

Ingénierie de la fouille et de la visualisation de données massives (RCP216)

Recherche et jointure par similarité. Systèmes de
recommandation

Michel Crucianu

(prenom.nom@cnam.fr)

<http://cedric.cnam.fr/vertigo/Cours/RCP216/>

Département Informatique
Conservatoire National des Arts & Métiers, Paris, France

16 octobre 2018

Plan du cours

2 Recherche et jointure par similarité

- Intérêt de la recherche et de la jointure par similarité
- *LSH* pour la distance cosinus
- *LSH* et la jointure par similarité
- Malédiction de la dimension

3 Systèmes de recommandation

- Le problème et les approches
- Recommandation par similarité de contenu
- Recommandation par filtrage collaboratif
- Filtrage collaboratif avec Spark

Recherche et jointure par similarité : intérêt

- Exemples d'utilisation de la similarité :
 - Recommandation pour le commerce en ligne : proposer à un client des produits achetés par d'autres clients ayant des profils d'achat similaires
 - Services de recommandation (films, musique) : proposer à un client des articles similaires à d'autres articles qu'il a appréciés
 - Services d'information sur le web : identifier les textes très similaires issus d'agences de presse ou d'autres sites d'informations afin d'en proposer un seul
- Recherche ou jointure ?
 - Recherche : trouver les données similaires à une donnée « requête »
 - ⇒ complexité $O(N)$ → complexité $O(\log N)$ ou $O(1)$
 - Jointure : trouver les paires de données très similaires (→ graphe → cliques ou composantes connexes)
 - ⇒ complexité $O(N^2)$ → complexité $O(N \log N)$ ou $O(N)$

Distance cosinus

- Souvent, la description des données est hybride :
 - 1 Ensembles d'articles, étiquettes (*tags*), mots, etc. L'ensemble associé à une donnée peut être très grand (par ex. client actif, de longue date) ou très petit (par ex. nouveau client)
 - 2 Variables nominales (à modalités) : genre (classique, policier, SF...), pays, navigateur web utilisé, mode de paiement...
 - 3 Variables quantitatives : nombre de pages, montant total déjà dépensé, fréquence des achats...
 - Quelle distance employer ?
 - Distance euclidienne : peu adaptée à (1)
 - Distance issue de l'indice de Jaccard : peu adaptée à (3)
 - Distance cosinus : peut cumuler les contributions des différents types de descriptions (les ensembles seront représentés par leurs vecteurs caractéristiques)
 - Cumul de plusieurs types de descriptions → choisir leurs pondérations relatives!
- $$\mathcal{D} = \mathbb{R}^m, d_{\cos}(\mathbf{x}, \mathbf{y}) = \arccos \frac{\mathbf{x}^T \cdot \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|}$$

LSH pour la distance cosinus

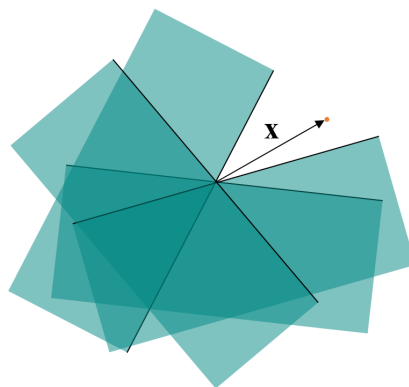
- Fonctions élémentaires $h \in \mathcal{H}_{\cos}$, $h : \mathbb{R}^m \rightarrow \{0, 1\}$,

$$h_{\mathbf{v}}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x}^T \cdot \mathbf{v} \geq 0 \\ 0 & \mathbf{x}^T \cdot \mathbf{v} < 0 \end{cases} \quad (1)$$

avec $\mathbf{v} \in \mathbb{R}^m$ tiré suivant la loi uniforme sur l'hypersphère unité

- \mathcal{H}_{\cos} est un ensemble de fonctions de hachage $(r_1, r_2, 1 - \frac{r_1}{180}, 1 - \frac{r_2}{180})$ -sensibles
Justification : $h_{\mathbf{v}}(\mathbf{x}) \neq h_{\mathbf{v}}(\mathbf{y})$ si l'hyperplan dont le vecteur normal est \mathbf{v} passe entre \mathbf{x} et \mathbf{y} ; cela arrive avec une probabilité $\frac{d_{\cos}(\mathbf{x}, \mathbf{y})}{180} \Rightarrow$ probabilité de collision est $1 - \frac{d_{\cos}(\mathbf{x}, \mathbf{y})}{180}$

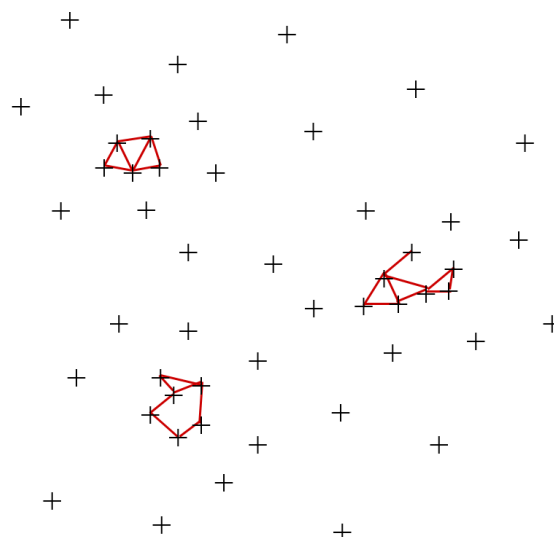
- Exemple dans \mathbb{R}^2 , avec 4 fonctions élémentaires :



(Auto-)jointure par similarité

- Rappel : auto-jointure avec seuil de distance θ , ensemble de N données \mathcal{D} :

$$K_{\theta} = \{(\mathbf{x}, \mathbf{y}) | \mathbf{x}, \mathbf{y} \in \mathcal{D}, d(\mathbf{x}, \mathbf{y}) \leq \theta\} \quad (2)$$

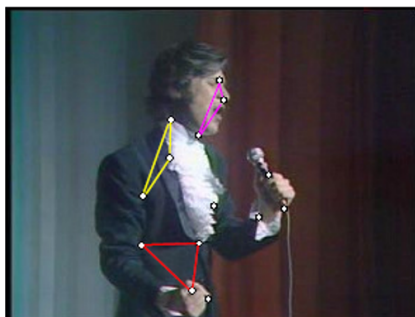


Auto-jointure par similarité avec LSH

- 1 Choix de fonctions de hachage suivant la distance utilisée, réglage de l'« amplification » en fonction du seuil de distance θ , ainsi que du coût de calcul et de stockage qui augmentent avec n (nombre de fonctions par table de hachage) et avec t (nombre de tables) $\rightarrow h_{\text{OR,AND}}$
 - 2 Application de la fonction de hachage à toutes les données ; pour chaque valeur de $h_{\text{OR,AND}}$, les données pour lesquelles le *hash* a cette valeur forment un *bucket*
 - 3 Pour chaque *bucket* non vide, calcul des distances $d(x, y)$ uniquement entre données du *bucket*
- ⇒ Évite de calculer la distance entre des données qui ne sont pas suffisamment similaires
- ⇒ Complexité : $O(N)$ pour étape (2) ; pour étape (3), borne inférieure qui dépend de la taille du résultat

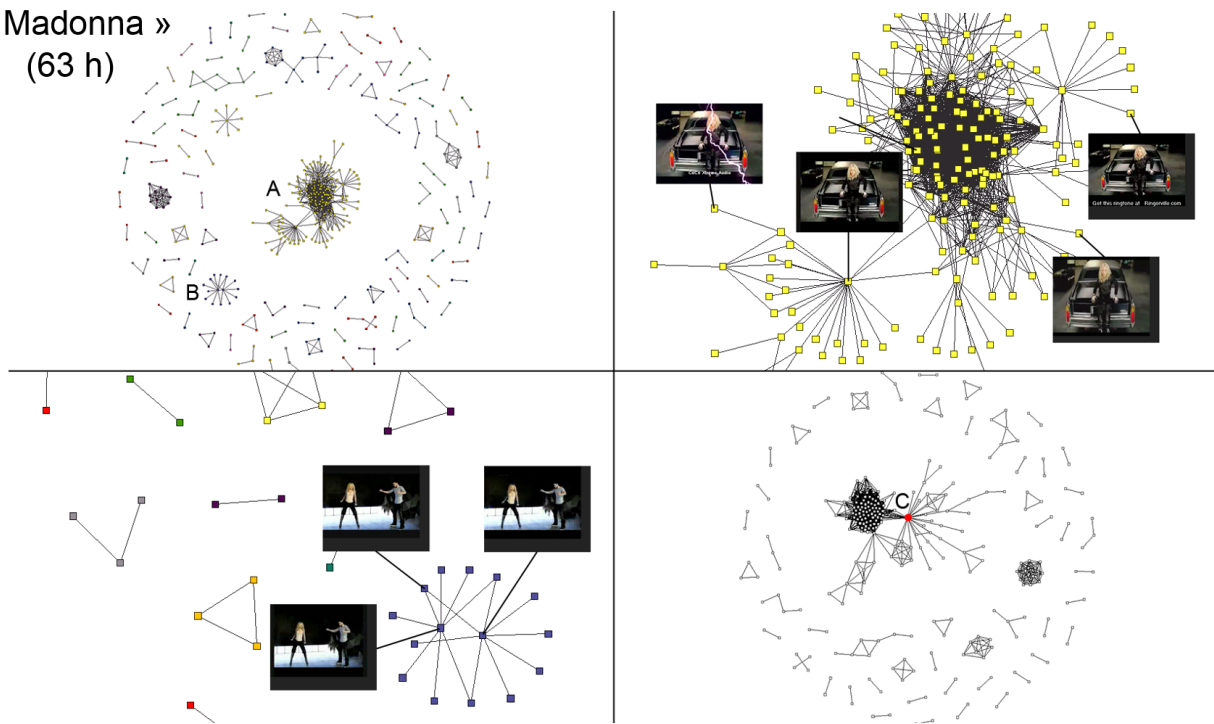
Exemple d'auto-jointure par similarité avec LSH

- Objectif : trouver des séquences vidéo **dupliquées avec transformations**
- Données : trames des vidéos d'une grande base
- Description de chaque trame : extraction et description de « points d'intérêt »
- Similarité : configuration géométrique de triplets de points d'intérêt
- LSH \leftarrow indice de Jaccard appliqué aux ensemble des triplets de points + prise en compte de la géométrie



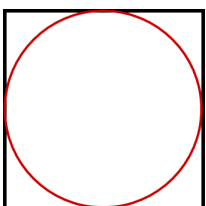
Exemple d'auto-jointure par similarité avec LSH (2)

« Madonna »
(63 h)



La « malédiction de la dimension » (*curse of dimensionality*)

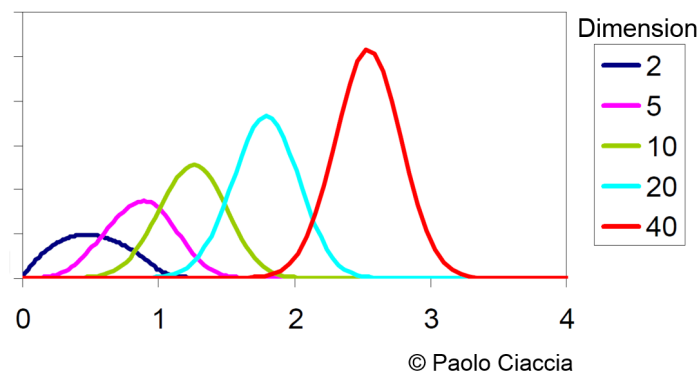
- A nombre de données fixé, la densité diminue exponentiellement avec la dimension
⇒ problèmes pour l'estimation de densités, tests statistiques
- Les données uniformément distribuées dans des volumes en dimension d sont proches des hypersurfaces externes (de dimension $d - 1$)



Dimension	Vol. sphère / vol. cube englobant
1	1
2	0,78732
4	0,329707
6	0,141367
8	0,0196735
10	0,00399038

La « malédiction de la dimension » (2)

- La variance de la distribution des distances diminue avec l'augmentation de la dimension (« concentration des mesures »)
 - ⇒ problèmes pour l'exploitation des distances : kNN, classification
 - ⇒ l'intérêt d'une recherche avec index diminue par rapport à une recherche exhaustive



Plan du cours

- 2 Recherche et jointure par similarité
 - Intérêt de la recherche et de la jointure par similarité
 - *LSH* pour la distance cosinus
 - *LSH* et la jointure par similarité
 - Malédiction de la dimension
- 3 Systèmes de recommandation
 - Le problème et les approches
 - Recommandation par similarité de contenu
 - Recommandation par filtrage collaboratif
 - Filtrage collaboratif avec Spark

Systèmes de recommandation

■ Définition du problème :

- Deux types d'entités : « utilisateurs » (*users*) et « articles » (*items*)
- Chaque utilisateur peut choisir et/ou noter plusieurs articles
- Données représentées sous forme matricielle (« utilités ») : 1 ligne par utilisateur, 1 colonne par article
 - Données explicites (par ex. achats, notes) : matrice **très** creuse
 - Données implicites (par ex. pages visualisées et durées de visualisation) : matrice mieux remplie

	A1	A2	A3	A4	A5	A6	A7	...
U1	9						2	
U2	2		8				7	
U3					4			
...								

- Objectif : prédire les valeurs manquantes de la matrice ; lorsqu'il s'agit de notes, ce sont en général les **valeurs élevées** qui intéressent

Systèmes de recommandation : approches

- 1 Similarité de contenu (*content-based filtering*) : proposer à l'utilisateur des articles qui correspondent à son **profil**
 - Accès nécessaire à des **descriptions** des articles
 - Construction de profils d'utilisateurs à partir de caractéristiques des articles qu'ils ont déjà choisis (ou notés avec une note élevée) et d'informations *a priori*
 - Difficultés à traiter les nouveaux utilisateurs, à extrapoler d'un domaine à un autre (par ex. de la musique aux livres)
- 2 Filtrage collaboratif : proposer à l'utilisateur des articles choisis (ou bien notés) par les utilisateurs similaires
 - Aucune connaissance **intrinsèque** des articles n'est nécessaire
 - Similarités entre utilisateurs à partir des articles choisis, similarités entre articles à partir des utilisateurs qui les ont choisis
 - Difficultés à traiter les nouveaux utilisateurs et les nouveaux articles
- 3 Hybride : combinaison de plusieurs méthodes, dont les deux précédentes

Synthèse récente : [2] (voir aussi [1] pour une synthèse sur les systèmes de recommandation d'articles de recherche)

Description des articles et profils des utilisateurs

■ Articles :

- Description initiale : ensembles (par ex. acteurs, réalisateur), variables nominales (par ex. genre), variables quantitatives (par ex. budget réalisation)
- Pondération des variables : par degré de présence et potentiel discriminant (par ex. $tf*idf$ pour éléments d'ensembles), par potentiel explicatif (par ex. homogénéité de la note donnée par des utilisateurs similaires)
- Réduction de dimension : résumer les variables initiales par un plus petit nombre de variables (révéler des « facteurs »)

■ Utilisateurs :

- Construction d'un profil agrégé de même nature (avec les mêmes variables) que les descriptions des articles
- Pourquoi : facilite la comparaison avec des descriptions d'articles
- Comment : cumul des descriptions des articles précédemment choisis (et/ou notés) ; si article noté, pondération de sa description par la note
- Alternative : construction de modèle décisionnel plutôt que profil agrégé

Recommandation par similarité de contenu

1 Emploi direct de la similarité entre profil utilisateur et description article :

- 1 Faire des suggestions à un utilisateur : recherche par similarité, avec comme « requête » le profil de l'utilisateur
- 2 Campagne promotionnelle (*push*) : jointure par similarité {ensemble profils utilisateurs} ↔ {ensemble descriptions articles}

2 Mise en œuvre de modèles décisionnels :

- Un modèle par utilisateur, construit à partir des données
- Objectif : prédire les articles auxquels l'utilisateur devrait donner des notes élevées
- Permettent d'obtenir des fonctions de décision plus complexes
- Un modèle décisionnel exige un volume de données plus important *par utilisateur* qu'un simple profil agrégé

Filtrage collaboratif

- Les utilisateurs, comme les articles, sont décrits essentiellement (ou exclusivement) par le contenu de la matrice de données

	A1	A2	A3	A4	A5	A6	A7	...
U1	9						2	
U2	2		8				7	
U3					4			

- Chaque utilisateur est décrit par son « profil ligne »
- Chaque article est décrit par son « profil colonne »

Filtrage collaboratif : approches

1. Basés sur la similarité (*memory-based*) :

- La matrice de notes est en général « normalisée » :
 - Moyenne nulle par utilisateur (ligne) pour équilibrer les « niveaux d'exigence »
 - Sans modification par article (colonne) pour prendre en compte leur « qualité intrinsèque »
- Comparer les lignes (utilisateurs) ou les colonnes (articles) : distance cosinus, corrélation linéaire

A. *User-based* :

- 1 Trouver les utilisateurs les plus « représentatifs » (par ex. les k les plus similaires) à l'utilisateur cible u
 - 2 Agréger leurs choix pour faire des propositions à u (par ex. les articles les plus choisis ou les mieux notés par ces k « voisins »)
- Difficulté : plus de diversité parmi les utilisateurs : x apprécie classique et jazz, y apprécie classique mais pas jazz

Filtrage collaboratif : approches (2)

1. Basés sur la similarité (*memory-based*) - suite :

B. *Item-based* :

- 1 Examiner les articles a_i choisis (ou bien notés) par l'utilisateur u
 - 2 Proposer à u les articles les plus similaires aux a_i (c'est à dire choisis ou bien notés ensemble par d'autres utilisateurs)
- Difficultés : peu d'articles notés par un même utilisateur → difficultés de discrimination ; dimension plus élevée des « profils » articles (car en général beaucoup plus d'utilisateurs que d'articles) → malédiction de la dimension

2. Basés sur un modèle (*model-based*) : développement d'un modèle (catégorisation utilisateurs et articles, facteurs latents, etc.) à partir de la matrice de données et prise de décision sur la base de ce modèle

3. Hybrides : combinaisons des deux précédentes

Filtrage collaboratif basé sur la factorisation matricielle

- Famille de méthodes dans le cadre de l'approche basée sur un modèle
- Principe : chercher des facteurs latents, en nombre relativement faible ($\sim 10^2$), qui expliquent les notes observées (le contenu de la matrice de données)
 - Articles et utilisateurs décrits par des vecteurs de même dimension, donnée par le nombre de facteurs latents considérés
 - Chaque article est décrit par des valeurs prises par les facteurs latents
 - Chaque utilisateur est décrit par les contributions de ces facteurs latents à la note qu'il donnerait à un article

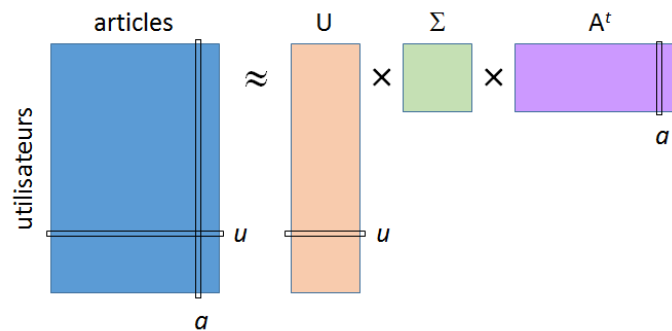
Filtrage collaboratif basé sur la factorisation matricielle (2)

- Une des premières méthodes explorées : décomposition en valeurs singulières (SVD) de la matrice utilisateurs-articles \mathbf{X} avec approximation de rang réduit

$$\mathbf{X} \approx \mathbf{U} \cdot \Sigma \cdot \mathbf{A}^t$$

Σ est diagonale et indique les poids des facteurs

- Chaque colonne de \mathbf{A}^t est la représentation « réduite » d'un article
- Chaque ligne de \mathbf{U} est la représentation « réduite » d'un utilisateur



- Difficulté majeure : la matrice utilisateurs-articles (\mathbf{X}) est très creuse, les valeurs absentes sont **manquantes** et non assimilables à 0

Filtrage collaboratif basé sur la factorisation matricielle (3)

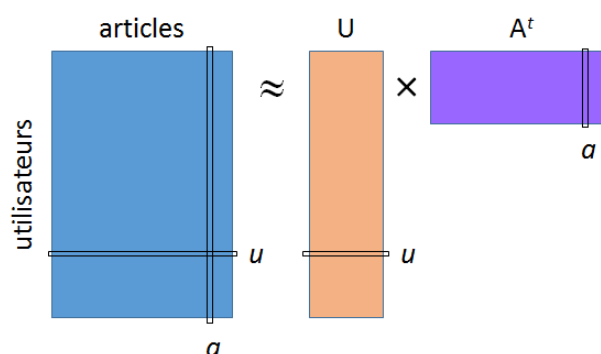
- Factorisation régularisée (voir par ex. [4], [3]) : approximation de rang réduit tenant compte seulement des valeurs **présentes** dans la matrice \mathbf{X} , avec régularisation

$$\min_{\mathbf{u}_i, \mathbf{a}_j} \sum_{\text{Present}(i,j)} (x_{ij} - \mathbf{u}_i^T \cdot \mathbf{a}_j)^2 + \lambda (\sum_i \|\mathbf{u}_i\|^2 + \sum_j \|\mathbf{a}_j\|^2) \quad (3)$$

\mathbf{u}_i est la représentation « réduite » d'un utilisateur

\mathbf{a}_j est la représentation « réduite » d'un article

λ contrôle la régularisation



Filtrage collaboratif basé sur la factorisation matricielle (4)

- Minimisation par algorithmes itératifs : moindres carrés alternés (dans Spark, par ex.) ou descente de gradient stochastique
- Après avoir trouvé les \mathbf{u}_i et \mathbf{a}_j pour tous les utilisateurs et respectivement articles, la prédiction de la note de k à l (non connue) est $x_{kl} = \mathbf{u}_k^T \cdot \mathbf{a}_l$
- La factorisation régularisée (3) et son algorithme de résolution itérative permettent d'intégrer dans le modèle d'autres aspects (voir par ex. [3]) :
 - Modélisation d'un biais par utilisateur (plutôt que la solution simple de « normalisation » mentionnée plus haut) et d'un biais par article :

$$\min_{\mathbf{u}_i, \mathbf{a}_j, b_{\mathbf{u}}, b_{\mathbf{a}}} \sum_{\text{Present}(i,j)} (x_{ij} - \mu - b_{\mathbf{u}_i} - b_{\mathbf{a}_j} - \mathbf{u}_i^T \cdot \mathbf{a}_j)^2 + \lambda \left[\sum_i (\|\mathbf{u}_i\|^2 + b_{\mathbf{u}_i}^2) + \sum_j (\|\mathbf{a}_j\|^2 + b_{\mathbf{a}_j}^2) \right]$$
 μ : moyenne globale, $b_{\mathbf{u}_i}$: biais pour utilisateur i , $b_{\mathbf{a}_j}$: biais pour article j
 - Modélisation de niveaux de confiance :






$$\min_{\mathbf{u}_i, \mathbf{a}_j, b_{\mathbf{u}}, b_{\mathbf{a}}} \sum_{\text{Present}(i,j)} c_{ij} (x_{ij} - \mu - b_{\mathbf{u}_i} - b_{\mathbf{a}_j} - \mathbf{u}_i^T \cdot \mathbf{a}_j)^2 + \lambda \left[\sum_i (\|\mathbf{u}_i\|^2 + b_{\mathbf{u}_i}^2) + \sum_j (\|\mathbf{a}_j\|^2 + b_{\mathbf{a}_j}^2) \right]$$
 c_{ij} : confiance dans la note donnée par l'utilisateur i à l'article j (ou plus généralement une **pondération** de cette note)

Le filtrage collaboratif dans Spark

- Méthode implémentée : factorisation régularisée (3) modifiée
 - Suivant [5], modification de la régularisation :

$$\min_{\mathbf{u}_i, \mathbf{a}_j} \sum_{\text{Present}(i,j)} (x_{ij} - \mathbf{u}_i^T \cdot \mathbf{a}_j)^2 + \lambda (\sum_i n_i \|\mathbf{u}_i\|^2 + \sum_j n_j \|\mathbf{a}_j\|^2)$$
 n_i : nombre de notes données par l'utilisateur i
 n_j : nombre de notes reçues par l'article j
 - Minimisation de (3) modifiée par *Alternating Least Squares* (ALS) : itérations qui alternent deux phases
 - 1 \mathbf{u}_i fixés, on obtient les \mathbf{a}_j comme solution d'un système linéaire
 - 2 \mathbf{a}_j fixés, on obtient les \mathbf{u}_i comme solution d'un système linéaire
- Exemple de recommandation de films avec MLlib : <http://ampcamp.berkeley.edu/big-data-mini-course/movie-recommendation-with-mllib.html>

Références I

-  J. Beel, B. Gipp, S. Langer, and C. Breiting.
Research-paper recommender systems : a literature survey.
International Journal on Digital Libraries, pages 1–34, 2015.
-  J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez.
Recommender systems survey.
Know.-Based Syst., 46 :109–132, July 2013.
-  Y. Koren, R. Bell, and C. Volinsky.
Matrix factorization techniques for recommender systems.
Computer, 42(8) :30–37, Aug. 2009.
-  A. Paterek.
Improving regularized singular value decomposition for collaborative filtering.
In *Proc. KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 39–42, 2007.
-  Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan.
Large-scale parallel collaborative filtering for the netflix prize.
In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management*, AAIM '08, pages 337–348, Berlin, Heidelberg, 2008. Springer-Verlag.