centrale_201606_sysrec

June 26, 2023

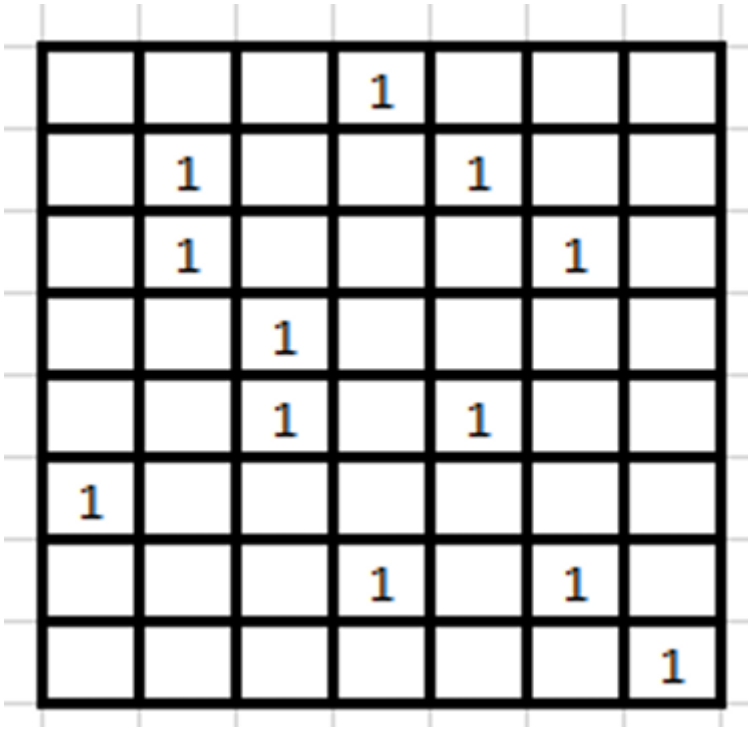
1 Recommandations sur le web

Quelques pistes sur le fonctionnement des moteurs de recommandations sur le web.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.1 Formalisation d'un système de recommandation



1.1.1 Tenseur

- Tenseur = matrice multidimensionnelle
  - Utilisateur
  - Produit
  - Temps
  - Localisation
  - Contexte
- Quelle valeur pour les paires non observées ?
- Chaque dimension multiplie les possibilités sans multiplier les données.
- Non-negative matrix factorization

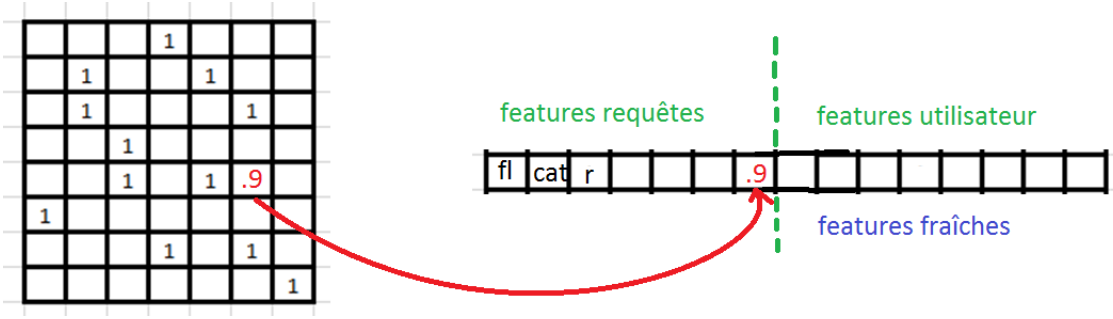
1.1.2 Système de recommandation, ranking

- Ranking similaire à un système de recommandation
  - Requête –> résultats conseillés
  - Requête –> requêtes associées (Related Searches)
- Recommandation pure
  - Le système n'utilise pas d'information sur le contenu
  - Uniquement basée sur l'évaluation implicite faite par l'utilisateur (le clic)

1.1.3 Recommandation en pratique

- Système de recommandation pure –> une feature
- Features extraites à partir du contenu, du contexte, de l'utilisateur
- Utilisation de classifieurs intermédiaires (annotation manuelle, catégorisation)

- Apprendre le feedback des utilisateurs, généraliser là où il n'est pas



1.2 Offline / Online

Online

- Le service qui délivre les réponses à l'utilisateur.
- Quelques millisecondes

Offline

- Tout les traitements qui produisent les données que le service online utilisent.
- Data workflows
- Rafraîchissements fréquents et périodiques

1.2.1 Offline – cycle long – beaucoup de map reduce - machine learning

- Utilisation de logs d'événements (achats, clic)
- Génération de candidats : requête –> liste de résultats, système de recommandation
- Extraction de features (requête, résultat, paire requête résultats)
- Machine learning, targets construites à partir des des clics

Construction d'une liste de recommandations à utiliser online $\sim O(100M)$.

1.2.2 Online – cycle très court

- Ranking online –> doit être rapide
- Personnalisation ?
- Compromis performance / pertinence

1.2.3 Systèmes de contrôle

- Black lists / while lists
- Possibilité de revenir à une version antérieure

1.2.4 Usure régulière

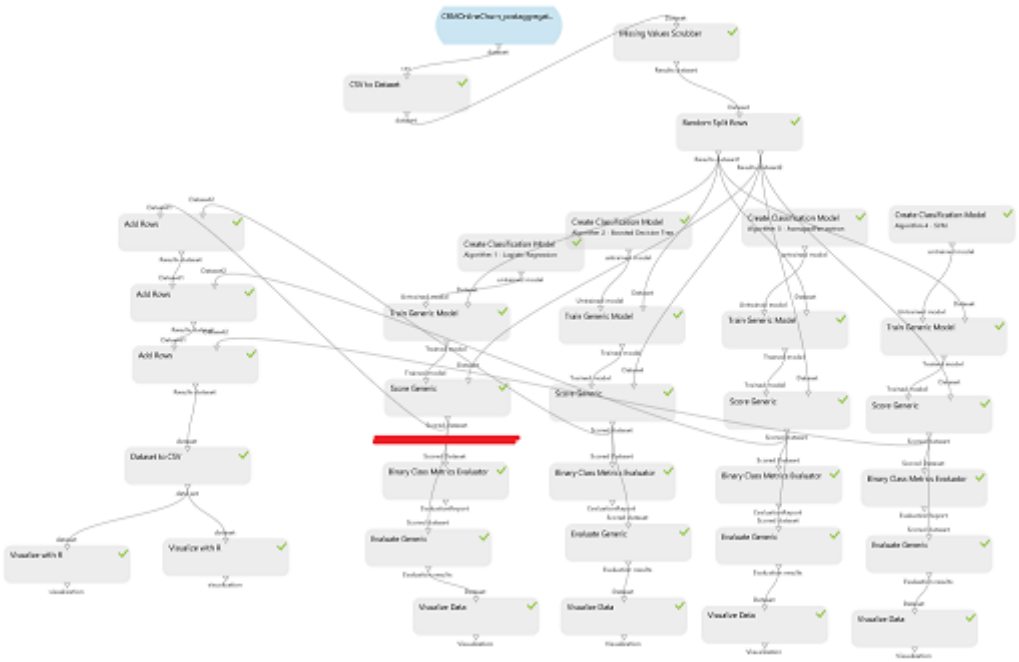
- Sans réapprentissage : dégradation des performances du ranker
- Croissance régulière des jeux de données
- Croissance des temps de traitements
- Nouvelles requêtes, livres, comment gérer ?

1.3 Kaggle VS machine learning en continu

- Kaggle = une compétition d'un jour
  - Ingéniérie << recherche
- Production = réapprentissage en continu
  - Ingéniérie > recherche

1.3.1 Workflow, pipeline ML

Always growing.



1.3.2 Questions difficiles

- Mesurer l'impact d'une modification ?
  - Au début du pipeline ?
  - A la fin du pipeline ?
- Corrélation entre
  - La performance offline du pipeline
  - Les préférences des utilisateurs (test A/B)
- Impact d'une modification sur les données d'apprentissages futures ?

1.3.3 Vécu

- Documentation manquante ou orale (black list, seuils…)
- On préfère ajouter un patchs que de modifier un code existant.
- Implémentation pas forcément conçue pour tout volume de données.
- Difficulté de changer de technos ou d'algorithmes
- Agilité contre robusté
- Tests unitaires sur des clusters : pas de solutions miracles

1.4 Quelques paris sur l'avenir

Métrique

- Rôle d'une suggestion dans le parcours utilisateur ?
- Clic sur une suggestion : est-elle drôle ou pertinente ?
- Appauvrissement des données dû aux suggestions

Personnalisation renforcée

- Un utilisateur voit différentes suggestions même s'il fait la même requête
- Prise en compte de l'heure, de la météo…

Apprentissage par renforcement

- Randomisation des algorithmes
- Utiliser l'utilisateur pour faire converger le système
- MWT