Projet 9 : My Content

Challenge : Réalisez une application de recommandation de contenu

https://github.com/blanchonnicolas/IA_Project9_Openclassrooms_IA_SystemeRecommandation

Agenda

01

Introduction

Contexte et Objectifs
Présentation du jeu de données
Préparation des données

02

Système de recommandation

Métriques et mesure de la performance
Item-Based
Collaborative Filtering - Model Based
Comparatif

03

Déploiement de l'outil

Stockage GIT
Architectures AS-IS et TO-BE

04

Démo

Démo
Conclusions

Back to AGENDA

Introduction

Contexte et Objectifs

Système de recommandation

Solution de recommandation d’articles et de livres à des particuliers, basée sur les interactions des utilisateurs avec les articles disponibles.

Application mobile de recommandation

Interface qui liste les identifiants des utilisateurs et affiche les résultats des 5 suggestions d’articles, suite à l’appel d’une API Serverless hébergée sous Azure Functions.

Objectifs de la mission

Construire une solution de recommandation d'articles pour les utilisateurs de l'application My Content

Encourager la lecture en recommandant 5 articles pertinents pour chaque utilisateur.

Articles (avec caractéristiques et embeddings).
Sessions des utilisateurs et les interactions avec les articles.

Développer et tester plusieurs systèmes de recommandations.
Réaliser une application simple qui retourne les 5 recommandations d'articles.
Stocker les scripts sur Github.
Synthétiser les réflexions pour l'architecture cible.

Présentation du jeu de données

Compréhension du jeu de données

My Content

2 fichiers

- articles_metadata.csv
  - 10.5Mo
- articles_embeddings.pickle
  - 347Mo

1 dossier “clicks”

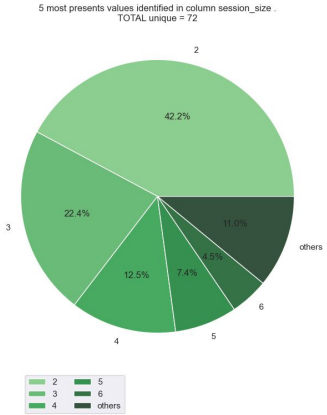
- 385 fichiers clicks.csv
  - 210Mo
  - 12 Colonnes
  - 322897 unique users
  - Object ⇒Integer and Datetimes

Après assemblage des 385 fichiers clicks, nous relions les metadata associés aux articles.

Les 385 fichiers clicks sont 2.988181e+06 relevés de sessions à dates très proches: 2017-10-01 à 2017-11-13.

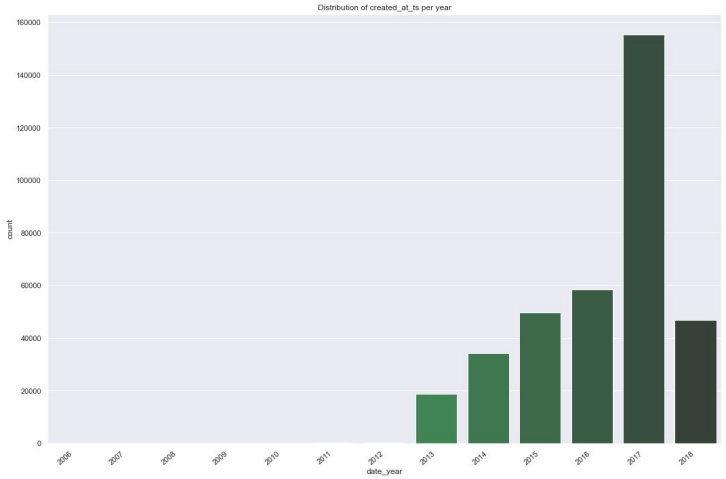
461 catégories d’articles sont recensées, au travers de 364047 articles pour lesquels le fichier Embedding fournit des informations sur leur contenu.

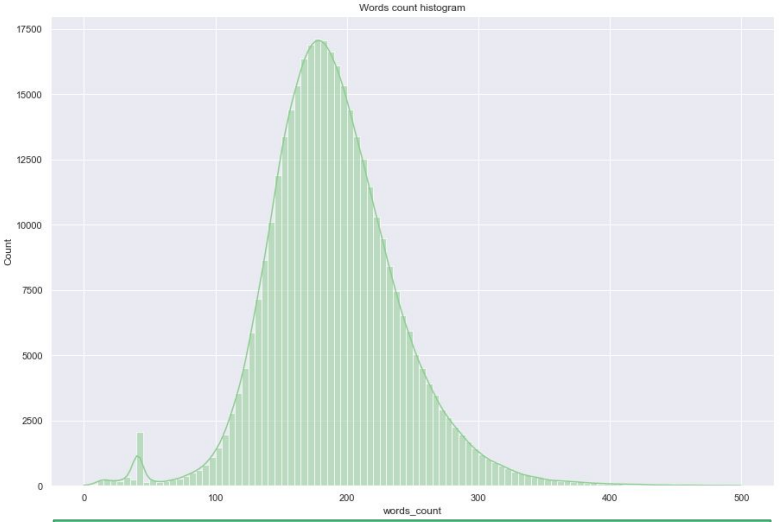
Chiffres clés du jeu de données



Les 72 tailles de sessions sont présentes sous formes de labels

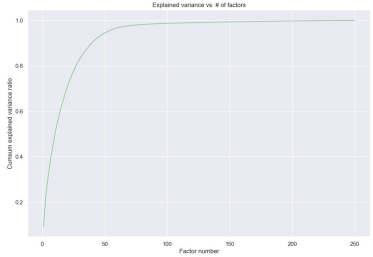
L'ancienneté des articles est présente, ainsi que les poids des 250 vecteurs représentant les informations sur le contenu (fichier Embedding).





La plupart des articles recensés contiennent entre 100 et 300 Mots.

Informations sur les articles



La variance expliquée après réduction des vecteurs de poids du fichier Embeddings (de 250 à 100 features) reste importante = 98.7%.

Nous passons d'un fichier de vecteurs de 347Mo à 138Mo, ce qui pourrait faciliter un déploiement éventuel, en limitant l'impact sur nos performances.

Préparation des données

Processus de Nettoyage

Merge des données

articles_metadata & clicks_alll (385) using pd.merge

Recherche d'Outliers et Discrétisation

words_count (1-3 int) & session_size (1-4 int) using pd.cut

Nettoyage (*5)

article_id < 3
popularity_article <15 using "for_loop"

Séparation Train / Test

click_timestamp (¾ vs ¼)

Nettoyage Test vs Train

user_id présent

La discrétisation de certaines valeurs est une proposition pour considérer les variables associées aux intéractions utilisateurs, en modulant l'effet sur le modèle de Collaborative Filtering.
La séparation du Train vs Test dataset est faite sur une logique chronologique, et dans un but de valider la performance de nos modèles.

Back to AGENDA

Système de recommandation

Métriques et Mesure de la performance

Recherche des meilleures métriques

Metric: Top 5 Average Precision

Classement par pertinence des éléments sélectionnés par la requête

01

Metric: Hit Count Rate

Proportion de fois où l'utilisateur clique sur l'article qui lui aurait été recommandé depuis le Top5

02

Metric: Moyennes Top 5 Précision & Hit Count

03

2 mesures permettent d'évaluer la pertinence de nos prédictions. Les moyennes de celles-ci permettront d'analyser la performance des modèles sur l'ensemble du jeu de données

Hit Count Rate = hit_count / 5

L'Average précision à pour avantage de considérer l'ordre de la prédiction, et donc la pertinence des Top k

AP@k= 1/GTP ∑P@k ✕rel@k
(Voir ici)

Item-Based

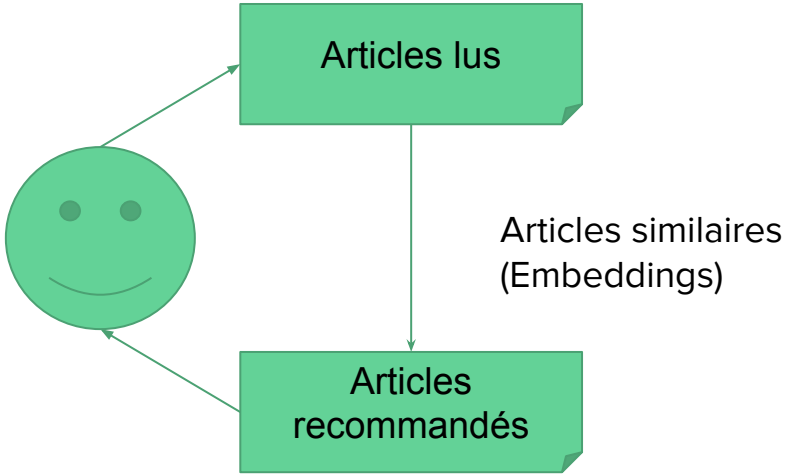
Système de recommandations Item Based

Ce système est basé sur l'analyse de similarité entre les contenus (ici les articles).

Notre modèle prend la moyenne des poids (sur les 250 Vecteurs d'embedding) sur les articles lus du train-set par utilisateur.

A partir de ces vecteurs, nous recherchons la similarité cosinus, pour proposer un top5 d'articles les plus proches..

In fine, nous mesurons si ces propositions correspondent au choix de l'utilisateur dans le test set.



123 Hit - 0.19% moyenne
Average Precision = 0, 0256 %
Temps écoulés sur la phase de validation: 340min

Model-Based

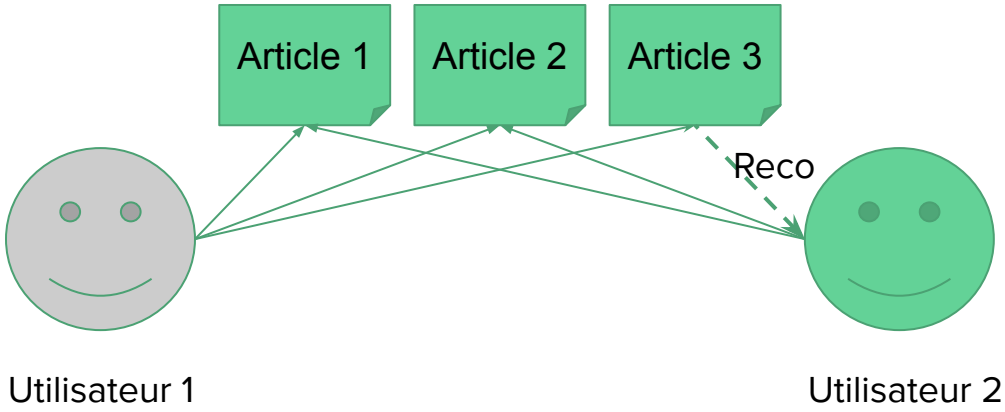
Collaborative Filtering Model Based

L’objectif ici est d'entraîner un modèle pour notre système de recommandation.

Pour se faire, nous allons générer un “rating”, pour représentation des intéractions Utilisateurs / Articles (voir slide suivante).

Ce Rating sera ensuite intégré dans une Matrice Utilisateur / Article / Rating.

Cette matrice (SparseMatrix) nous servira de base d’entrainement pour notre modèle Implicit ALS.



Note: Stratégie mal-adaptée à la problématique du Cold-Start (nouvel utilisateur ou nouvel article).

Recherche des meilleures Rating

01

Easy Rating

Rating = Nombre de click par article

02

Medium Rating

Rating = (Taille de session discrétisée) / (Nombre de click par article, par session)

03

Complex Rating

Rating = (Taille de session discrétisée) / (Nombre de click par article, par session * Nombre de mots discrétisé)

3 Matrices Utilisateur/Article/Ratings créées (*Sparse Matrix) à partir du jeu d'entrainement, au travers de l'utilisation de la librairie Scipy.*
Puis Entraînement du modèle de recommandations avec l'utilisation de la librairie Implicit (*AlternatingLeastSquares)*

84 Hit - 0.13% moyenne

Average Precision = 0, 0237 %

(Run in 30min)

126 Hit - 0.19% moyenne

Average Precision = 0, 0304 %

(Run in 30min)

110 Hit - 0.17% moyenne

Average Precision = 0, 0278 %

(Run in 30min)

Comparatif

Comparatif des systèmes de recommandations

	Item-Based	Model-Based
Avantages	Démarrage à froid (Nouvel utilisateur ou nouvel article). Facilité de mise en œuvre et d'interprétation.	Pas d'information individualisée à collecter/construire (RGPD OK) Temps de calcul ⇒ Ressource Capacité à proposer des articles variés.
Inconvénients	Nécessite des Embeddings à jour (Informations sur les articles = Travail de référencement préalable). Se limite à des propositions d'articles logiques (Peu d'hétérogénéité attendue). Temps de calcul ⇒ Ressource	Nécessité de créer un rating offrant une bonne représentativité des interactions Utilisateur / Article (vs Intérêt réel). Pas adapté au démarrage à froid.

Back to AGENDA

Déploiement de l’outil

Mobile App link & Azure Function link

Versioning GIT et Stockage GitHub

Stratégie de déploiement

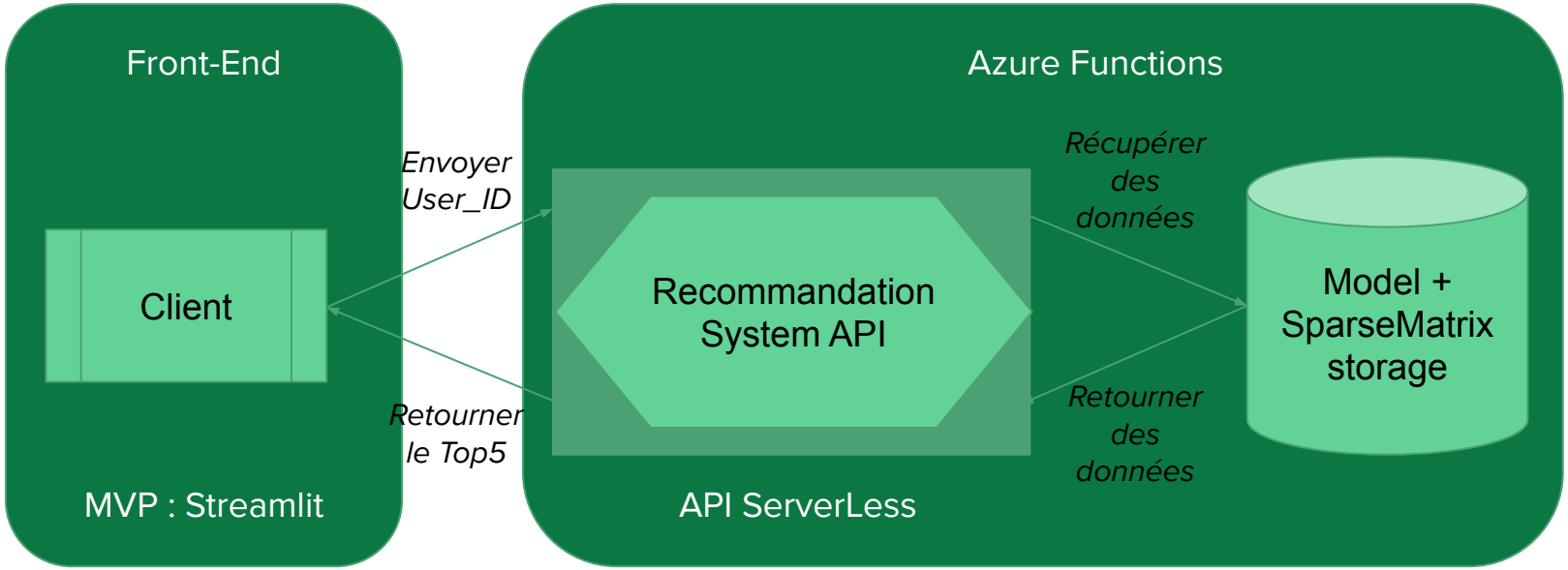
https://github.com/blanchonnicolas/IA_Project9_Openclassrooms_IA_SystemeRecommandation

Avec l'utilisation de Git et GitHub, nous permettons l'utilisation du modèle, des scripts et des données en accès concurrent (pour les équipes de Dev, Test, Métiers).

La chaîne d'intégration continue pourra être ainsi industrialisée afin d'aligner Backend / FrontEnd, avec l'utilisation des meilleurs modèles.

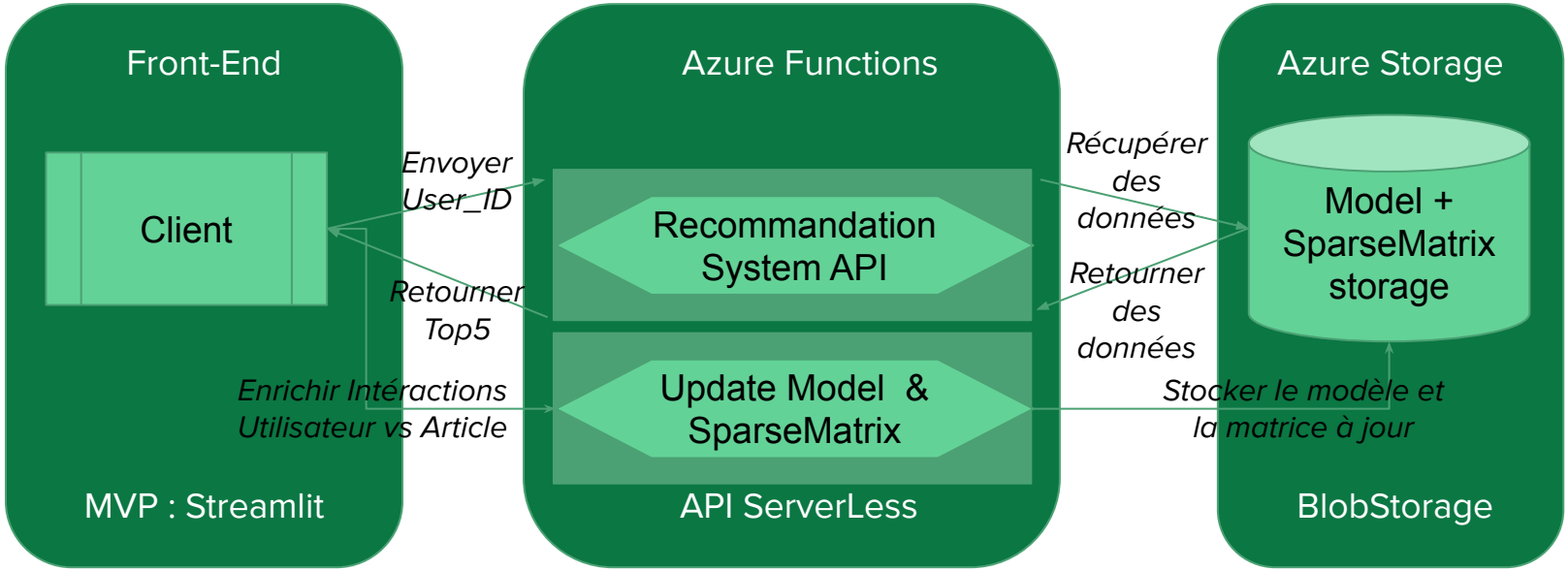
Architectures : AS-IS et TO-BE

Architecture AS-IS



Architecture basée sur Azure Functions, avec utilisation d'API Serverless (httptrigger détectant un événement)
Stockage des donnée figé, obligeant redéploiement complet pour chaque mise à jour (ex: Ajout utilisateur / article)

Architecture TO-BE



L'architecture To-Be stockera les données utilisateurs et de sessions en base de données, ce qui permettra de mettre à jour nos modèles et matrice (tjs via l'utilisation d'Azure Functions)

Back to AGENDA

Demo

Mobile App link & Azure Function link

Video link

Conclusions

Scripts permettant d'entraîner et de tester le modèle de recommandation.
Fonction serverless renvoyant la prédiction du modèle.

Scripts stockés dans un dossier GitHub.
Mise en place des fondations pour l'intégration continue.

Comparaisons de 2 approches pour le système de recommandations.
Création de rating pertinents, sur la base de valeurs discrétisées.
Construction de Matrice Utilisateur/Article pour l'entraînement d'un modèle

Optimisation du modèle avec l'approche Hybride (ColdStart), et l'optimisation du Rating utilisé.
Amélioration de l'architecture, pour la récupération des interactions, et la mise à jour du modèle.