs varient donc au fur et à mesure que ce dernier gagne en expérience. Vu la proximité entre cette approche et la nôtre, nous l'utilisons comme modèle de référence lors de nos expériences.

Dans cet article, nous avons décidé de représenter le temps à l'aide de traces, soit une séquence ordonnée d'événements. Nous avons décidé d'utiliser les séquences pour caractériser la dimension dynamique de l'utilisateur. Ce dernier est caractérisé par son cheminement dans l'espace de représentation des items. L'apprentissage de trajectoires est un sujet qui a été massivement étudié dans le cadre de problèmes de robotique [3] ou de vision par ordinateur : par exemple, dans [13, 32] les auteurs proposent différentes méthodes pour déduire les trajectoires de véhicules à partir d'images de caméras de vidéoprotection. De plus, au cours des dernières années, ont paru quelques articles s'intéressant à la modélisation de trajectoire pour la recommandation de listes de lecture musicales [12], ou de parcours touristiques [18, 6].

Dans la plupart des articles, la découpe des jeux de données entre apprentissage et expérimentation se fait de manière aléatoire [37]. Ce cadre présente l'inconvénient d'être très éloigné des condition réelles d'utilisation d'un système de recommandation en ligne [20]. De plus dans cet article, nous considérons des problèmes de recommandation temporellement contextualisées; il est donc clair que ce type de découpage est inadapté. Cependant, plusieurs réglages sont envisageables : doit-on choisir arbitrairement une date limite, considérant tout événement antérieur (*resp.* postérieur) comme faisant partie de l'ensemble d'apprentissage (*resp.* d'expérimentation) [29, 40]? Ou doit-on considérer la coupe comme relative à chaque trace, ne prenant que les n derniers événements dans l'ensemble d'expérimentation, comme cela a été fait lors de la compétition Netflix[1] [7]? La réponse à ces questions est intrinsèquement subjective et dépend grandement de notre façon de considérer le problème, c'est pourquoi nous ne proposons pas de meilleure méthodologie. Toutefois, dans le cas présent, nous avons opté pour un découpage à date fixe pour éviter d'être confrontés à des problèmes d'effets de mode.

[1] http://www.netflixprize.com/

3 Prédiction d'Items

Cette partie commence par une définition des notations utilisées dans l'article. Ensuite nous décrirons les modèles de référence utilisés pour l'évaluation. Enfin, nous aborderons nos contributions : nos modèles de prédiction d'items à trajectoires.

Par la suite, nous utiliserons les notation suivantes :

- Les items sont indexés à l'aide de la variable i et rassemblés dans un ensemble $I = \{i_k | k \in [\![1, N]\!]\}$.
- Les utilisateurs sont indexés à l'aide de la variable u et rassemblés dans un ensemble $U = \{u_k | k \in [\![1, M]\!]\}$.
- Le temps (au sens d'ordre) est représenté par la variable t.
- A chaque utilisateur u correspond une trace (*i.e.* une séquence ordonnée d'items) $\theta_u = \{i_t | i \in I\}$. Ces traces sont rassemblées dans un ensemble $\Theta = \{\theta_u | u \in U\}$.
- On dénote $\phi_i \in \mathbb{R}^d$ la représentation latente à d dimensions d'un item.
- La transformation dans l'espace des items associée à un utilisateur est notée Ψ_u. Dans une trace θ_u, on suppose que $\Psi_u(\phi_{i_t})$ est une bonne approximation de $\phi_{i_{t+1}}$.
- On appelle *next*, la fonction de prédiction. A l'aide d'un oracle, nous aurions $next(i_{t-1}) = i_t$. La plus part du temps, *next* dépend directement de Ψ_u.
- Soit r_{ui} la note donnée par u à i.

3.1 RankALS

Pour évaluer de manière efficace notre modèle et l'apport de la dynamique temporelle, nous proposons ici un modèle indépendant du temps, RankALS [38]. Ce modèle de prédiction du prochain item, basé sur l'utilisation des

notes, permet la prise en charge des retours implicites (*implicit feedback*). Il consiste en l'application d'un critère d'ordonnancement à la fonction de coût suivante :

$$\mathcal{L} = \sum_{\theta} \sum_{(i,j)\in\theta} c_{ui} \sum_{f\in I\setminus j} s_j \left((\hat{r_{ui}} - \hat{r_{uj}}) - (r_{ui} - r_{uj})\right)^2 \tag{1}$$

où c_{ui} et s_{ui} sont les paramètres de la fonction objectif et $\hat{r_{ui}}$ est calculé à l'aide de la méthode classique de factorisation matricielle.

L'inférence est ensuite effectuée en récupérant les items ayant le meilleur score (calculé avec la formule de factorisation matricielle classique).

3.2 Modèles de Trajectoires Générales

Le calcul des représentations latentes des itemps en fonction de leur place dans les traces est une tâche située à la croisée de différents domaines comme le tourisme, la prédiction de liste de lecture [12], les problèmes origine-destination [34] et l'exploration de données textuelles [28]. Dans cette section, nous nous intéressons à deux modèles de référence pour l'apprentissage des représentation des items.

Modèle Latent à Entrée/Sortie Ce modèle à entrée-sortie est inspiré de [12]. On associe à chaque item deux représentations. Une correspondant à un point d'entrée (représentation de l'item *a priori*) , et l'autre à un point de sortie (représentation de l'item *a posteriori*). Ainsi, on optimise la position des items dans l'espace latent de façon à minimiser la distance parcourue lors d'une transition entre deux items. Soit ϕ^{in} et ϕ^{out} les représentations de chaque item,, on minimise alors le critère suivant :

$$\mathcal{L} = \sum_{\theta} \sum_{(i,j)\in\theta} \sum_{f\in I\setminus j} (||\phi_i^{out} - \phi_j^{in}|| - ||\phi_i^{out} - \phi_f^{in}||) \tag{2}$$
avec j suivant i dans θ

L'inférence est alors la même pour tous les utilisateurs. Elle est calculée à par minimisation de la distance euclidienne :

$$next_{IOLM}(i) = \underset{j\in I\setminus i}{\operatorname{argmin}} \|\phi_i^{out} - \phi_j^{in}\| \tag{3}$$

modèle CBOW Dans [28], les auteurs ont proposé une méthode efficace pour modéliser les séquences de mots : le sac de mots continu (CBOW). L'idée générale consiste en la prédiction de la représentation d'un mot, connaissant celle de son contexte. Ici, nous avons transposé notre problème aux séquences d'items, gardant l'algorithme inchangé. L'inférence est encore une fois effectuée par recherche des plus proches voisins :

$$next_{CBOW}(i) = \underset{j\in I\setminus i}{\operatorname{argmin}} \|\phi_i - \phi_j\| \tag{4}$$

3.3 Personnalisation

Nous proposons maintenant d'ajouter une personnalisation des résultats. Pour ce faire nous définissons un vecteur de translation propre à chaque utilisateur. Ce vecteur représente son comportement dans l'espace de représentation des items.

$$\Psi_u(\phi_{i_t}) = \phi_{i_t} + b_u, \qquad b_u \in \mathbb{R}^d \tag{5}$$

Ainsi, nous utilisons les représentations des items comme des points d'ancrage, et concentrons l'apprentissage sur les vecteurs b_u. En considérant la fonction de coût on définit ce problème comme une minimisation de distance euclidienne :

$$\mathcal{L} = \sum_{\substack{\theta\in\Theta \\ i\in\theta \\ j\in I\setminus i}} \|\Psi_u(\phi_{i_{t-1}}) - \phi_j\|^2 + \lambda\Omega(\Psi) \tag{6}$$

Nous proposons alors une résolution par récursion. Cette méthode a deux avantages : en plus d'être rapide, elle permet de gérer artificiellement l'amortissement temporel de la mémoire du modèle (à l'aide du paramètre α):

$$\forall u, i_t = \theta_{u,t}, \qquad (b_u)_{t+1} = \alpha(b_u)_t + (1-\alpha)(\phi_{i_{t+1}} - \phi_{i_t}), \\ (b_u)_0 = 0^d, \qquad b_u \in \mathbb{R}^d, \qquad \alpha \in [0,1] \tag{7}$$

L'inférence s'effectue encore une fois par recherche des plus proches voisins de l'item courant.

$$next(i,u) = \underset{j\in I}{\operatorname{argmin}} \left\|\Psi_u(\phi_i) - \phi_j\right\| \tag{8}$$

3.4 Trajectoires et Communautés

Lorsque nous avons essayé des modèles plus complexes, nous nous sommes heurtés à un problème de données. Les

traces utilisateur étaient en moyenne trop courte pour apprendre efficacement les paramètres, devenus trop nombreux. Il a donc fallu trouver une méthode d'agrégation efficace des utilisateurs. Ainsi, nous avons tout naturellement ajouté un contexte communautaire. Pour ce faire, nous avons segmenté la base utilisateur en fonction de leurs traces. Pour chaque communauté ainsi définie, nous avons appris une métrique sur l'espace des items permettant d'adapter leurs représentations aux goûts des communautés. La métrique associée à une communauté c est la matrice A_c, de dimension $d \times d$.

extraction de communautés Chaque utilisateur est représenté par un vecteur cl_u définit comme il suit :

$$\forall u \in U \begin{cases} cl_u[i] = 1 & ssi \;\; i \in \theta_u \\ cl_u[i] = 0 & ssi \;\; i \notin \theta_u \end{cases} \tag{9}$$

Pour mettre en évidence l'importance de l'information disponible dans les communautés, nous avons décider d'utiliser une méthode de segmentation aussi simple et naïve que possible, c'est pourquoi nous avons opté pour l'algorithme des k-moyennes. On appelle c_u la communauté associée à l'utilisateur u.

utilisation des communautés Soit c_u la communauté associée à l'utilisateur u, on applique sa métrique à l'ensemble de l'espace de représentation des items. On définit alors une version des vecteurs item dépendante de la communauté de la façon suivante :

$$\phi_{i_{c_u}} = A_{c_u}\phi_i, \qquad Ac_u \in M_n(\mathbb{R}) \tag{10}$$

Ici, la matrice A_c représente l'endomorphisme associé aux catégories. Le modèle est alors appris par descente de gradient sur une version mise à jour de la fonction Eq.6:

$$\mathcal{L} = \sum_{\substack{\theta \in \Theta \\ i \in \theta \\ j \in I \setminus i}} \big[1 - \|\Psi_u(\phi_{i_{c_u,t-1}}) - \phi_{j_{c_u}}\|^2 + \|\Psi_u(\phi_{i_{c_u,t-1}}) - \phi_{i_{c_u,t}}\|^2 \;\big]^+ + \lambda\Omega(\Psi) \tag{11}$$

Le terme de régularisation est définit comme il suit :

$$\Omega(\Psi) = \sum_{u \in U} \|b_u\|^2 + \|A_c\|_F^2 \tag{12}$$

4 Prédiction de Notes

Dans cette section, nous présentons d'abord trois modèles issus de la littérature : la factorisation matricielle [24], timeSVD [23], et un modèle à niveaux d'expérience [26], puis notre contribution.

On ajoute aux notations précédentes :

- Les représentation enrichies des utilisateurs (*resp.* items) sont nommées γ_u (*resp.*γ_i)
- Elles sont construites en augmentant la dimension des représentations apprises précédemment. Ces nouveaux facteurs sont respectivement notés $\bar{\gamma}_u$ et $\bar{\gamma}_i$.
- μ, μ_u et μ_i représentent respectivement les biais général, utilisateur et item.

  En suivant cette formulation, γ sont en dimension d' avec $d' = 2d$.

4.1 Factorisation Matricielle et Extensions

Dans le cadre de la prédiction de notes, la factorisation matricielle (MF) est un modèle de référence incontournable. Pour quantifier l'impact du contexte temporel, nous avons choisi de présenter deux modèles.

TimeSVD (TSVD), qui correspond à l'incorporation dans MF de termes dépendant du temps [23]. Nous avons utilisé l'implémentation disponible dans la librairie Dato[2].

Le modèle à niveaux d'expériences (EXP) présente l'intérêt d'avoir une philosophie proche de celle proposée dans cet article : ici le temps est replacé par des niveaux d'expérience discret (ce qui se rapproche de notre notion d'ordonnancement) [26]. Ce modèle peut être vu comme un modèle de Markov caché où chaque état est représenté par un problème de factorisation matricielle distinct.

La difficulté de se modèle réside donc dans l'apprentissage des changements de niveau d'expérience. Pour répondre à ce problème, les auteurs proposent l'utilisation un schéma d'optimisation en alternance. Nous avons utilisé notre propre implémentation du modèle pour l'évaluation.

Pour tous les modèles cités précédemment, la note qu'un utilisateur donne à un item est estimée de la façon

[2] https://dato.com/products/create/open_source.html

suivante :

$$\hat{r}_{u,i} = \mu + \mu_u + \mu_i + \langle \gamma_u, \gamma_i \rangle \quad (13)$$

Le problème d'optimisation associé est une minimisation du critère suivant :

$$\mathcal{L} = \sum_{u \in U, i \in I} [\mu + \mu_u + \mu_i + \langle \gamma_u, \gamma_i \rangle - r]^2 + \lambda \Omega(\gamma) \quad (14)$$

Nous utilisons le terme de régularisation L_2 (comme précédemment) $\Omega(\gamma) = \sum_u \|\gamma_u\|^2 + \sum_i \|\gamma_i\|^2$.

4.2 Factorisation Matricielle et Exploitation des Trajectoires

Dans cet article, nous voulons améliorer les résultats de la factorisation matricielle en tirant un maximum d'information des représentations utilisées pour la prédiction d'items. Cette tâche n'est pas évidente car il n'y a pas de corrélation directe entre la trace utilisateur et les notes attribuées aux items. Les nouveaux profils sont définis de la façon suivante :

$$\gamma_u = \begin{bmatrix} \bar{\gamma}_u \\ b_u \end{bmatrix} \in \mathbb{R}^{2d}, \qquad \gamma_i = \begin{bmatrix} \phi_i \\ \bar{\gamma}_i \end{bmatrix} \in \mathbb{R}^{2d} \quad (15)$$

De cette façon, nous assurons une certaine proximité entre les utilisateurs (*resp.* items) avec un b_u (*resp.* ϕ_i) proche sans pour autant changer le principe de la factorisation matricielle (Eq. 13).

Les représentations $\{\bar{\gamma}_u, u \in U\}$ et $\{\bar{\gamma}_i, i \in I\}$ sont initialisées aléatoirement et apprises par descente de gradient sur la fonction de coût classique présentée dans (14). b_u et ϕ_i restent constantes durant tout le processus d'apprentissage pour assurer la proximité des profils similaires.

Cette méthode est très proche de MF. Cependant, le problème d'optimisation est non convexe; nous pensons donc que l'initialisation est un paramètre important. De plus, garder de nombreux coefficients constants donne au modèle une ligne directrice que permet d'améliorer la prédiction de notes.

4.3 Modèle Communautaire

Au modèle précédent, nous ajoutons l'utilisation des endomorphismes de transformation associés à chaque communauté. Les représentations présentées dans (15) deviennent donc :

$$\gamma_u = \begin{bmatrix} \bar{\gamma}_u \\ b_u \end{bmatrix} \in \mathbb{R}^{2d}, \qquad \gamma_i = \begin{bmatrix} A_{c_u} \phi_i \\ \bar{\gamma}_i \end{bmatrix} \in \mathbb{R}^{2d} \quad (16)$$

5 Protocole Expérimental

Dans cette section, nous proposons d'évaluer les performances de nos modèles pour deux tâches différentes : la prédiction d'items, et la prédiction de notes. Dans le premier cas, nous utilisons une mesure de rappel@k, et dans le second, le critère d'erreur quadratique moyenne (MSE).

5.1 Jeux de données

Nous avons effectué nos expériences sur cinq jeux de données : *BeerAdvocate* et *RateBeer* proviennent de [26] et contiennent des revues de bières. MovieLens (10m), (Amazon) Movies et Flixster traitent de films. Les propriétés de chaque jeu de données sont répertoriées dans la Table 1.

Table 1: Propriétés des jeux de données.

Dataset	# items	# users	# ratings
BeerAdvocate	66051	33387	1586259
RateBeer	110419	40213	2924127
MovieLens	10000	72000	10000000
Flixster	49000	1000000	8200000
Movies	253059	889176	7911684

5.2 Cadre Général

Nous avons normalisé les notes sur l'ensemble $[0,5]$ et utilisé le cadre proposé dans [40] : nous n'avons gardé que les traces utilisateur contenant au moins dix revues et chaque jeu de données a été divisé en dix fenêtres temporelles de même durée. Pour apprendre les hyperparamètres du modèle, nous avons utilisé les huit premières fenêtres comme base d'apprentissage et la neuvième comme base de validation. Une fois les hyperparamètres appris, nous avons ajouté la fenêtre de validation à l'ensemble d'apprentissage et utilisé la dernière fenêtre comme ensemble de test. Pour minimiser l'influence des processus stochastiques (l'initialisation

aléatoire et la descente de gradient stochastique associée à un problème non-convexe), nous avons lancé chaque expérience cinq fois et rapporté la moyenne des résultats.

Nous présentons le rappel@k pour k variant entre 5 et 50. Nous pensons que des valeurs plus élevées de k n'ont que peu d'intérêt dans le cadre de la recommandation.

5.3 Modèles de Référence et Paramètres

Les modèles sont développés en détail dans les sections 3 et 4. Nous rappelons ici leurs dénominations et présentons leurs réglages.

Prédiction d'Items Dans le cadre de la prédiction de notes, nous utilisons comme modèles de référence, RankALS [38], un modèle à entrée-sortie (IOLM) et un modèle basé sur une modélisation de l'espace des items par CBOW.

On appelle TRANS le modèle de personnalisation à translation et COMM celui prenant en compte les communautés.

Lorsque l'on utilise un espace latent, il est nécessaire de déterminer sa dimensionnalité optimale. Nos expérimentations sur l'ensemble de validation montrent que l'impact de la dimensionnalité diminue pour $d > 20$. Nous avons donc fixé la dimension à 20 pour toutes nos expériences.

Détermination du Nombre de Communautés Nous avons effectué de nombreuses expériences sur les ensembles de validation pour déterminer le nombre optimal de communautés pour chaque jeu de données. Les résultats sont présentés en Fig. 1.

Prédiction de Notes Pour cette tâche nous avons utilisé trois modèles de référence : la factorisation matricielle classique avec termes de biais telle que proposée dans [24] (MF), une extension temporelle de ce modèle [23] (TSVD) et un modèle à niveaux d'expérience [26] (EXP).

Nos expériences sur l'ensemble de validation nous ont permis de fixer ne nombre de niveaux d'expérience à 5 (comme dans l'article d'origine). En conséquence, ce modèle contient cinq fois plus de paramètres que MF. CBOW est calculé sur un espace latent de dimension vingt TRANS et COMM sont donc en dimension quarante. Dans un souci d'équité, nous avons évalué toutes nos modèles dans un espace de dimension $d = 40$.

6 Results and Discussion

6.1 Prédiction d'Items (Rappel@k)

Pendant nos travaux préliminaires, nous avons considéré plusieurs mesures d'évaluation comme la précision@k ou le rang moyen, cependant ces méthodes semblaient mal adaptées. Le rang moyen est peu robuste et se focalise sur un seul item. De l'autre côté, la précision@k est trop stricte. En effet, un utilisateur peut avoir raté un item simplement parce qu'il ne le connaissait pas et non par manque d'intérêt. Il fallait donc une méthode plus souple pour une évaluation hors ligne. C'est pourquoi nous avons utilisé le rappel@k, comme préconisé dans [40].

Fig. 2 contient les courbes de performance en rappel@k pour les différents modèles. On, remarque que, dans l'ensemble, les modèles temporels (COMM, TRANS, CBOW et IOLM) donnent de meilleurs résultats que RankALS, ce qui montre l'importance du contexte temporel. De plus, CBOW propose de meilleures performances ainsi que moins de paramètres que IOLM (sauf sur le jeu de données Movies).

Les modèles personnalisés (TRANS et COMM) tiennent le haut du tableau, spécialement COMM, qui écrase tous les autres modèles. Cette démonstration de l'impact des informations sur les communautés est encourageante pour la suite de nos travaux.

Table 2: Résultats en prédiction de notes, exprimés en MSE pour les modèles MF, TSVD, EXP, TRANS et COMM.

Dataset	MF	TSVD	EXP	TRANS	COMM
BeerAdvocate	0.4	0.381	0.367	**0.361**	**0.360**
RateBeer	0.331	0.301	0.297	**0.279**	**0.279**
MovieLens	0.691	0.681	0.684	0.663	**0.660**
Flixster	0.912	0.867	0.827	0.816	**0.811**
Movies	1.377	1.211	1.05	**0.913**	**0.913**

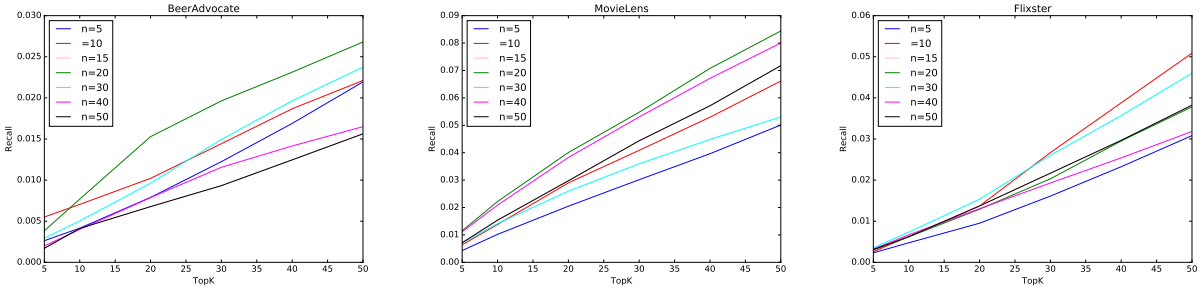

Figure 1: Evolution du rappel@k sur les ensembles de validation pour différents nombres de communautés $n \in [5, 50]$.

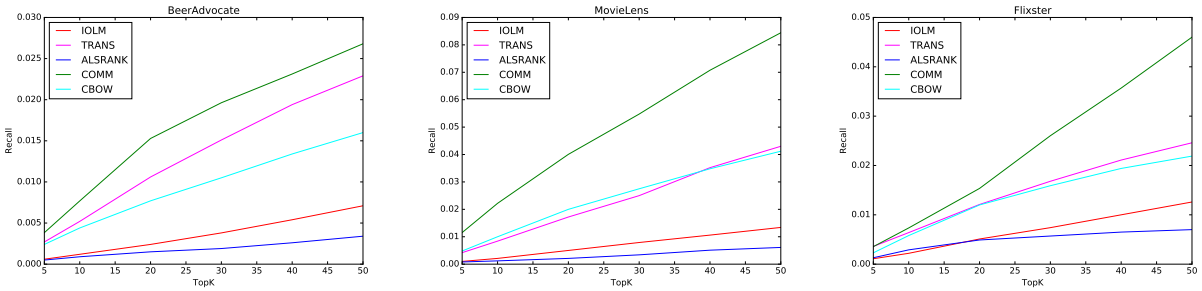

Figure 2: Evolution du rappel@k sur les ensembles de test pour chaque modèle.

6.2 Prédiction de Notes (MSE)

Nous avons utilisé la méthode classique d'évaluation pour la prédiction de notes, le critère d'erreur quadratique moyenne dont l'adéquation n'est plus à démontrer.

Les résultatssont disponibles dans la Table 2. Les meilleurs résultats sont surlignés en gras.

Comme mis en avant dans [23], la marge d'amélioration en terme de MSE est très faible et même un faible gain peut avoir un énorme impact sur la recommandation. Tout d'abord, on remarque que les modèles temporels surpassent clairement MF ce qui encore une fois montre l'intérêt de la prise en compte du contexte temporel. De plus nos modèles surpassent clairement TSVD et EXP, démontrant l'ajout d'information que représente l'utilisation des représentations de trajectoire.

De plus l'ajout des représentations de COMM permet d'améliorer encore les résultats de TRANS sur la plupart des jeux de données. Même si MF fonctionne par segmentation implicite des utilisateurs en communautés, on voit ici que l'ajout d'information explicite améliore encore les performances.

Il est intéressant de se pencher sur la question de la complexité spatiale : TRANS et COMM comptent moitié moins de paramètres que MF (la moitié de leurs paramètres sont constants et appris en amont) alors que EXP en compte n fois plus (avec n le nombre total de niveaux d'expérience). Ceci montre que même la plus simple des approches a assez de degrés de libertés. Le défi n'est pas la complexification, mais plutôt l'adoption de la stratégie la plus intelligente.

Le comportement des modèles d'un jeu de données à l'autre est somme toute stable. Certains présentent un léger gain, mais aucun domaine ne semble particulièrement privilégié par notre approche.

7 Conclusion

Dans cet article, nous avons montré qu'il était possible de tirer beaucoup d'information de la caractérisation des

transitions entre items pour chaque utilisateur, permettant ainsi d'améliorer les performances des modèles de recommandation, aussi bien pour la prédiction d'items que pour la prédiction de notes.

De plus, nous avons prouvé l'intérêt de l'ajout d'un contexte communautaire explicite. En apprenant une métrique propre à chaque communauté, nous avons pu adapter les représentations latentes des items et ainsi faciliter la modélisation des dynamiques de l'utilisateur.

Dans cet article, Nous avons proposé une méthode de recommandation légère et originale. En gardant le nombre de communautés négligeable par rapport au nombre d'utilisateurs, nous avons pu significativement améliorer les résultats des autres modèles présentés tout en gardant une faible complexité.

De plus, notre système est aussi capable de modéliser les intentions de l'utilisateur. Cependant, les résultats sur cette tâche sont plus compliqués à interpréter. En effet, l'évaluation est très fortement pénalisée par le paradigme d'évaluation hors ligne.

References

[1] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 2011.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE TKDE*, 2005.

[3] J. Aleotti and S. Caselli. Robust trajectory learning and approximation for robot programming by demonstration. *Robotics and Autonomous Systems*, 2006.

[4] T. Anagnostopoulos, C. Anagnostopoulos, S. Hadjiefthymiades, M. Kyriakakos, and A. Kalousis. Predicting the location of mobile users: A machine learning approach. In ICPS, 2009.

[5] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *CARS*, 2009.

[6] R. Baraglia, C. I. Muntean, F. M. Nardini, and F. Silvestri. Learnext: learning to predict tourists movements. In *CIKM*, 2013.

[7] J. Bennett and S. Lanning. The netflix prize. In *KDD*, 2007.

[8] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, 1998.

[9] R. Burke. Hybrid web recommender systems. In *The adaptive web*. Springer, 2007.

[10] P. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 2014.

[11] P. G. Campos, F. Diez, and A. Bellogin. Temporal rating habits: A valuable tool for rating discrimination. In *Proceedings of CAMRa*, 2011.

[12] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims. Playlist prediction via metric embedding. In *Proceedings of ACM SIGKDD*, 2012.

[13] B. Coifman, D. Beymer, P. McLauchlan, J. Malik, and J. M. B. A real-time computer vision system for vehicle tracking and traffic surveillance, 1998.

[14] A. K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 2001.

[15] Y. Ding and X. Li. Time weight collaborative filtering. In *Proceedings of CIKM*, 2005.

[16] Y. Ding, S. Liu, J. Pu, and L. M. Ni. Hunts: A trajectory recommendation system for effective and efficient hunting of taxi passengers. In *MDM (1)*, 2013.

[17] G. Ganu, N. Elhadad, and A. Marian. Beyond the stars: Improving rating predictions using review text content. In *WebDB*, 2009.

[18] É. Guàrdia-Sebaoun, V. Guigue, and P. Gallinari. Latent trajectory modeling: Recommandation Dynamique dans les Graphes Géographiques. In *MARAMI*, 2014.

[19] É. Guàrdia-Sebaoun, V. Guigue, and P. Gallinari. Latent trajectory modeling: A light and efficient way to introduce time in recommender systems. In *Proceedings of RecSys*, 2015.

[20] A. Gunawardana and G. Shani. A survey of accuracy evaluation metrics of recommendation tasks. *J. Mach. Learn. Res.*, 2009.

[21] T. Hussein, T. Linder, W. Gaulke, and J. Ziegler. Hybreed: A software framework for developing context-aware hybrid recommender systems. In *User modeling and user adapted interaction*, 2012.

[22] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *ACM SIGKDD*, 2008.

[23] Y. Koren. Collaborative filtering with temporal dynamics. In *ACM SIGKDD*, 2009.

[24] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.

[25] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *RecSys*, 2013.

[26] J. J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *World Wide Web*, 2013.

[27] M. R. McLaughlin and J. L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *ACM SIGIR*, 2004.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, 2013.

[29] U. Panniello, M. Gorgoglione, and C. Palmisano. Comparing pre-filtering and post-filtering approach in a collaborative contextual recommender system: An application to e-commerce. In *EC-Web*, 2009.

[30] U. Panniello, A. Tuzhilin, and M. Gorgoglione. Comparing context-aware recommender systems in terms of accuracy and diversity. *UMUAI*, 2014.

[31] M. Poussevin, V. Guigue, and P. Gallinari. Extended recommendation framework: Generating the text of a user review as a personalized summary. *CoRR*, 2014.

[32] R. Rad and M. Jamzad. Real time classification and tracking of multiple vehicles in highways. *Pattern Recognition Letters*, 2005.

[33] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *ACM CSCW*, 1994.

[34] V. Salnikov, R. Lambiotte, A. Noulas, and C. Mascolo. Openstreetcab: Exploiting taxi mobility patterns in new york city to reduce commuter costs. *CoRR*, 2015.

[35] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender Systems Handbook*, 2011.

[36] S. M. Siniscalchi, J. Reed, T. Svendsen, and C. Lee. Exploiting context-dependency and acoustic resolution of universal speech attribute models in spoken language recognition. In *INTERSPEECH*, 2010.

[37] H. Stormer. Improving e-commerce recommender systems by the identification of seasonal products. In *AAAI*, 2007.

[38] G. Takács and D. Tikk. Alternating least squares for personalized ranking. In *recSys*, 2012.

[39] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In ACM SIGKDD, 2010.

[40] C. Zhang, K. Wang, H. Yu, J. Sun, and E. Lim. Latent factor transition for dynamic collaborative filtering. In *SIAM*, 2014.