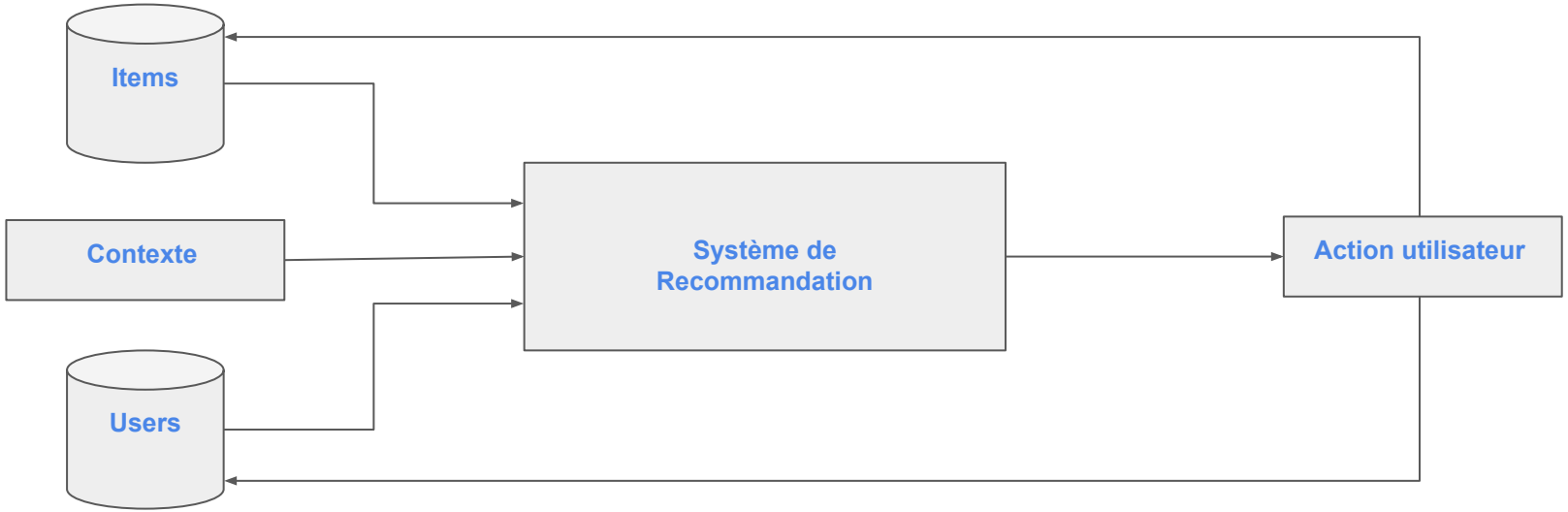
(toute petite)

Introduction aux systèmes de recommandation

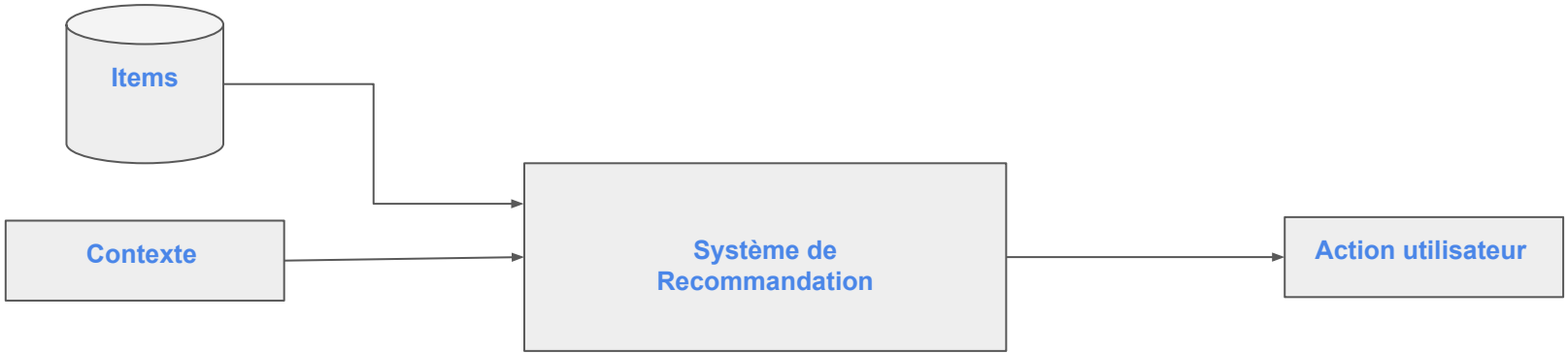
Rodolfo Ripado | Pizza Talk Radio France | Janvier 2016

Quelques exemples

Qu’est-ce qu’un Système de Recommandation ?



Le degré zéro de la reco : la similarité des items



Recommendation par similarité

- On définit un tableau d'éléments (items) et de leurs propriétés (features)
- On calcule les similarités entre les items
- On recommande les *N* items le plus proches de l'item actuel

Example

Items

	Rock	Jazz	RnB	Artiste_1	Artiste_2
song_1	1	0	0	1	0
song_2	0	1	0	1	0
song_3	1	0	0	0	1

Score de similarité

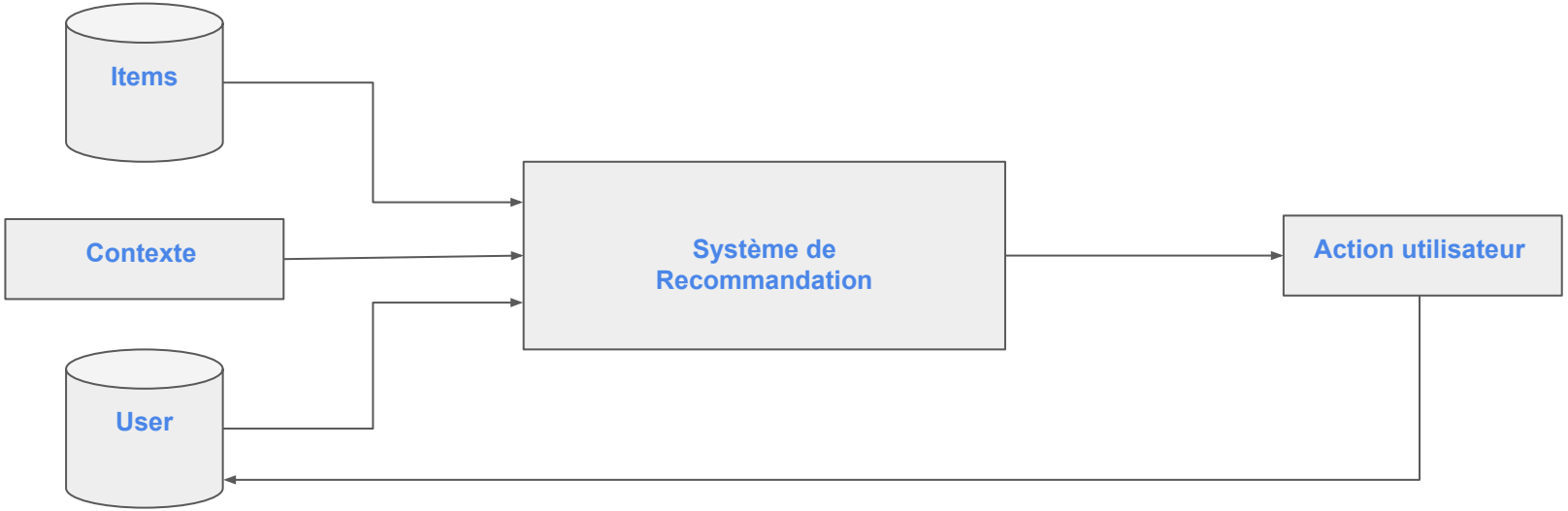
	song_1	song_2	song_3
song_1	1	3/5	3/5
song_2	3/5	1	1/5
song_3	3/5	3/5	1

Si l’utilisateur a écouté la chanson *song_2*, on lui propose *song_1*

Recommendation par similarité

- Implémentation très simple : zéro personnalisation, zéro apprentissage
- Scalabilité au top : aucune boucle de feedback, calculs totalement *offline*.
- Deux gros soucis :
  - Le *semantic-gap*
  - La *echo-chamber*

Content-based filtering



Content-based filtering

- On définit une matrice d’items et de leurs features
- On calcule une matrice d’utilisateurs, avec leurs préférences
- On calcule une matrice user-item : les similarités entre les users et les items
- On recommande des items les plus proches du profil utilisateur

Content-based filtering : les données de base

Items

	Rock	Jazz	RnB	Artiste_1	Artiste_2
song_1	1	0	0	1	0
song_2	0	1	0	1	0
song_3	0	0	1	0	1

User profiles

	Rock	Jazz	RnB	Artiste_1	Artiste_2
user_1	1	1	0	1	0
user_2	0	1	1	0	1
user_3	1	0	1	1	0

Mesure de similarité : ***cosine similarity***

Score de -1 à 1, proportionnel à l'angle formé par les deux vecteurs (l'item et le profil de l'utilisateur).

Content-based filtering : la matrice user-item

User -Item

	song_1	**song_2**	**song_3**
user_1	0.82	0.82	0
user_2	0	0.41	0.82
user_3	0.82	0.41	0.41

Si l'utilisateur *user_2* vient de jouer la *song_3*, on recommande la *song_2*.

En fonction de son comportement (like, skip, ...) on met à jour le profil (ou pas).

Construction du profil utilisateur

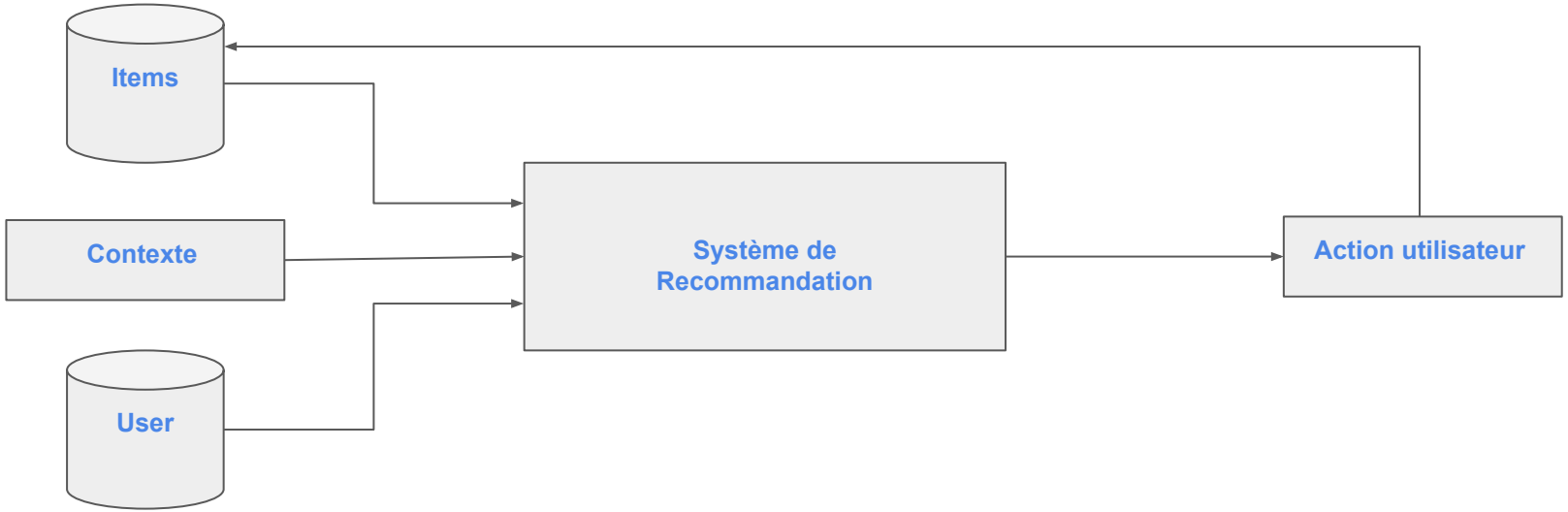
- Feedback explicite : lui demander ses préférences
- Feedback implicite simple : compter les “likes”, les “étoiles”, …
- Feedback implicite plus complexe : apprendre le profil qui mieux explique ses choix via des algorithmes de machine learning

(…)

Content-based filtering : les soucis

- Déjà abordés : *semantic-gap*, *echo-chamber*
- Scalabilité : tailles des matrices
- Le *cold start*
- La mécanique de mise à jour des profils
- Les goûts sont souvent contextuels (évolution temps, heure du jour, activité, geoloc, ...)

Collaborative filtering



Collaborative filtering : la matrice *user-item*

On commence par enregistrer les items likés / achetés par les utilisateurs :

	song_1	**song_2**	**song_3**
user_1	1	1	1
user_2	0	1	1
user_3	0	1	0

Collaborative filtering : la matrice *item-item*

On calcule ensuite la similarité entre les items basée uniquement sur leur popularité (similarité de *Jaccardi*):

similarité(song_1, song_2) = #(users like song_1 ET song_2) / #(users like song_1 OU song_2)

User / item

	song_1	**song_2**	**song_3**
user_1	1	1	1
user_2	0	1	1
user_3	0	1	0

Item / item

	song_1	**song_2**	**song_3**
song_1	1	1/3	1/2
song_2	1/3	1	1/2
song_3	1/2	1/2	1

Si un utilisateur *a liké la song_2* on recommande la *song_3*

Collaborative filtering : les difficultés

- Scalabilité : tailles des matrices
- Le *cold start*
- Les goûts contextuels (heure du jour, activité, geoloc, ...) et évolutifs
- La *longue traîne* : des chansons ou genres de niche ne sont jamais recommandés car très peu likés.

Plein d’autres approches

- Systèmes hybrides : alterner entre les deux systèmes, votes, ...
- Knowledge-based : ajout contraintes métier, …
- Profil contextualisé : démographique, activité, temps, …
- ...

Que retenir : 2 axes majeurs de la recommandation

Content-based filtering

basé sur la similarité des items et des profils utilisateur

Collaborative filtering

basé sur la popularité conjointe des items

Que retenir : les difficultés

L’echo-chamber et la sérendipité

Personne ne veut se faire recommander tout le temps la même chose

La longue traîne

Il est difficile de recommander des articles de niche, mais si important ...

Le semantic gap

La similarité est perceptuelle, culturelle, contextuelle, ...

La scalabilité de tout ça

L’espace de recherche grossit très vite

Les goûts ne sont pas figés

Les goûts évoluent et sont contextuels (pays, activité, geoloc, ...)

Pendant ce temps, dans le monde réel ...

- Les algos réels mélangent les deux types, et y ajoutent beaucoup d’information contextuelle.

- Il faut avancer très lentement, et évaluer en permanence car peu de marge pour l’erreur.

Qu’est-ce qu’un bon RS ?

- **Précision :** l’utilisateur va liker / acheter ce qu’on recommande
- **Diversité :** on a des recommandations d’items *pareils mais différents*
- **Persistence :** items qu’on veut faire acheter reviennent
- **Sérendipité :** la recommandation d’items étonnants, inattendus.
- **La confiance :** un user mécontent, ne revient plus et vous discrédite dans son réseau.