Entre factorisation de matrices et apprentissage profond pour la recommandation dans le domaine du pneumatique

Kilian Bourhis**, Khalid Benabdeslem*
Bruno Canitia**

*Université Lyon1, 43 Boulevard du 11 novembre 1918, Villeurbanne cedex 69622
khalid.benabdeslem@univ-lyon1.fr
**Lizeo Online Media Group, 42 Quai Rambaud, 69002 Lyon
kilian.bourhis,bruno.canitia@lizeo-group.com

Résumé. Les moteurs de recommandations ont aujourd'hui une place de plus en plus importante dans l'aiguillage de nos choix de consommation sur internet. Cependant, les données disponibles pour effectuer une recommandation varient selon les utilisateurs, les moyens de l'industrie et le type des produits. Dans cet article, nous nous intéressons d'une part, aux performances des modèles de factorisation de matrice ainsi qu'à l'apprentissage profond, et d'autre part, à la recommandation sur des données massives. Une étude comparative entre plusieurs modèles de l'état de l'art est proposée dans un cadre applicatif lié à un industriel spécialisé dans la captation et la gestion des données sur le marché des pneumatiques.

1 Introduction

Un système de recommandation peut effectuer principalement trois sortes de tâches (Zhang et al. (2017)) : prédire l'évaluation qu'un utilisateur associerait à un produit ; prédire les produits qu'un utilisateur aurait sélectionnés à travers une liste classée de N produits ; et enfin, prédire la classe d'appartenance d'un produit (classification).

Dans notre cas, la recommandation de pneumatiques, nous visons à proposer un classement de pneus pertinents vis-à-vis des utilisateurs. Pour cela, nous devrons aussi, selon l'approche utilisée, prédire l'évaluation que donnerait un utilisateur envers les produits. Cependant, le cas particulier d'un comparateur en ligne de pneumatiques, qui est notre cas applicatif, nous impose des contraintes au niveau des données. En effet, nous n'avons à notre disposition aucun profil d'utilisateur, aucune évaluation explicite d'un utilisateur envers un produit, ni aucune information validant le fait qu'un utilisateur ait bien acheté un produit après avoir été redirigé vers un site marchand. Nous devons donc nous appuyer sur les éléments restants, à savoir les interactions implicites des utilisateurs ainsi que leurs parcours (sessions) et les caractéristiques techniques des produits. Ces contraintes, ainsi que notre volonté d'explorer les modèles de factorisation de matrices et d'apprentissage profond, nous ont amenés à effectuer les choix suivants parmi les principales approches de l'état de l'art (*cf.* fig. 1).

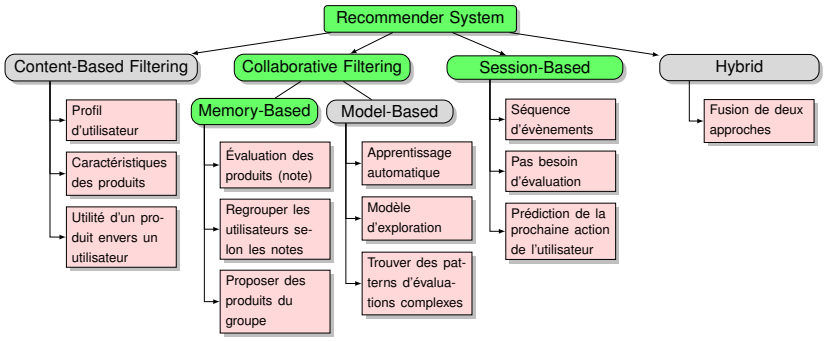


FIG. 1: Principales approches de la recommandation.

Deux approches ont ainsi retenu notre attention. Le **Memory-Based Collaborative Filtering** (*e.g.* Hu et al. (2008); Gong (2010); Bokde et al. (2015); He et al. (2016, 2017)) qui se base sur l'hypothèse suivante : les utilisateurs ont tendance à acheter les mêmes produits que d'autres utilisateurs ayant des goûts similaires. On veut ainsi regrouper les utilisateurs similaires, afin de proposer à un utilisateur des produits qu'il na pas encore consultés mais avec lesquels les autres membres de son groupe ont interagi. Pour déterminer si des utilisateurs ont des goûts similaires, on peut par exemple comparer le score qu'ils ont attribué aux produits. On dispose alors de la note d'un utilisateur envers un produit (explicite) ou on la déduit des interactions (implicite). Enfin l'approche **Session-Based** (*e.g.* Hidasi et al. (2015); Quadrana et al. (2017); Tuan et Phuong (2017)), qui consiste à traiter le problème de la recommandation sous la forme d'une séquence d'événements (ordonnés chronologiquement) où l'objectif est de prédire l'élément suivant pour chaque interaction. Contrairement à une vision plus classique, cette méthode a l'avantage de pouvoir se passer des évaluations que les utilisateurs peuvent donner aux produits. Ainsi, elle se concentre uniquement sur les produits avec lesquels les utilisateurs vont interagir et de quelle manière. Dans cet article, nous présentons dans un premier temps les méthodes d'apprentissage qui feront l'objet d'une comparaison dans l'étude. Puis, nous détaillerons les données et le protocole expérimental utilisés. Enfin, nous terminerons par les résultats de notre étude comparative et statistique.

2 Méthodes d'apprentissage pour la recommandation

Notre objectif, est d'étudier les performances de ces deux approches à travers des techniques de factorisation de matrices et d'apprentissage profond. Notre choix s'est porté sur cinq modèles issus de l'état de l'art car ils présentent des caractéristiques variés et n'ont pas été évalués dans des conditions similaires (métriques, taille du classement et nombre de produits).

	Approche	Type de données	Modèles	Références
eALS [1]	Memory-Based CF	Implicite	FM	He et al. (2016)
NeuMF [2]	Memory-Based CF	Implicite	FM + MLP	He et al. (2017)
DMF [3]	Memory-Based CF	Explicite	FM + DSSM	Xue et al. (2017)
GRU4Rec [4]	Session-Based	Implicite	RNN GRU	Hidasi et al. (2015)
3D CNN [5]	Session-Based	Implicite + Annexe	CNN	Tuan et Phuong (2017)

TAB. 1: Résumé des caractéristiques des modèles sélectionnés pour l'étude.

Le premier modèle, **Element-wise Alternating Least Square**, possède deux implémentations. Un algorithme *offline* qui sert à pré-entraîner le modèle grâce aux données de départ et un algorithme *online* pour traiter le nouveau flux de données en temps réel. Mais, la particularité de ce modèle est sa gestion des produits populaires. On entend par *populaire* le fait qu'un produit soit très présent sur l'ensemble des interactions utilisateur-produit. Pour cela, l'eALS calcule la *confidence* d'un produit, soit la probabilité qu'un produit n'ait pas été vu par les utilisateurs. En effet, plus un produit est populaire, plus les chances qu'il soit recommandé (ou du moins visible sur la page) pour les utilisateurs sont importantes. Dans ce cas, il faut apporter un poids plus important au fait que l'utilisateur n'ait pas interagi avec lui, non pas parce qu'il ne l'a pas vu, mais bien parce qu'il ne l'intéresse pas.

Le **Neural Collaborative Filtering**, est une hybridation de deux modèles pouvant être utilisés indépendamment pour produire une recommandation. Le *GMF* (Generalized Matrix Factorization) va modéliser les facteurs latents des interactions de manière linéaire. Le *MLP* (Multi-Layer Perceptron) va apprendre la fonction d'interaction, de manière non-linéaire, afin de modéliser lui aussi les interactions utilisateur-produit. Enfin, une fois que l'on a les poids de sortie de chacun des deux modèles, on les concatène pour produire le score d'un utilisateur envers un produit.

Le **Deep Matrix Factorization** est lui aussi une hybridation, mais entre une factorisation de matrices et un réseau de neurones *DSSM* (Deep Semantic Similarity Model), qui ne sont pas indépendants car le DSSM travaille sur la sortie de la factorisation de matrices (matrice d'interaction). La particularité du DSSM est qu'il peut calculer le vecteur latent d'un utilisateur sur une «branche» et le vecteur latent d'un produit sur une deuxième. Chaque branche va ainsi transposer le vecteur dans un nouvel espace à plus faible dimension, au fur et à mesure des couches. La fonction *cosinus* sera ensuite appliquée sur la concaténation de la sortie de chaque branche pour calculer le score d'une interaction utilisateur-produit.

Le **GRU4Rec** est un *RNN* (Recurrent Neural Network) utilisant l'architecture GRU (Gated Recurrent Unit). Ses différentes mémoires et portes lui permettent de traiter des séquences d'événements. Ainsi, l'idée est de prédire le prochain produit avec lequel l'utilisateur va interagir en fonction des produits précédents. Comme ce modèle utilise des interactions explicites, nous avons recréé artificiellement des données explicites via nos données implicites en attribuant des poids à chaque type d'interaction utilisateur-produit (*cf.* section 3 pour les types d'interactions). Ces poids agissent ainsi comme une note qu'aurait attribué un utilisateur envers un produit. Nous avons attribué ces poids de façon arbitraire en fonction de la proximité du type d'interaction avec un achat potentiel.

Enfin, le **3D CNN** (Convolutional Neural Network) est un réseau de neurones qui utilise des convolutions en trois dimensions, afin d'encoder l'aspect temporel d'une séquence d'interactions entre un utilisateur et des produits. L'objectif est le même que pour le GRU4Rec, à la différence que l'on n'utilise pas seulement l'ID des produits, mais aussi des informations annexes. Dans notre implémentation, nous utilisons la nature de l'interaction utilisateur-produit ainsi que l'ID de l'utilisateur à qui appartient la session, afin de faire le lien avec d'autres sessions pouvant lui appartenir.

1. Code disponible ici : https://github.com/hexiangnan/sigir16-eals
2. Code disponible ici : https://github.com/hexiangnan/neural_collaborative_filtering
3. Code disponible ici : https://github.com/RuidongZ/Deep_Matrix_Factorization_Models
4. Code disponible ici : https://github.com/hidasib/GRU4Rec
5. Le code d'origine ainsi que notre réimplémentation sont non publics.

3 Étude comparative sur des données de pneumatique

Nos jeux de données proviennent tous d'un fichier log du comparateur en ligne de pneumatique de l'industriel. Ce fichier répertorie l'ensemble des interactions utilisateur-produit depuis septembre 2016 jusqu'en avril 2018. Le tableau suivant résume leur volumétrie.

	eALS	NeuMF	DMF	GRU4Rec [6]	3D CNN	
Interaction	61 204	61 204	77 864	62 490	79 547	**Pattern Full**
Produit	4 260	4 260	4 281	4 192	4 216	
Utilisateur	24 761	24 761	30 626	25 169	29 611	
Session	-	-	-	25 169	31 881	
Interaction	59 007	59 007	75 789	60 223	77 470	**Product Full**
Produit	3 278	3 278	3 288	3 197	3 211	
Utilisateur	23 924	23 924	30 154	24 237	28 814	
Session	-	-	-	24 237	31 038	

TAB. 2: Volumétrie des deux jeux de données principaux.

Parmi les types d'interaction utilisateur-produit possible, nous en avons sélectionné cinq selon leur présence dans le fichier log et leur pertinence pour la recommandation (présentés par ordre croissant de leur poids) :
InfoProduit : Affichage de la fiche détaillée du produit cible. Quelques caractéristiques sont déjà visibles pour le top 3 des produits recommandés.
VoirOffre : Affichage des revendeurs qui vendent le produit cible avec leur localisation.
InfoMarchand : Affichage d'une fiche avec les informations détaillées du revendeur.
AfficherNuméroMarchand : Génération puis affichage d'un numéro pour appeler le centre du revendeur. N'indique en aucun cas si l'utilisateur a bien appelé le centre.
AcheterEnLigne : Redirection vers le site du revendeur. N'indique en aucun cas si l'utilisateur a bien acheté le produit par la suite.

Enfin, pour mener à bien des tests statistiques, nous avons créé quatre autres jeux de données en découpant en deux les deux premiers. On obtient ainsi les jeux *Pattern A*, *Pattern B*, *Product A* et *Product B*.

En ce qui concerne notre méthodologie, les cinq modèles effectuent leur recommandation sur tous les produits d'un jeu de données. Ils sont ensuite évalués par le **HR** (Hit Rank) et le **NDCG** (Normalized Discounted Cumulative Gain) sur des top 100 et 200. Le *HR* va mesurer la capacité du système à donner toutes les solutions pertinentes. Dans le cadre de la recommandation, cela correspond à indiquer si oui ou non le produit cible est bien présent dans la recommandation top N. Si l'on a recommandé le produit cible à chaque fois, alors on aura un score de **1**. Le *NDCG*, va mesurer la qualité de notre classement. Ainsi, plus notre produit cible sera en haut du classement, plus la valeur sera proche de **1**.

Pour l'étude statistique nous avons utilisé les tests suivants (Demsar (2006)) :

1. **Test de Friedman** : le but est de réfuter l'hypothèse suivante : «Tous les algorithmes ont des performances équivalentes». Une p-value de $0,05$ sera utilisée ;
2. **Test de Nemenyi** : une fois le test de Friedman validé, on peut utiliser celui-ci pour montrer qu'un algorithme ou un groupe d'algorithmes est significativement plus performant qu'un autre. La confiance utilisée sera de $0,1$ à cause du faible nombre de jeux de données pour mener cette étude.

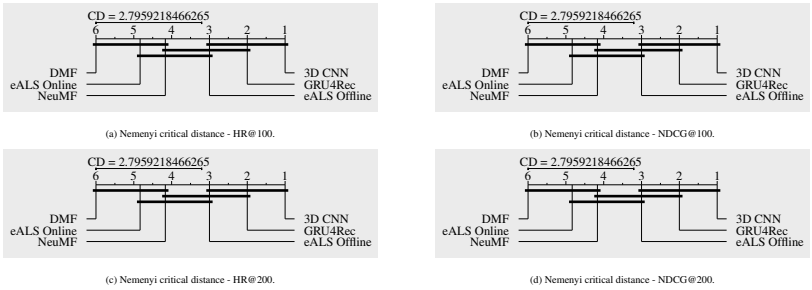
6. Une session égale un utilisateur dans cette approche.

La table 3 présente nos résultats, avec en gras le meilleur algorithme de la ligne. Chaque valeur représente la moyenne obtenue sur dix expériences, tandis que l'intervalle de confiance est calculé par l'écart type.

Jeu de données	Métrique	eALS Online	eALS Offline	NeuMF	DMF	GRU4Rec	3D CNN
Pattern Full	**HR@100**	0,5223 ± 0,0076	0,5756 ± 0,0241	0,528 ± 0,0018	0,4868 ± 0,0447	0,6674 ± 0,0009	**0,6861 ± 0,0055**
	NDCG@100	0,1747 ± 0,0035	0,1987 ± 0,0102	0,1816 ± 0,0057	0,1566 ± 0,0145	**0,2507 ± 0,0015**	0,2253 ± 0,009
	HR@200	0,6008 ± 0,0091	0,6668 ± 0,0025	0,6253 ± 0,0023	0,5863 ± 0,0413	0,7247 ± 0,0011	**0,743 ± 0,0075**
	NDCG@200	0,1871 ± 0,0034	0,2113 ± 0,0023	0,1988 ± 0,0053	0,169 ± 0,0119	**0,2584 ± 0,0003**	0,2332 ± 0,0012
Product Full	**HR@100**	0,5794 ± 0,0062	0,647 ± 0,0074	0,5767 ± 0,0054	0,5269 ± 0,0375	0,7108 ± 0,0007	**0,7195 ± 0,0062**
	NDCG@100	0,1954 ± 0,0031	0,2203 ± 0,0015	0,1928 ± 0,0076	0,178 ± 0,0137	**0,2634 ± 0,0001**	0,2328 ± 0,0009
	HR@200	0,6429 ± 0,0073	0,7169 ± 0,0074	0,6861 ± 0,0039	0,6409 ± 0,0513	0,757 ± 0,0008	**0,7821 ± 0,0078**
	NDCG@200	0,2019 ± 0,0023	0,2314 ± 0,0025	0,2104 ± 0,0071	0,1866 ± 0,0147	**0,2698 ± 0,0001**	0,2419 ± 0,0012
Pattern A	**HR@100**	0,5151 ± 0,0082	0,5462 ± 0,0293	0,5395 ± 0,0033	0,4946 ± 0,0463	0,6717 ± 0,0017	**0,7033 ± 0,0064**
	NDCG@100	0,1719 ± 0,0023	0,1911 ± 0,0117	0,1861 ± 0,0054	0,1579 ± 0,0149	**0,2569 ± 0,0007**	0,2321 ± 0,0011
	HR@200	0,5946 ± 0,0057	0,6394 ± 0,0133	0,6393 ± 0,0027	0,5916 ± 0,041	0,7106 ± 0,0012	**0,7689 ± 0,0095**
	NDCG@200	0,1855 ± 0,0044	0,2029 ± 0,0052	0,1987 ± 0,0052	0,1748 ± 0,0071	**0,2624 ± 0,0006**	0,2416 ± 0,0016
Pattern B	**HR@100**	0,5149 ± 0,0057	0,5549 ± 0,0272	0,5345 ± 0,0025	0,4847 ± 0,0326	0,675 ± 0,0024	**0,7355 ± 0,0055**
	NDCG@100	0,1748 ± 0,0029	0,1937 ± 0,0106	0,1821 ± 0,0052	0,1552 ± 0,0121	**0,2635 ± 0,0007**	0,2471 ± 0,0012
	HR@200	0,5861 ± 0,0073	0,636 ± 0,0203	0,6349 ± 0,0015	0,6193 ± 0,0338	0,7238 ± 0,0012	**0,7865 ± 0,0076**
	NDCG@200	0,1834 ± 0,0029	0,2098 ± 0,0093	0,1945 ± 0,0043	0,1735 ± 0,0113	**0,2703 ± 0,0005**	0,254 ± 0,0012
Product A	**HR@100**	0,5708 ± 0,007	0,6382 ± 0,0012	0,5881 ± 0,002	0,5484 ± 0,0355	0,7063 ± 0,0015	**0,7272 ± 0,0061**
	NDCG@100	0,1918 ± 0,0027	0,2198 ± 0,0007	0,1996 ± 0,0046	0,1799 ± 0,0074	**0,2669 ± 0,0011**	0,2348 ± 0,0009
	HR@200	0,6488 ± 0,0076	0,6981 ± 0,0194	0,6958 ± 0,0023	0,6332 ± 0,047	0,7436 ± 0,0014	**0,7911 ± 0,0133**
	NDCG@200	0,202 ± 0,0019	0,2293 ± 0,0074	0,2127 ± 0,0032	0,1863 ± 0,0099	**0,2722 ± 0,0011**	0,2441 ± 0,0021
Product B	**HR@100**	0,5598 ± 0,0085	0,6208 ± 0,0209	0,5827 ± 0,0028	0,5535 ± 0,0308	0,7143 ± 0,0014	**0,7248 ± 0,0114**
	NDCG@100	0,19 ± 0,0035	0,2089 ± 0,0047	0,1968 ± 0,004	0,1829 ± 0,0078	**0,2717 ± 0,0004**	0,2379 ± 0,0019
	HR@200	0,6333 ± 0,006	0,6968 ± 0,0047	0,6928 ± 0,0023	0,6542 ± 0,0472	0,7511 ± 0,0017	**0,7967 ± 0,0086**
	NDCG@200	0,2004 ± 0,0028	0,2222 ± 0,0046	0,2142 ± 0,0051	0,1901 ± 0,013	**0,2769 ± 0,0003**	0,2485 ± 0,0014

TAB. 3: Résultats des algorithmes sur les six jeux de données.

Une fois le test de *Friedman* validé pour le top 100 et 200, nous avons procédé au test de *Nemenyi* avec pour résultat les figures suivantes.



(a) Nemenyi critical distance - HR@100.

(b) Nemenyi critical distance - NDCG@100.

(c) Nemenyi critical distance - HR@200.

(d) Nemenyi critical distance - NDCG@200.

FIG. 2: Résultats des tests de Nemenyi pour le top 100 et 200

4 Conclusion

Dans ce papier, nous avons présenté cinq modèles appartenant à deux approches de la recommandation, *Memory-Based CF* et *Session-Based*. Notre étude sur des données réelles (*cf.* table 3), a montrée que la vision en session des données est significativement meilleure (*cf.* figures 2) que celle du collaborative filtering, pour une quantité d'information équivalente (*cf.* table 2 : eALS, NeuMF et GRU4Rec) et des données de même nature (implicites). La mauvaise performance des hybrides, peut s'expliquer en partie par la difficulté à les paramétrer (grand nombre de paramètres). En outre, il est à noter la bonne performance de notre implémentation du 3D CNN, qui pourrait être encore plus importante en utilisant plus d'informations annexes

(le prix du produit, sa consommation ou ses performances) qui seraient susceptibles d'influencer le choix des consommateurs.

Références

Bokde, D. K., S. Girase, et D. Mukhopadhyay (2015). Role of matrix factorization model in collaborative filtering algorithm: A survey. *CoRR 1503.07475*.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research 7*, 1–30.

Gong, S. (2010). A collaborative filtering recommendation algorithm based on user clustering and item clustering. *5*, 745–752.

He, X., L. Liao, H. Zhang, L. Nie, X. Hu, et T. Chua (2017). Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, Volume 3038912.3052569 of *WWW '17*, pp. 173–182. Springer.

He, X., H. Zhang, M. Kan, et T. Chua (2016). Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Volume 2911451.2911489 of *SIGIR '16*, pp. 549–558. ACM.

Hidasi, B., A. Karatzoglou, L. Baltrunas, et D. Tikk (2015). Session-based recommendations with recurrent neural networks. *CoRR 1511.06939*.

Hu, Y., Y. Koren, et C. Volinsky (2008). Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, Volume 1510528.1511352 of *ICDM '08*, pp. 263–272. IEEE Computer Society.

Quadrana, M., A. Karatzoglou, B. Hidasi, et P. Cremonesi (2017). Personalizing session-based recommendations with hierarchical recurrent neural networks. *CoRR 1706.04148*.

Tuan, T. X. et T. M. Phuong (2017). 3d convolutional networks for session-based recommendation with content features. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, Volume 3109859.3109900 of *RecSys '17*, pp. 138–146. ACM.

Xue, H., X. Dai, J. Zhang, S. Huang, et J. Chen (2017). Deep matrix factorization models for recommender systems. *3172077.3172336*, 3203–3209.

Zhang, S., L. Yao, et A. Sun (2017). Deep learning based recommender system: A survey and new perspectives. *CoRR 1707.07435*.

Summary

The recommendation engines have today an increasing influence on our consumption on the Internet. However, the data available to make a recommendation varies according to the users, the means of the industry and the type of products. In this paper, we focus on one hand, on the performances of matrix factorization as well as deep learning models and on the other hand, the recommendation on big data. A comparative study between several state-of-the-art models was carried out with an applicative purpose linked to a tyre data industrial.