



My Content

Réalisez une application de recommandation de contenu

Étudiant : Dhaker KACEM

Mentor : Samir TANFOUS

Sommaire

1)Présentation du projet

2)Présentation des données

3) Modélisation

4)Déploiement

5)Conclusion

Présentation du projet

- *Il s'agit de réaliser un premier MVP qui prendra la forme d'une application de recommandation d'articles et de livres,*
- *Notre méthodologie consistera à utiliser le jeu de données fournis pour tester deux techniques: Based Content Filtering et Collaborative Filtering,*
- *Une 1ère application avec la Cosine Similarity et une 2ème avec le modèle SVD,*
- *Nous mettrons, par la suite, en place une architecture serverless pour un déploiement en production,*

Présentation du dataframe

- Les données sont disponibles sur Kaggle et fournies par Globo.com
- Nous avons en tout 4 fichiers:
 - *articles_metadata.csv* : informations sur les articles,
 - *articles_embeddings.pickle* : embeddings Tf-Idf des articles,
 - *clicks_sample.csv* : informations sur les sessions,
 - *clicks* : dossier avec 385 fichiers contenant l'ensemble des informations sur les sessions utilisateurs (un fichier/heure),
- « *clicks_sample.csv* » et « *articles_metadata.csv* » ne seront pas utilisées,
- En tout nous avons ,
 - 364047 articles,
 - 2 988 181 clicks,
 - 322897 utilisateurs,
 - 250 colonnes d'embeddings des articles,

Modélisation

- Deux techniques seront utilisées:
- Content Based Filtering : c'est une technique qui consiste à suggérer du contenu à un utilisateur sur la base de ses préférences.
- Collaborative Filtering : c'est une technique qui vise à prédire les préférences d'un utilisateur en se basant sur les préférences d'utilisateurs similaires (ayant lus les mêmes articles).

Modélisation

Content Based Filtering

- Nous utilisons la Cosine Similarity entre les articles lus par l'utilisateur et tous les autres articles,
- La cosine similarity mesure l'angle entre deux vecteurs dans un espace multidimensionnel,
- Pour réaliser la Cosine Similarity, 3 scénarios possibles:
 - Soit, le dernier article lu est pris en compte,
 - Soit, un article pris au hasard (parmi ceux lus par l'utilisateur),
 - Soit, une moyenne des embeddings des articles lus est prise en compte,



Modélisation

Content Based Filtering

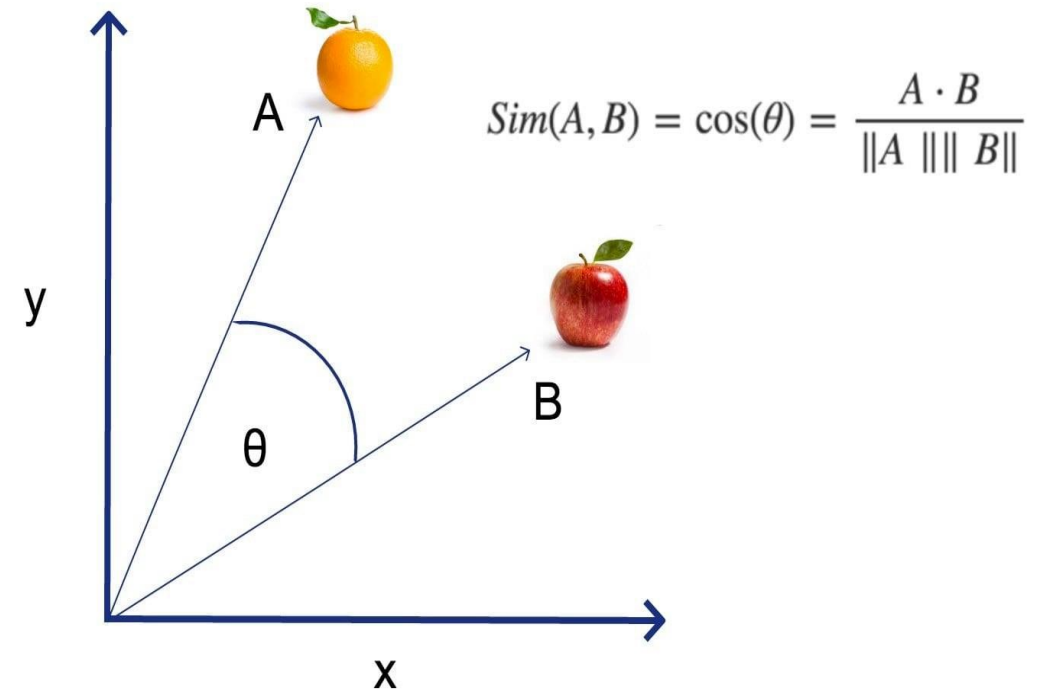
	Stratégie	Qualités	Faiblesses
0	Dernier Article Cliqué	- Pertinence récentes\ Simplicité\ Recommandations opportunes	- Contexte limité\ Sensibilité au bruit\ Perspectives historiques limitées
1	Article Cliqué au Hasard	- Perspectives diverses\ Réduction de l'impact du bruit\ Flexibilité	- Incohérence\ Prévisibilité réduite
2	Moyenne des Embeddings	- Vue globale\ Stabilité\ Robustesse	- Dilution des intérêts spécifiques\ Complexité de calcul\ Adaptation lente aux nouveaux intérêts

Modélisation

Content Based Filtering

- Dans un premier temps, nous réalisons nos calculs en utilisant la matrice des embeddings sans ACP,
- Elle est définie comme le produit scalaire des deux vecteurs divisés par le produit de leurs normes,
- Si la valeur de la cosine similarity est proche de 1, cela indique une forte similarité.
- En revanche, une valeur proche de 0 indique une similarité faible.
- Notre modèle recommande les 5 articles les plus similaires,

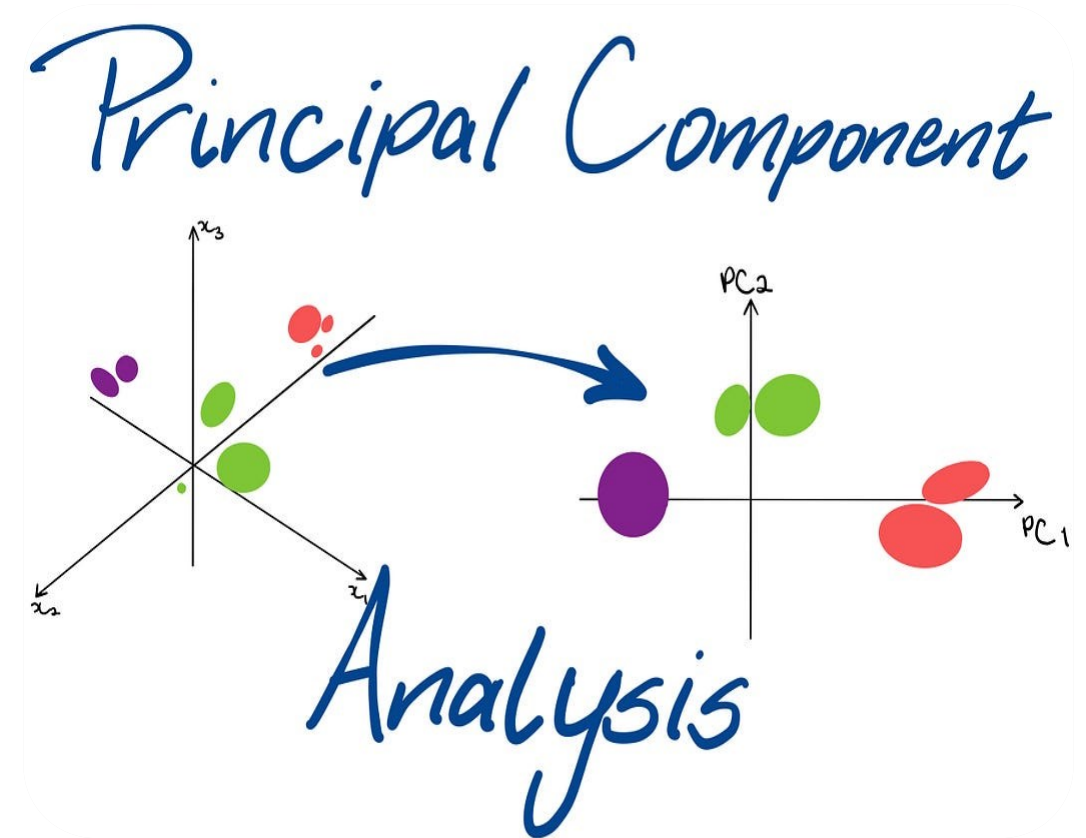
Cosine Similarity



Modélisation

Content Based Filtering

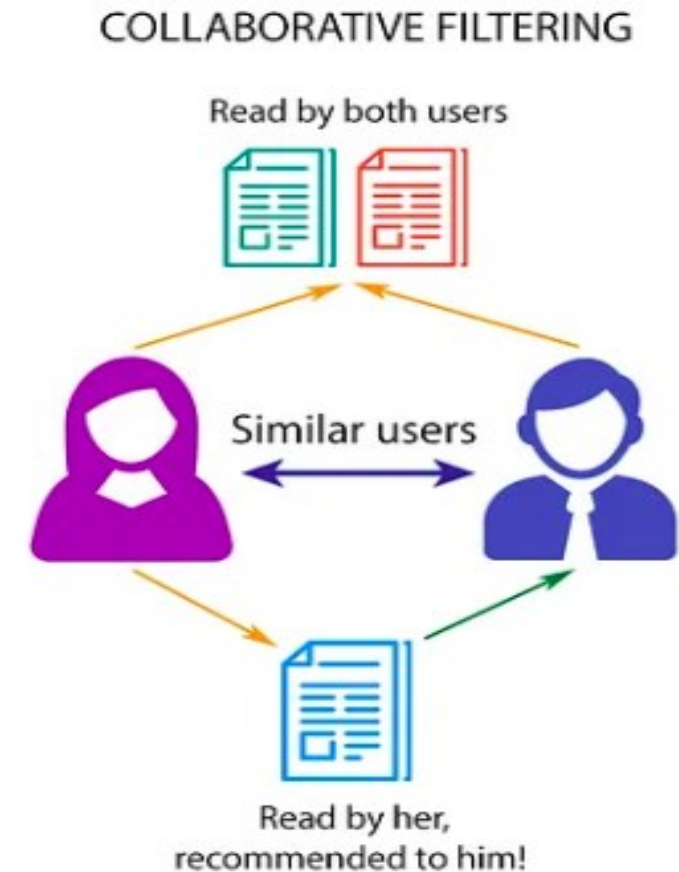
- Dans un deuxième temps, nous réalisons une ACP sur notre matrice des embeddings,
- Nous gardons les 80 premières composantes,
- Celles-ci captent 0,9821 de l'inertie totale,
- Les résultats sont, cependant, différents selon que l'on utilise la matrice avec ACP ou sans ACP,
- Nous nous attendions à des résultats similaires,



Modélisation

Collaborative Filtering

- Nous utilisons un modèle SVD (Singular Value Decomposition) implémenté dans la bibliothèque Surprise,
- SVD permet de prédire les scores qu'aurait donné un utilisateur à des articles non lus sur la base des notes qu'ont effectivement accordé des utilisateurs similaires à ces mêmes articles,
- Nous calculons un score implicite pour pouvoir entraîner notre modèle,
- Nous divisons le nombre de clicks sur un article donné par l'ensemble des clicks réalisés par l'utilisateur,



Modélisation

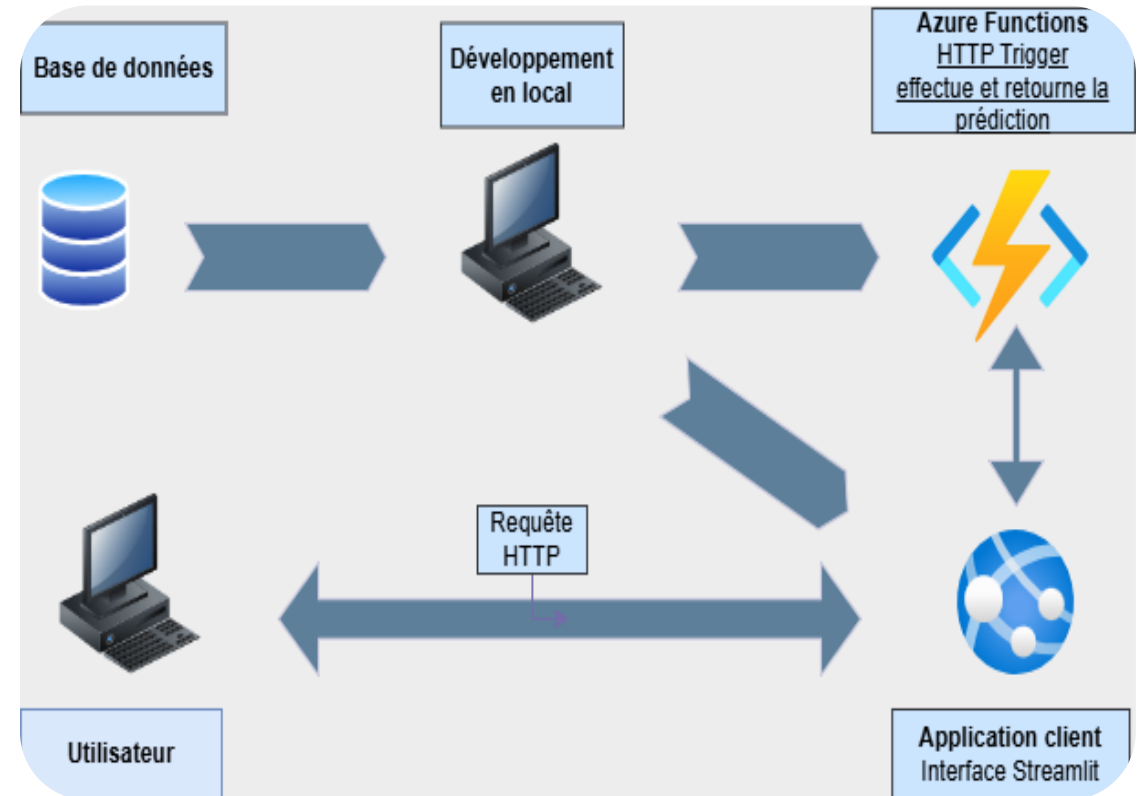
Collaborative Filtering

- Nous utilisons un dataframe composé du User_id, click_article_id et de notre score implicite,
- Nous Divisons nos données en train set et test set,
- Nous entraînons notre modèle sur les données d'entraînement,
- Nous obtenons un score RMSE(Root Mean Square Error) de 0,1729 sur les données de test,
- Le modèle retourne un score compris entre 0 et 1 qui correspond à une prédiction de notre score implicite,
- Pour chaque utilisateur, seront recommandés les 5 articles qui obtiennent les meilleurs scores,

Déploiement

Architecture MVP

- *Nous utilisons une architecture serverless pour le déploiement,*
- *Nous utilisons Azure Functions,*
- *La création du projet, de l'Azure Functions et son déploiement se fait sur VS Code,*
- *Notre Azure Functions est de type HTTP Trigger (déclencheur HTTP),*



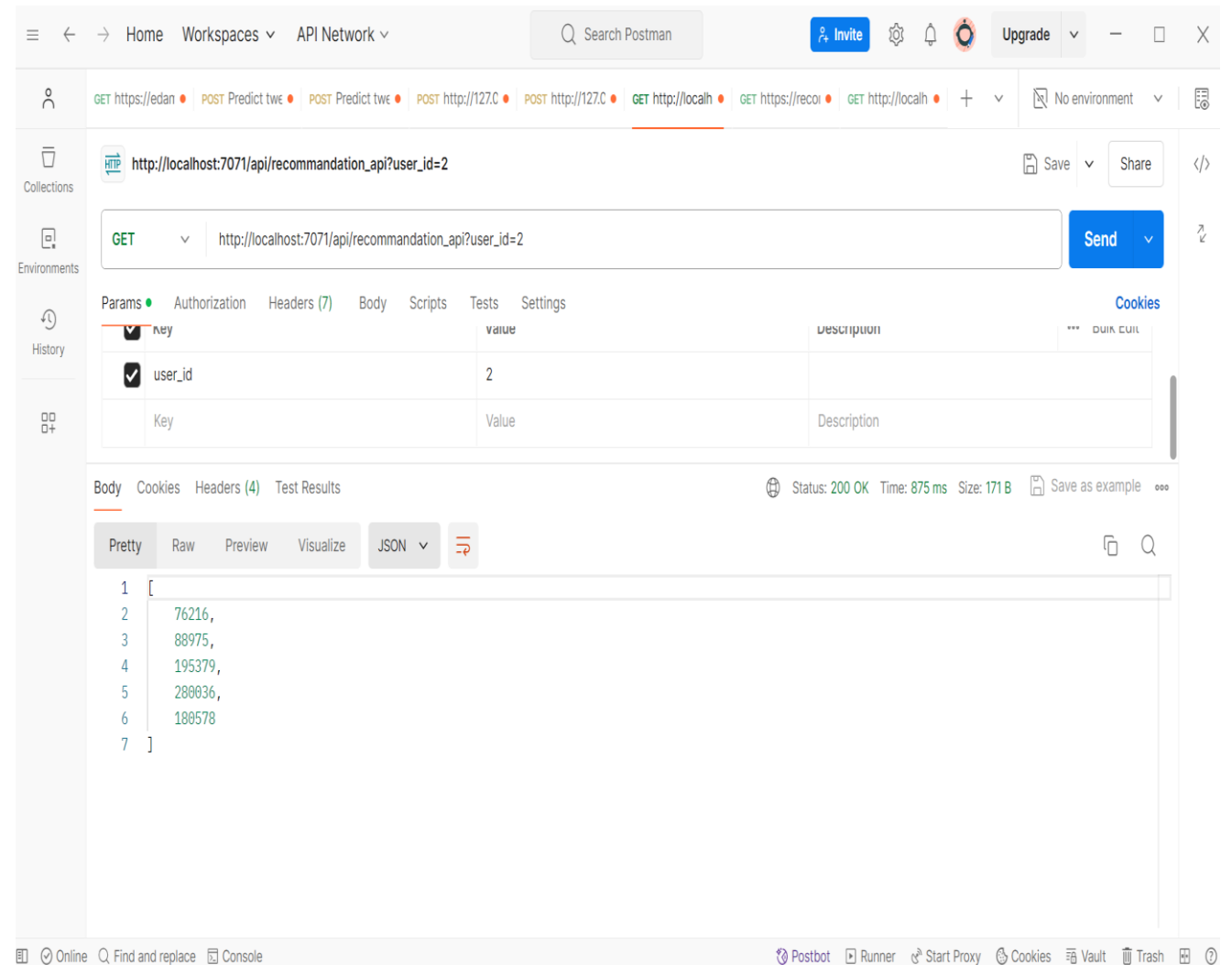
Déploiement

- *Nous déployons uniquement notre modèle SVD,*
- *Nous créons, d'abord, notre projet via VS Code en local,*
- *Nous procédons, par la suite à la création de notre Azure Functions,*
- *La dernière étape consiste en son déploiement et son exécution sur Azure,*
- *Le modèle et les données au format Pickle sont uploadés automatiquement,*

Déploiement

Test en local

- *L'API est testée localement avant sa mise en production,*
- *Le test est réalisé sur Postman,*
- *L'API fonctionne correctement en local,*
- *Elle prend en entrée un user_id et renvoie une recommandation de 5 articles,*



Déploiement

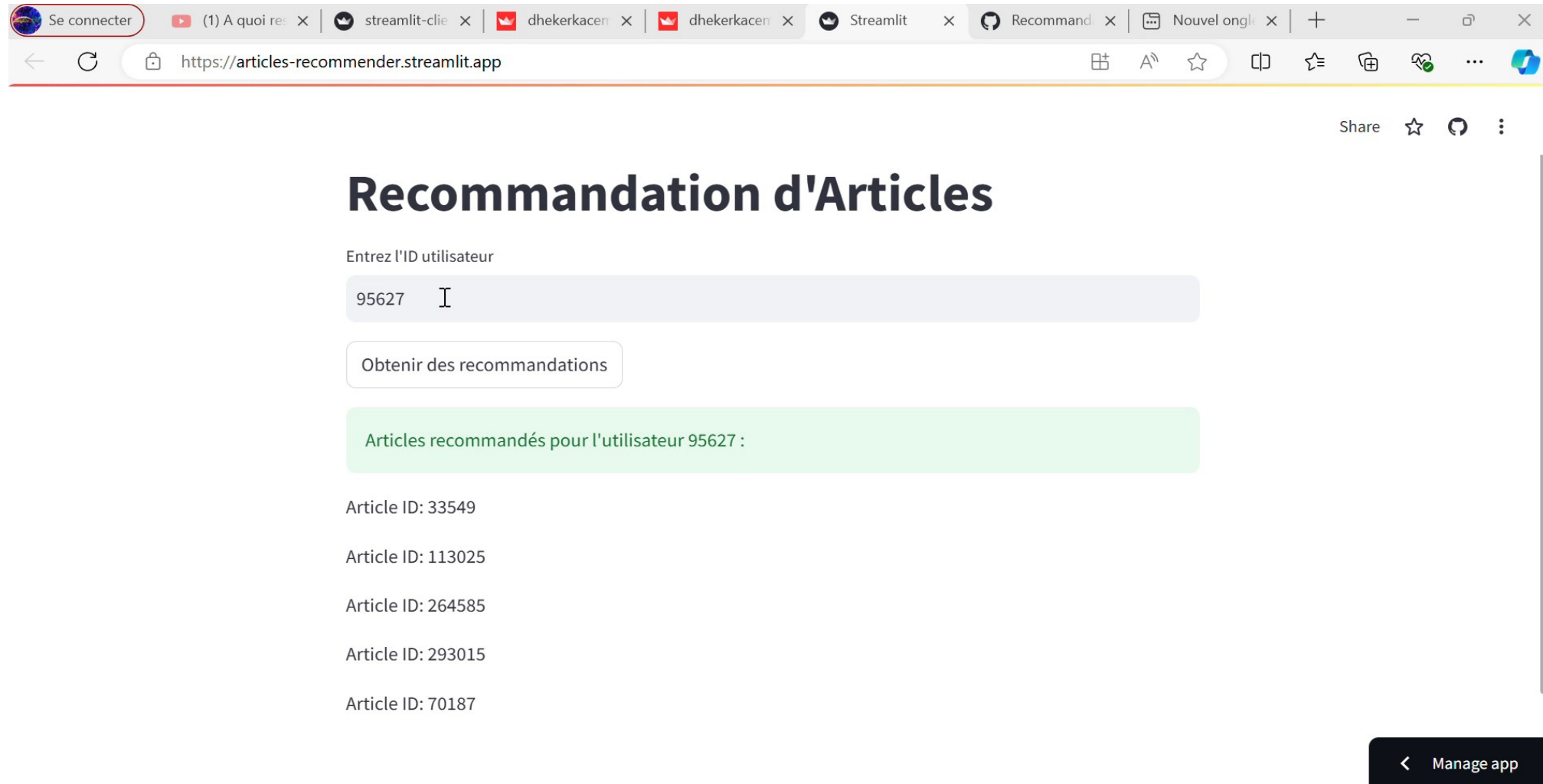
Application Streamlit

- La partie frontend est assuré par une application Streamlit,
- Notre application Streamlit permet de saisir un `user_id`, de l'envoyer à l'API qui effectue la prédiction et renvoie une recommandation de 5 articles ,
- Cette application est hébergée directement sur Streamlit Cloud,

```
1 import streamlit as st
2 import requests
3 import json
4
5 # URL de l'API de recommandation
6 API_URL = "https://recommenderapi.azurewebsites.net/api/recommandation_api"
7
8 # Titre de l'application
9 st.title("Recommandation d'Articles")
10
11 # Entrée pour le user_id
12 user_id = st.text_input("Entrez l'ID utilisateur", "")
13
14 # Bouton
15 if st.button("Obtenir des recommandations"):
16     if user_id:
17         try:
18             # Appel à l'API de recommandation
19             response = requests.get(API_URL, params={"user_id": user_id})
20
21             # Vérification de la réponse de l'API
22             if response.status_code == 200:
23                 recommendations = response.json()
24                 st.success(f"Articles recommandés pour l'utilisateur {user_id} :)")
25                 for article_id in recommendations:
26                     st.write(f"Article ID: {article_id}")
27             else:
28                 st.error(f"Erreur {response.status_code}: {response.text}")
29         except Exception as e:
30             st.error(f"Erreur lors de l'appel de l'API: {str(e)}")
31     else:
32         st.warning("Veuillez entrer un ID utilisateur valide.")
33
```

Déploiement

Démonstration



The screenshot shows a web browser window with the URL `https://articles-recommender.streamlit.app`. The page title is "Recommandation d'Articles". It features a form with a label "Entrez l'ID utilisateur" and a text input containing "95627". Below the input is a button labeled "Obtenir des recommandations". A green box displays the heading "Articles recommandés pour l'utilisateur 95627 :". Below this, five article IDs are listed: "Article ID: 33549", "Article ID: 113025", "Article ID: 264585", "Article ID: 293015", and "Article ID: 70187". In the bottom right corner, there is a dark button labeled "< Manage app".

Se connecter

(1) A quoi res... x

streamlit-clie x

dhekerkacem x

dhekerkacem x

Streamlit x

Recommand x

Nouvel onglet x

+

←

↻

🔒

https://articles-recommender.streamlit.app

🔍

A

☆

📄

🔖

🔗

👤

...

🌐

Share

☆

🔗

⋮

Recommandation d'Articles

Entrez l'ID utilisateur

95627

Obtenir des recommandations

Articles recommandés pour l'utilisateur 95627 :

Article ID: 33549

Article ID: 113025

Article ID: 264585

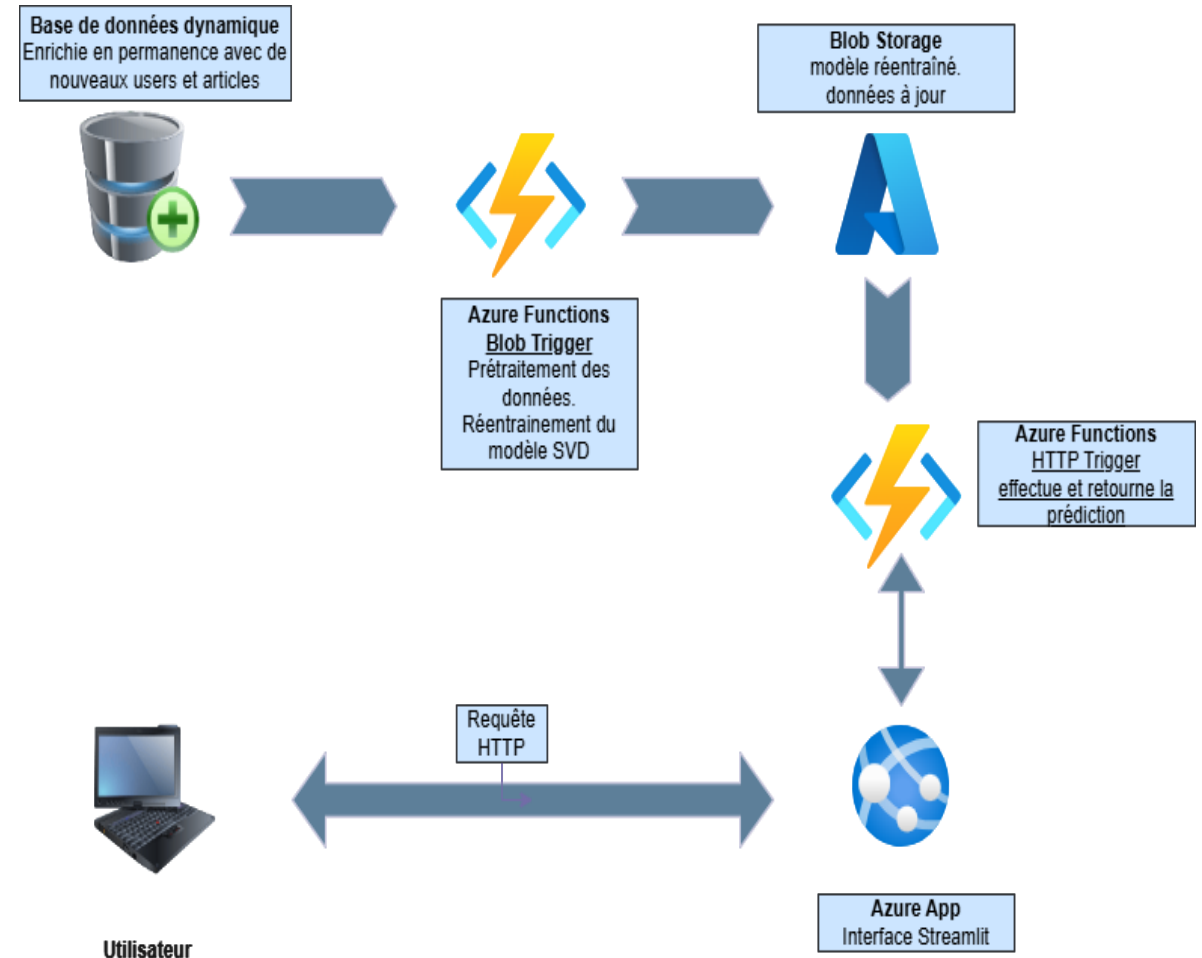
Article ID: 293015

Article ID: 70187

< Manage app

Architecture Cible

- *Un contexte de production est différent d'un contexte MVP,*
- *L'architecture doit évoluer pour prendre en considération les contraintes liées à une mise en production,*
- *Elle doit, également, pouvoir gérer automatiquement l'ajout de nouveaux utilisateurs et articles,*
- *Deux Azure Functions sont mise en place, une de type HTTP Trigger (déclencheur HTTP) et une autre de type Blob Trigger,*



Conclusion

- *Dans ce projet, nous avons pour mission de réaliser un premier MVP d'une application de recommandation de contenu,*
- *Nous avons utilisé deux techniques: la Content Based Filtering et la Collaborative Filtering,*
- *Notre modèle SVD basé sur la technique du Collaborative Filtering a été déployé avec succès en serverless avec Azure Functions,*
- *Dans une logique de mise en production, quelques ajustements s'imposent:*
 - *D'abord, mettre en place l'architecture cible qui permettra de gérer automatiquement l'ajout de nouveaux utilisateurs et articles,*
 - *Créer un modèle hybride qui tire un maximum de profit des avantages des deux techniques tout en minimisant leurs inconvénients,*
- *Ceci permettra, entre autre, de mieux gérer la problématique du « cold start »,*

Annexes

- *Repository : Recommandation_APP: https://github.com/DhekerKacem/Recommandation_App*