

MOTEURS DE RECOMMANDATION (Recommender systems)

INTRODUCTION

Objectif : proposer aux utilisateurs des choix **pertinents**.

Applications nombreuses. En particulier : produits culturels (Amazon, Netflix), publicité (Criteo, Facebook), réseaux sociaux (Facebook, LinkedIn, Pinterest, YouTube), recherche Web (Google, Bing), sites de rencontres (Meetic)...

Principe intuitif : utilisation des caractéristiques de l'utilisateur et/ou des produits pour proposer des choix parmi un catalogue de choix possibles.

Cours plus poussé de M. Habib.

EXEMPLE

Quelle méthode utiliser pour proposer à des étudiants de M2 informatique des cours qui les intéresseraient?

"Les étudiants qui ont pris ce cours ont pris ces cours..."

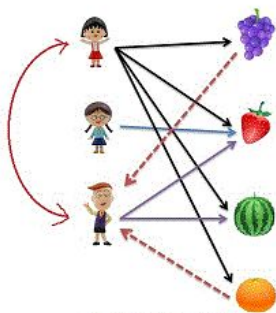
Le cours de Fouille de Données a des **mots-clés similaires** à ceux du cours de moteurs de recherche : si on aime l'un, on devrait aimer l'autre.

L'utilisateur **ressemble** à un utilisateur qui a aimé le cours de fouilles de données. Il devrait l'aimer aussi.

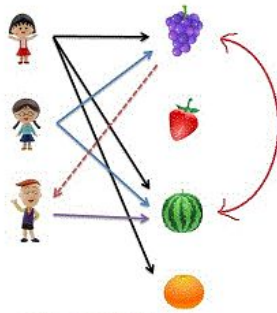
L'utilisateur **aime les mêmes cours** qu'un utilisateur qui a aimé le cours de fouilles de données. Il devrait l'aimer aussi.

Différents systèmes, tous pertinents mais pas forcément dans les mêmes contextes !

DEUX MODÈLES PRINCIPAUX



User-based filtering



Item-based filtering

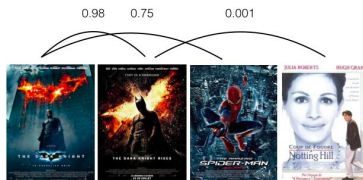
Source : *paxcel.net*

User-based : basé sur les utilisateurs

Content- ou Item-based : basé sur le contenu

- 1 Algorithmes basés sur le contenu
- 2 Algorithmes basés sur l'utilisateur
- 3 Autres méthodes

PRINCIPE



Camille	4	5	?	4
Mehdi	1	?	5	?
Julien	5	?	?	2

On modélise:

D'une part : le **profil-utilisateur**, i.e. les contenus qu'il a aimés.

D'autre part, les nouveaux contenus.

Objectif : donner une note aux nouveaux contenus en fonction des anciens.

MODÉLISATION DES CONTENUS

On cherche à repérer ce qui **caractérise** un contenu parmi les autres.

Souvent, on va donc utiliser un modèle **TF-IDF** qui met en avant l'originalité du contenu par rapport à un ensemble de contenus.

Une autre manière de faire est d'extraire des **mots-clés** du contenu (mais en général, on s'aide aussi d'un TF-IDF).

EXEMPLE

Un étudiant aime le cours de fouille de données (ce qui est normal). Ce cours, **comparé aux autres**, a de fortes valeurs TF-IDF pour les mots : données, fouille, intelligence, apprentissage, statistiques.

En revanche, il a des faibles valeurs TF-IDF pour les mots : algorithme (qui apparaît dans beaucoup de cours), systèmes, architecture, etc.

On mesure la **similarité** entre le TF-IDF du cours "Fouille de données" avec celui de chacun des autres cours.

On propose à l'étudiant des cours similaires, comme "Moteurs de recherche", qui a un **profil** similaire.

CALCUL DE LA SIMILARITÉ

	algorithmes	systemes	statistiques	donnees
FDD	0.1	0.05	3.5	3
MDR	0.1	1.2	2	2
SA	0.05	4	0	1

(Données non réelles)

On peut mesurer une **distance** entre chaque couple de cours et l'ordonner par ordre croissant.

On peut calculer d'autres **similarités**, que nous allons voir maintenant.

SIMILARITE “COSINUS”

Si x et y sont deux vecteurs TF-IDF, la **similarité cosinus** entre eux est :

$$S(x, y) = \frac{\langle x, y \rangle}{||x|| \times ||y||} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2 \sum_{i=1}^n y_i^2}}$$

SIMILARITE “COSINUS”

Si x et y sont deux vecteurs TF-IDF, la **similarité cosinus** entre eux est :

$$S(x, y) = \frac{\langle x, y \rangle}{||x|| \times ||y||} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$

- $\langle x, y \rangle$ est le produit scalaire entre x et y . Par exemple:

$$\langle \text{FDD}, \text{MDR} \rangle = 0.1 \times 0.1 + 0.05 \times 1.2 + 3.5 \times 2 + 3 \times 2 = 13.07$$

- $||x||$ est la norme de x . C'est aussi la racine du produit scalaire de x avec lui-même. Elle représente la “taille” de x .
- Le numérateur est grand si x et y ont de grandes valeurs aux mêmes endroits.
- Pour éviter un effet taille, on divise par le produit des normes.

SIMILARITÉ COSINUS - SUITE

En réalité, on voudrait calculer **l'angle** entre les deux documents. Mais en pratique, il est plus simple de calculer le cosinus.

Plus l'angle est faible, plus le cosinus est grand : $\cos(0) = 1$ pour deux vecteur identiques, $\cos(90) = 0$ pour deux vecteurs indépendants.

Exemple : $x = (0, 1, 2, 0)$, $y = (0, 1, 2, 0)$, $z = (1, 0, 0, 1)$:
 $\cos(x, y) = 1$ et $\cos(x, z) = 0$.

En pratique, cette similarité se trouvera toujours entre 0 et 1 car on ne considère que des termes positifs.

Donc il sera facile de savoir si deux documents se ressemblent:

0 : aucune ressemblance

1 : documents identiques

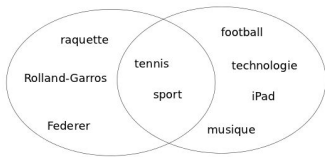
0.5: ressemblance moyenne

SIMILARITÉ DE DICE-SORENSEN

Si A et B sont deux ensembles de mots-clés, la **similarité de Dice** entre eux est :

$$S(A, B) = \frac{2|A \cap B|}{|A| + |B|} = 2 \times \frac{\text{Nombre de mots en commun}}{\text{Nombre total de mots}}$$

Objet décrit par des mots-clé



Mots-clé préférés de l'utilisateur

Exemple: ici,

$$S(A, B) = \frac{2 \times 2}{5 + 6}$$

Source : podcastscience.fm

Modélisation du profil-utilisateur : méthode 1

Supposons qu'un utilisateur ait consulté, ou, mieux, "liké" quatre documents A, B, C et D. On veut mesurer son **profil**.

- Le plus simple est de regarder la **moyenne** de ce qu'il aime:

$$P = \frac{TFIDF(A) + TFIDF(B) + TFIDF(C) + TFIDF(D)}{4}$$

On obtient alors un vecteur de ses **goûts moyens**.

- Ce vecteur P est celui qui est comparé aux nouveaux contenus. Les contenus les plus proches de P sont proposés à l'utilisateur.
- Inconvénient** : s'il aime des choses très différentes, faire leur moyenne n'a aucun sens !

Modélisation du profil-utilisateur : méthode 2

- On commence par **clusteriser** les contenus passés. Par exemple, on trouve que A et B forment un cluster et que C et D en forment un autre.
- Pour chaque nouveau contenu, on regarde la **similarité avec chaque cluster**, en utilisant la méthode que l'on veut : distance minimale, maximale, moyenne, Ward ou encore similarité cosinus avec la moyenne de chaque cluster.
- Ainsi, on est sûr de ne moyenner que des **contenus comparables**.

Algorithm 1 Recommandation basée sur le contenu

- 1: **Initialisation** : X : matrice composée de vecteurs TF-IDF ou mots-clés des contenus passés aimés de l'utilisateur.
 - 2: **for** Chaque visite sur le site **do**
 - 3: Calculer le profil-utilisateur avec la méthode 1 ou 2;
 - 4: Trouver le produit le plus similaire au profil-utilisateur;
 - 5: **if** L'utilisateur aime ce contenu **then**
 - 6: Il est ajouté à X ;
 - 7: **end if**
 - 8: **end for**
-

Problème dit de "cold-start"

On parle de **cold-start** lorsque l'on ne connaît pas du tout l'utilisateur et qu'il n'a aimé aucun contenu.

On est alors obligé de lui proposer des contenus choisis **au hasard**.

Conclusion sur la recommandation par contenu

Avantages :

- L'algorithme fait des propositions personnalisées.
- Il ne requiert pas la participation de milliers d'utilisateurs.

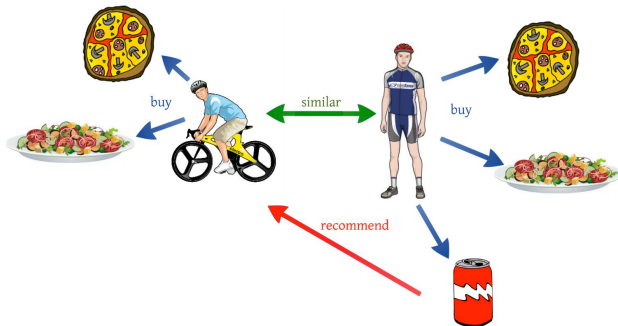
Inconvénients :

- L'algorithme ne se base que sur le profil-utilisateur, il ne prend pas en compte les autres utilisateurs.
- Il ne répond pas au problème du cold-start.
- Certains types de contenus (subjectifs) sont difficiles à comparer.

- 1 Algorithmes basés sur le contenu
- 2 Algorithmes basés sur l'utilisateur
- 3 Autres méthodes

PRINCIPE

On part du principe que les utilisateurs ayant aimé les mêmes choses dans le passé aimeront les mêmes choses dans le futur.



Source : blog.comsysto.com

On parle de **collaborative filtering**.

PROBLÈME DE DÉPART



Camille	4	5	?	4
Mehdi	1	?	5	?
Julien	5	?	?	2

On cherche à **remplir les cases inconnues**.

SIMILARITÉ DES UTILISATEURS

Si x et y sont deux utilisateurs représentés par leur vecteur de goûts, alors la similarité entre eux est mesurée par le **coefficient de corrélation** :

$$\rho(x, y) = \frac{\sigma_{xy}}{\sigma_x \sigma_y} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}$$

σ_x et σ_y sont les **écarts-type** de x et y . Ils mesurent la variance, la variabilité de chacun des utilisateurs.

σ_{xy} est la **covariance** entre x et y . Elle décrit le comportement d'une des deux variables en fonction de l'autre. Par exemple, une covariance positive signifie que x et y varient dans le même sens. La **corrélation** ρ se trouve entre -1 (opposés) et 1 (identiques).

GESTION DES DONNÉES MANQUANTES

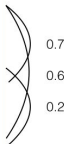
Comment mesurer la corrélation entre deux utilisateurs si certaines "cases" sont vides?

Solution 1: ne prendre en compte que les "lignes" complètes pour les deux utilisateurs.

Solution 2: remplacer les cases manquantes par la moyenne des notes des utilisateurs.



Camille	4	5	?	4
Mehdi	1	?	5	?
Julien	5	?	?	2



Source : podcastscience.fm.

D'autres solutions existent mais elles sont beaucoup plus compliquées !

PROPOSITION DE CONTENUS

On commence par **calculer la corrélation** entre l'utilisateur-cible et tous les autres.

Les valeurs de corrélation sont alors utilisées comme des poids pour calculer une **moyenne pondérée** de leurs notes (ratings) pour chaque nouveau contenu.

Cette moyenne pondérée est utilisée pour **prédire une note** utilisateur-cible/nouveau contenu.

EXAMPLE

	Etudiant 0	Etudiant 1	Etudiant 2
Systèmes avancés	0	1	0
Bases de données	1	1	1
Fouille de données	1	0	?
Moteurs de recherche	1	0	1
XML	1	0	0

L'étudiant 2 va-t-il aimer le cours de Fouilles de Données ?

```
E = np.array([[0,1,1,1],[1,1,0,0],[0,1,1,0]])
```

```
C = np.corrcoef(E)
```

```
print C
```

```
[[ 1.          -0.57735027  0.57735027]
 [-0.57735027  1.          0.          ]
 [ 0.57735027  0.          1.          ]]
```

```
rating = C[0,2]* 1 + C[1,2]* 0
```

```
print rating
```

```
0.57735026919
```

VARIANTE

	Etudiant 0	Etudiant 1	Etudiant 2
Systèmes avancés	1	3	0
Bases de données	5	4	4
Fouille de données	4	1	?
Moteurs de recherche	4	2	1
XML	5	1	1

L'étudiant 2 va-t-il aimer le cours de Fouilles de Données ?

```
E = np.array([[1,5,4,5],[3,4,2,1],[0,4,1,1]])
```

```
C = np.corrcoef(E)
```

```
print C
```

```
[[ 1.          -0.2045983    0.66082714]
 [-0.2045983    1.          0.59628479]
 [ 0.66082714   0.59628479   1.          ]]
```

```
rating = C[0,2]* 4 + C[1,2]* 1
```

```
print rating
```

```
3.23959334643175
```

PROBLEME DE COLD START

Ici, le problème de **cold-start** revient à avoir une ligne entière vide (cold-start item) ou une colonne entière vide (cold-start user).

Dans beaucoup de cas, on va donc demander aux utilisateurs de donner des ratings **explicitement**.



Source : bogost.com

PROBLÈMES DE CALCUL

Si, comme Netflix, on a des dizaines de millions d'utilisateurs, le calcul **en temps réel** devient trop lourd.

On va donc **précalculer off-line** les similarités.

On choisit les K utilisateurs **plus proches voisins** (K-nearest neighbors), c'est-à-dire les K utilisateurs ayant les plus fortes corrélations avec l'utilisateur-cible.

On ne calcule le rating qu'à partir de ces K utilisateurs.

Variante : on pré-sélectionne les utilisateurs ayant une corrélation supérieure à c avec l'utilisateur-cible.

Idée sous-jacente : les autres sont négligeables.

USER-CENTRIC VS ITEM-CENTRIC

L'approche qu'on vient de voir compare d'abord les utilisateurs. On l'appelle **user-centric**.

On peut inverser les calculs en calculant les similarités sur les contenus (**item-centric**) : deux contenus avec les mêmes ratings utilisateurs ont une similarité 1.

Si un utilisateur aime un contenu, on lui propose un contenu corrélé.

L'approche item-centric est moins gloutonne et permet le temps réel.

C'est l'approche inventée par **Amazon**.

Attention : cette approche est différente de la recommandation par contenu seulement qui n'utilise pas les notes des utilisateurs.

COLLABORATIVE FILTERING: CONCLUSION

Avantages:

Pas besoin de connaissance sur le contenu : on ne se base que sur les notes des utilisateurs.
Il est donc plus facile à implémenter.

Inconvénients:

Calculs lourds.
Problèmes de cold-start non résolus.

Sparsity (parcimonie, rareté) : sur un site comme Amazon, il y a des millions de produits. Les utilisateurs n'en noteront qu'une minuscule partie => beaucoup de valeurs manquantes.

Outline

- 1 Algorithmes basés sur le contenu
- 2 Algorithmes basés sur l'utilisateur
- 3 **Autres méthodes**

Approche basée sur le comportement passé de l'utilisateur (clics, recherches Google, likes...). On va simplement lui recommander des contenus en fonction de ce qu'il a déjà cherché.

Exemples : AdSense, Criteo

Recommandation hybride

La **recommandation hybride** se base sur les 3 approches précédentes. Elle combine donc les actions passées de l'utilisateur, les similarités entre contenus et le collaborative filtering.

Bien sûr, elle est plus performante que n'importe laquelle des autres seulement. Elle permet de résoudre les problèmes de cold-start et de sparsity.

En pratique, on peut par exemple pré-remplir la matrice de collaborative filtering avec un algorithme basé sur le contenu et l'historique de l'utilisateur.

C'est l'approche utilisée par Google, Amazon et Netflix.

Exemple : Moteur de recommandation type Youtube

- **Recommandation personnalisée** : en fonction de l'historique et de nos caractéristiques sociales (localisation, âge, etc.)
- **Recommandation basée sur le contenu** : les vidéos ont des tags similaires
- **Collaborative filtering** : les utilisateurs qui ont aimé la vidéo que vous regardez ont aussi aimé ces autres vidéos.

Attention : ce ne sont pas les mêmes choses ! Il ne faut pas les confondre.

Ces deux types d'algorithmes n'ont en commun que l'utilisation de **distances** entre objets. (Et parfois le clustering est utilisé dans le cadre de la recommandation)

- **Clustering** : algorithmes **non supervisés** dont le but est de grouper des objets dans des classes.
- **Recommandation** : algorithmes qui utilisent des informations données par les utilisateurs pour leur proposer d'autres choix.

Conclusion sur la recommandation

- En fonction du problème et surtout du type de données, on choisit un algorithme approprié.
- **Croyances...** Croyez-vous qu'il soit pertinent de proposer des contenus en fonction de ce qu'ont aimé les autres ? Si oui, le collaborative filtering a un sens. Sinon, il faut utiliser autre chose. Cela dépend également du type de contenu.
- **Algorithmes fructueux** : en 2009, Amazon annonçait que 30% de son chiffre d'affaire était obtenu via de la recommandation !
- **Algorithmes customisables** : le bon sens et les idées "non-mathématiques" en général vont jouer un rôle important.
- Beaucoup de projets basés sur la recommandation. Nous verrons quelle équipe fait les meilleures!