Programme d’Ingénierie en Intelligence Artificielle – OC IA P10 - Réalisez une application de recommandation de contenu

Le projet consiste à concevoir et **déployer un moteur de recommandation** d’articles basé sur les **données d’interaction** des utilisateurs (clics, vues). Après une **analyse exploratoire** et un **filtrage basé sur le contenu** (approche Content-Based), nous avons finalement **retenu le filtrage collaboratif (Collaborative Filtering)** comme méthodologie principale. Le **modèle ALS** (Alternating Least Squares) s’est révélé le plus performant sur les différentes **métriques d’évaluation** (MAP, NDCG, précision).

La **phase d’industrialisation** s’appuie sur une **architecture serverless** grâce à **Azure Functions**, qui expose une **API** pour traiter les requêtes de recommandation à la demande. Les fichiers de données (CSV, matrices .npz) sont **stockés sur Azure Blob Storage** et récupérés en temps réel. **Une application Streamlit** assure la partie interface utilisateur : en entrant simplement un **user_id**, on obtient un **top 5 d’articles recommandés**.

Des **évolutions futures de ce MVP** pourraient inclure l’intégration d’un mécanisme pour gérer efficacement les **nouveaux utilisateurs**, l’ajout dynamique de **nouveaux articles**, et la **mise à jour automatique du modèle** (re-training) pour suivre l’évolution des interactions.

patrice preudhomme 02/2025

Préambule

Cette présentation s'adresse autant aux **responsables produit** (ou contenus) souhaitant comprendre la **valeur métier** d'un système de recommandation qu'aux **professionnels de la Data Science** cherchant à en connaître les **aspects techniques**. Elle résume les **étapes clés** de la démarche et présente les **résultats d'analyse** les plus pertinents.

Un **compromis** a été recherché pour offrir des **explications accessibles** tout en fournissant assez de **détails techniques** sur :
- la préparation et l'analyse des **données d'interaction**,
- la **modélisation** (filtrage collaboratif vs. filtrage basé sur le contenu),
- l'**évaluation** à travers des métriques (MAP, NDCG, etc.),
- l'**industrialisation**** (**Azure Functions, Streamlit**).

Bien que ce support n'inclue pas le **code** en détail, les techniciens intéressés peuvent consulter le **notebook** et **dépôts GitHub** pour plus d'informations et une **mise en œuvre** complète. Les **non-experts** en Data Science sont invités à poser leurs **questions** lors de la présentation pour éclaircir les **concepts techniques**.

Sommaire

1) Liens github et de l'application

- **Dépôt Notebook (R&D)** :
  https://github.com/preudh/OC_IA_P10_Recommandation_contenu

- **Dépôt Azure Functions (API)** :
  https://github.com/preudh/OC_IA_P10_RecoFunction

- **Dépôt Application Streamlit (Front-end)** :
  https://github.com/preudh/OC_IA_P10_streamlit_app

- **Lien de l'application déployée :**
  https://p10-streamlit-app-2025.azurewebsites.net/

2) Rappel de la problématique

Entreprise : My Content est un service en ligne où des **articles** sont régulièrement publiés pour des lecteurs (utilisateurs). Le défi principal consiste à **recommander ces articles de façon pertinente et personnalisée.**

Problématique actuelle :

Les utilisateurs reçoivent souvent des **propositions de contenus peu adaptés à leurs centres d'intérêt, ce qui impacte l'engagement et la satisfaction**.

La base d'articles s'agrandit rapidement, **rendant difficile toute sélection manuelle.**

L'entreprise anticipe une **croissance exponentielle du volume de contenus**, justifiant la **nécessité d'un moteur de recommandation automatisé.**

Objectifs :

- Étudier la faisabilité d'un système de recommandation s'appuyant sur les données d'interaction (clics, vues), afin d'optimiser la pertinence et l'expérience utilisateur.

- Mettre en place un algorithme de filtrage collaboratif (ou une autre approche comme le content-based) capable de s'adapter à un large catalogue d'articles.

- **La phase d'industrialisation** repose sur un **MVP serverless (Azure Functions)**, qui expose une API de recommandation.
Les données (CSV, matrices .npz) sont hébergées sur **Azure Blob Storage** et consultées en temps réel.
Une **application Streamlit** (en saisissant un user_id) renvoie un top 5 d'articles personnalisé

3) Présentation des jeux de données

Présentation du Jeu de Données (Observations sur les “clicks”)

Nous disposons de **385 fichiers CSV** totalisant :

- **322 897 utilisateurs uniques** (certains multipliant les sessions),
- **1 048 594 sessions** et **2 988 181 clics** enregistrés,
- **46 033 articles** distincts cliqués, dont l’article le plus populaire (ID : 160974) recense **37 213 clics**.

D’autres colonnes (pays, environnement, référents, etc.) révèlent un **biais fort** : un **environnement** majoritaire (2,9 M de clics), un **pays** regroupant 2,85 M de clics, et un **référent principal** (1,6 M de clics).

Pour simplifier le travail sur un **jeu de données volumineux (~3 M de lignes)**, nous utilisons un **fichier échantillon (`clicks_sample.csv`) e**n l**imitant les temps de calcul**. **Les analyses et entraînements** (modèles de recommandation) se baseront donc sur cet échantillon pour des raisons de **performance et de praticité**.

4) Analyse des clics utilisateurs

Objectif : Comprendre l'activité des utilisateurs en analysant **leurs sessions** et **leurs clics** sur différents articles.

1. **Début de session**
- Un **user_id** lance une **session_id** à un moment précis (**session_start**).
- Chaque session inclut un **nombre d'articles (session_size)** visionnés.

2. **Enregistrement des clics**
- Chaque **ligne** du CSV représente **un clic** (ex. `click_article_id`).
- Le **timestamp** (`click_timestamp`) précise l'instant exact de consultation.
- Plusieurs clics peuvent se rattacher **à la même session**.

3. **Contexte du clic**
- **Environnement** (mobile, desktop), appareil (smartphone, PC), OS (Windows, iOS, etc.).
- **Pays/Région** : localise l'utilisateur.
- **Référent** (moteur de recherche, réseau social) : indique la source du trafic.

4. **Fin de session**
- Tous les clics d'une session se regroupent via le **session_id**.
- La **durée** peut se calculer à partir des timestamps (différence entre premier et dernier clic), permettant de mesurer l'**engagement**.

5) présentation des différents modèles analysés et de leurs avantages et inconvénients - Méthodes de Filtrage Collaboratif

	model	mean_cos_similarity	precision	map	ndcg
0	AlternatingLeastSquares	0.454395	0.04301	0.02514	0.03099
1	LogisticMatrixFactorization	0.461744	0.01882	0.00526	0.00871
2	BayesianPersonalizedRanking	0.512054	0.03226	0.02149	0.02500

Filtrage Collaboratif

- Principe : recommander à un utilisateur des articles “aimés” par d’autres utilisateurs ayant un profil similaire (mêmes clics, vues, etc.).

- Avantages : efficace lorsque beaucoup d’historique d’interactions est disponible ; plus la base de clics est riche, plus les recommandations gagnent en pertinence.

- Inconvénients : problème du cold start (nouveaux utilisateurs ou nouveaux articles sans historique) ; nécessité d’une quantité importante de données.

Meilleur modèle : AlternatingLeastSquares (ALS)

Très bonne performance globale en précision, MAP, NDCG. Facile à déployer sur de grands volumes de données implicites (clics).

5) présentation des différents modèles analysés et de leurs avantages et inconvénients - Filtrage Basé sur le Contenu (Content-Based)

	model	rmse	mse	mae
0	SVD	0.073214	0.005360	0.064250
1	BaselineOnly	0.107112	0.011473	0.097903
2	CoClustering	0.363128	0.131862	0.326518

Filtrage Basé sur le Contenu (Content-Based)

- Principe : s'appuie sur les caractéristiques intrinsèques des articles (titre, texte, tags, etc.) pour déterminer la similarité avec des articles déjà consultés.

- Avantages : pas besoin d'un large historique utilisateur ; fonctionne pour les nouveaux articles si leurs descripteurs sont disponibles.

- Inconvénients : nécessite des métadonnées ou descriptions précises ; peut être moins performant lorsqu'il y a beaucoup d'articles très variés ou peu de détails textuels.

Meilleur Modèle :
- **SVD est le modèle le plus performant**, obtenant les **meilleurs scores (donc les plus faibles) en RMSE, MSE et MAE**, ce qui signifie qu'il offre des **prédictions plus précises et plus fiables**.

<u>Modèle et Méthode retenus</u> => le filtrage collaboratif (Collaborative Filtering) comme méthodologie principale. Le **modèle ALS** (Alternating Least Squares) s'est révélé le plus performant sur les différentes **métriques d'évaluation** (MAP, NDCG, précision).

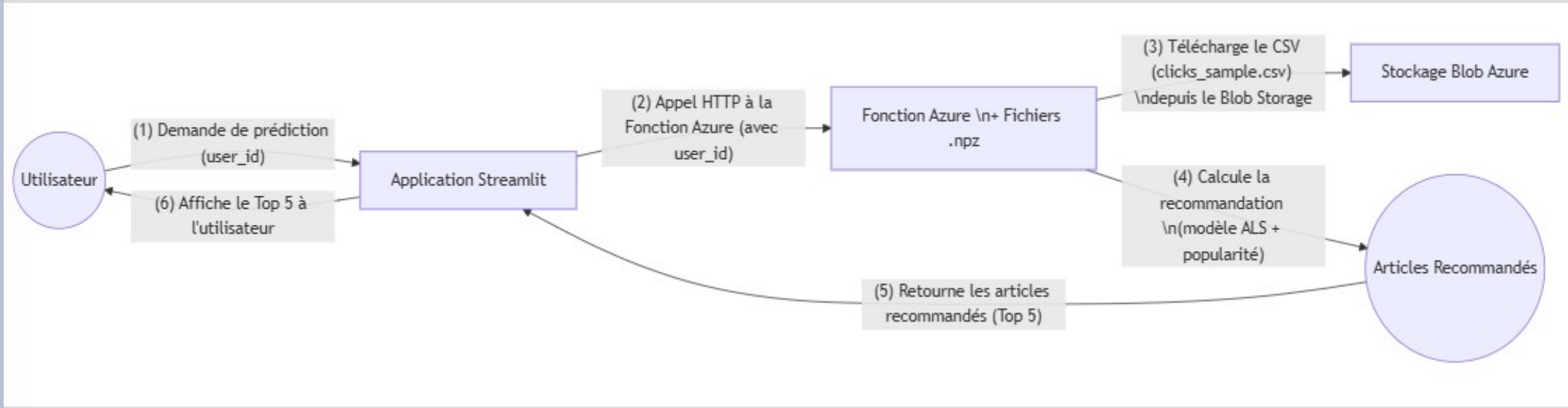
6) Schéma de l’Architecture Retenue

Schéma de l’Architecture Retenue
Le schéma dans le slide suivant illustre comment chaque composant interagit au sein du MVP :
1. **Azure Functions** : Héberge le code de recommandation dans `recommend_articles/__init__.py`. Les fichiers `.npz` (modèle ALS et matrice de popularité) sont présents **localement** dans le répertoire de la fonction.
2. **Azure Blob Storage** : Contient le **CSV** des interactions utilisateur (`clicks_sample.csv`). La fonction Azure appelle `shared/azure_blob.py` pour télécharger ce fichier en temps réel.
3. **Application Streamlit** : Fait une requête à l’API Azure (via HTTP) en fournissant un `user_id`. L’API charge les données depuis Blob (CSV) et utilise les **.npz** déjà disponibles localement pour effectuer la recommandation.
4. **Réponse** : L’API renvoie une liste d’articles recommandés, que Streamlit affiche sous forme d’un top 5.

« csr_article_popularity.npz » sert de point de départ : il indique “qui a cliqué sur quoi” (les données d’interactions).
« als_implicit_model.npz » utilise justement ces informations de clic (ainsi que celles de nombreux autres utilisateurs) pour calculer les articles les plus susceptibles de plaire à chaque utilisateur.

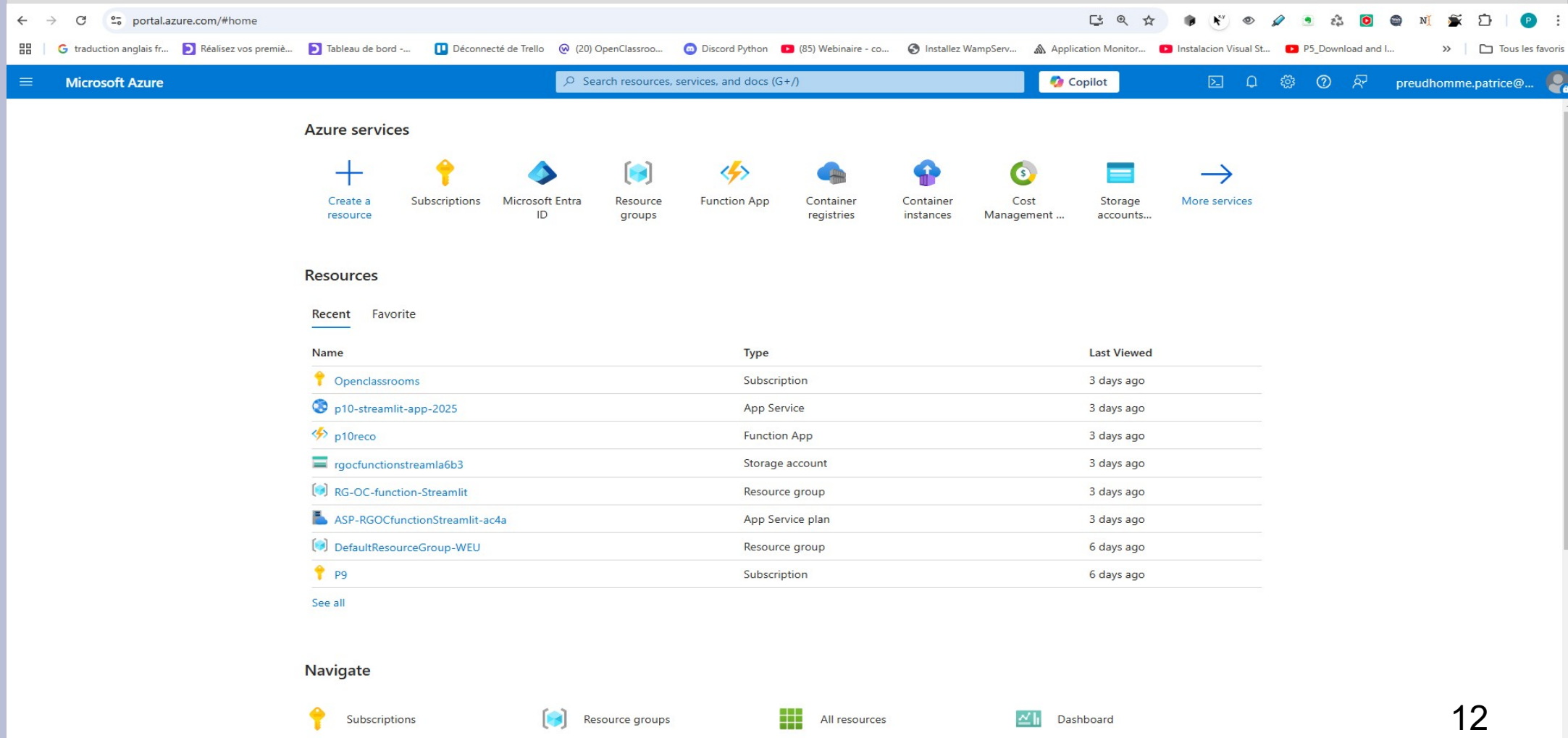
En résumé, la matrice CSR décrit le passé (les clics déjà effectués), tandis que le modèle ALS utilise ces données pour faire des prédictions sur les articles à recommander.

6) Schéma de l’Architecture Retenue (suite)



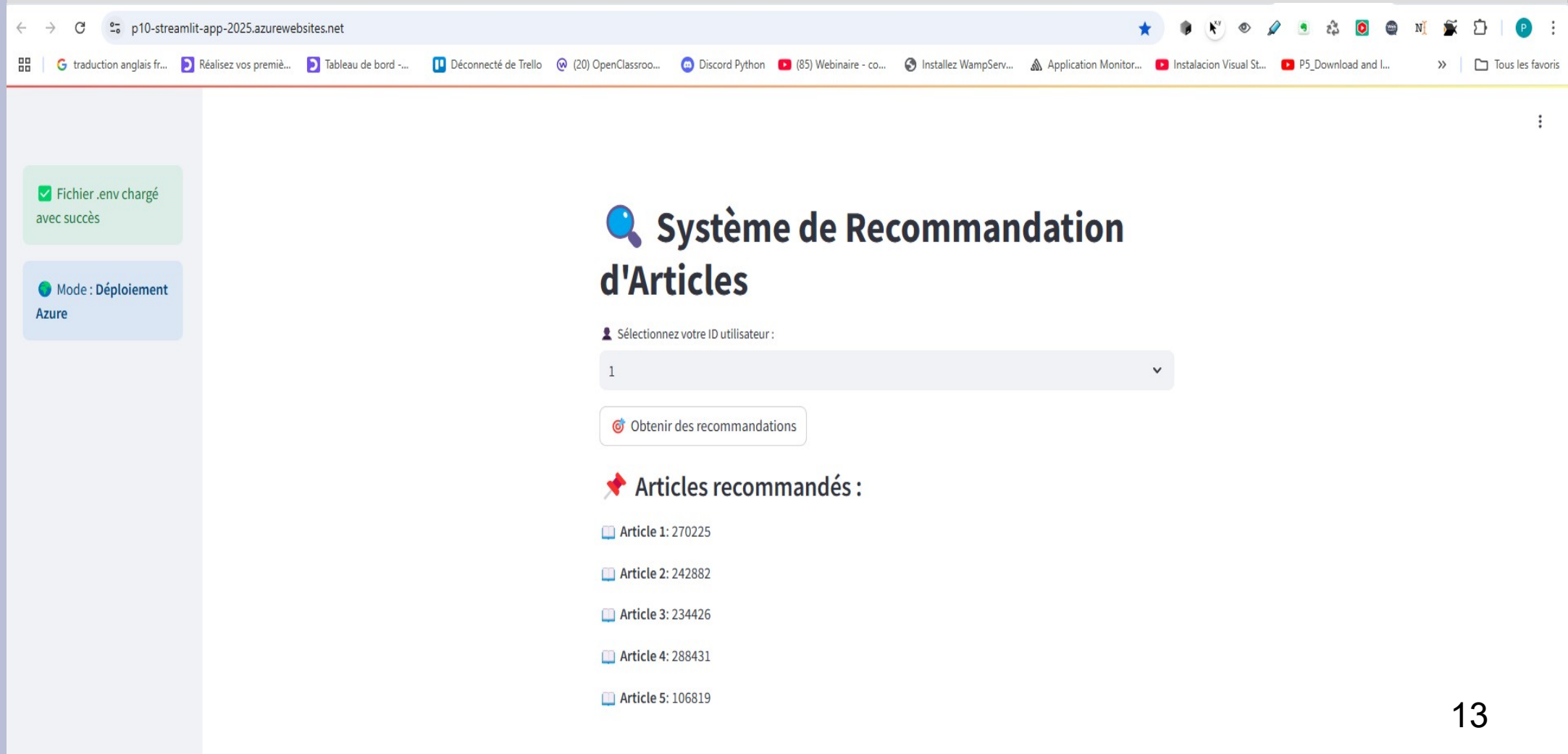
7) Présentation du système de recommandation utilisé

La souscription « Openclassrooms » regroupe l'ensemble de nos ressources : l'application Streamlit est hébergée sur l'App Service « p10-streamlit-app-2025 », la logique de recommandation réside dans la Function App « p10reco » et le fichier CSV des interactions est stocké dans « rgocfunctionstreamla6b3 ». Le groupe de ressources « RG-OC-function-Streamlit » associe ces différents éléments, tandis que « ASP-RGOCfunctionStreamlit-ac4a » est le plan App Service qui fournit l'infrastructure nécessaire.



7) Présentation du système de recommandation utilisé (suite)

La capture d'écran ci-dessous illustre l'interface **Streamlit** de notre système de recommandation. L'utilisateur y sélectionne son **ID** dans une liste déroulante, puis clique sur le **bouton « Obtenir des recommandations »**. L'application envoie alors une requête à l'Azure Function, qui analyse l'historique d'interactions et retourne un **Top 5** d'articles les plus pertinents, affichés en temps réel.



8) Schéma de l'architecture cible permettant de prendre en compte la création de nouveaux utilisateurs et de nouveaux articles.

Approche proposée pour gérer la création de nouveaux utilisateurs et de nouveaux articles

Dans l'architecture cible, nous prévoyons d'étoffer le flux de données pour prendre en compte l'évolution constante de la plateforme (3 M de clics, 322 897 utilisateurs, 46 033 articles). Les points suivants décrivent comment intégrer ces changements :

- **Nouveaux articles** : Lorsqu'un nouvel article est créé, il est ajouté au stockage (ex. Azure Blob). Une routine de mise à jour (planifiée ou à la demande) intègre ces contenus dans le modèle de recommandation (recalcul partiel ou rafraîchissement des matrices).

- **Mises à jour et suppressions** : Les modifications de métadonnées (titre, catégorie) et la suppression d'articles obsolètes sont gérées par le même mécanisme, garantissant que le système ne propose pas de contenus périmés.

- **Nouveaux utilisateurs** : Lors de l'inscription, un enregistrement est créé en base. En attendant que l'algorithme ALS capte leurs préférences, le système peut proposer par défaut les articles les plus populaires (logique de démarrage à froid).

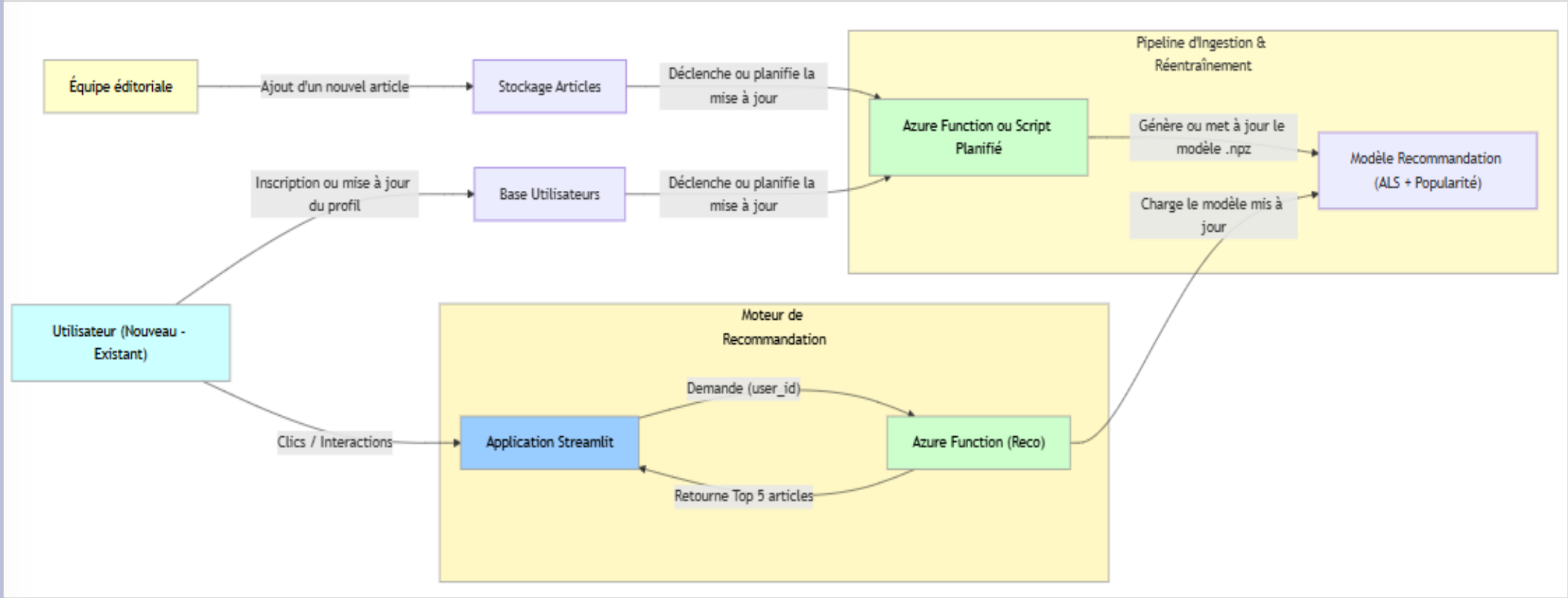
- **Mise à jour des profils** : Les utilisateurs peuvent modifier leurs préférences (centres d'intérêt, catégorie favorite, etc.) ; le modèle en tient compte pour affiner les recommandations au fil du temps.

- **Gestion des comptes inactifs** : Des comptes désactivés ou inactifs sont régulièrement retirés de la base, afin d'alléger la charge et d'éviter d'influencer négativement le modèle.

Cette **extension de l'architecture** n'est pas encore mise en place, mais elle décrit comment maintenir un système de recommandation **dynamique** et **scalable**, capable d'intégrer de nouvelles données grâce à des pipelines d'ingestion ou des déclencheurs (Azure Functions ou scripts planifiés). L'objectif final est de conserver une pertinence élevée des recommandations pour tous les utilisateurs, même en présence d'un important volume de clics et d'articles.

8) Schéma de l'architecture cible permettant de prendre en compte la création de nouveaux utilisateurs et de nouveaux articles (suite)

Ce schéma illustre la manière dont l'architecture cible gère en continu l'arrivée de nouveaux articles ou utilisateurs : l'ajout ou la mise à jour des données déclenche la réactualisation du modèle (via une Azure Function et/ou un script planifié), tandis que Streamlit sert d'interface pour consulter et afficher les recommandations mises à jour.



9) Conclusion

Ce projet s'appuie sur un échantillon **(clicks_sample.csv)** issu de plus de **3 millions de clics** et **364 000 articles**, permettant de **valider l'approche collaborative (ALS)** tout en limitant le temps de calcul. À l'avenir, un **système hybride** — combinant **filtrage collaboratif** et **analyse de contenu** — offrirait plusieurs avantages :

- **Améliorer les recommandations** et **gérer le démarrage à froid** (nouveaux utilisateurs ou articles).
- **Renforcer la pertinence** en fusionnant les interactions utilisateurs et la connaissance issue du contenu.

Concrètement, pour combiner filtrage collaboratif et analyse de contenu :

1. **Entraîner deux modèles** : l'un fondé sur l'historique d'interactions (filtrage collaboratif), l'autre sur les caractéristiques textuelles ou les métadonnées (analyse de contenu).
2. **Fusionner les scores** : chaque article obtient un score collaboratif et un score « contenu », qu'on agrège (par exemple, par une moyenne pondérée) pour obtenir un classement final.
3. **Ajuster les pondérations** : renforcer la part du « contenu » quand les données d'interactions sont rares (démarrage à froid), et l'inverse lorsque l'historique d'utilisateur est plus complet.

La mise en place d'un **pipeline de mise à jour automatisé** renforcerait la valeur métier en assurant une **adaptation rapide aux nouveaux contenus** et un **réentraînement régulier du modèle**, pour offrir une plateforme **scalable**, **personnalisée** et prête à **monter en puissance**.

10) Glossaire

ALS (Alternating Least Squares)
Méthode de filtrage collaboratif implicite, utilisée pour générer des recommandations en factorisant la matrice utilisateur-article.

Architecture Azure
Combinaison d'Azure Functions (pour héberger la logique de recommandation), d'Azure Blob Storage (pour stocker CSV et modèles .npz) et d'une application Streamlit (interface utilisateur).

Streamlit
Framework Python pour créer rapidement des applications web interactives et visualiser les résultats de recommandation ou d'analyse de données.

Blob Storage
Service de stockage d'objets dans Azure. Contient ici les données de clics (clicks_sample.csv) et les métadonnées (articles_metadata.csv) et autres fichiers.

.npz
Format de fichier NumPy qui permet de stocker des matrices ou des modèles entraînés (ex. modèle ALS, matrice CSR de popularité) de manière compacte.

Embeddings
Représentations vectorielles des articles, utiles pour mesurer la similarité de contenu ou la proximité sémantique. Les embeddings permettent d'enrichir un système de recommandation par une approche centrée sur le contenu (textes, métadonnées, etc.).

10) Glossaire (suite)

- **MSE (Mean Squared Error)** : Mesure la moyenne des carrés des écarts entre les prédictions et les valeurs réelles. Plus la MSE est faible, plus la prédiction est précise.

- **RMSE (Root Mean Squared Error)** : Racine carrée de la MSE. Elle s'exprime dans la même unité que les valeurs prédites, ce qui la rend plus interprétable que la MSE brute.

- **MAE (Mean Absolute Error)** : Mesure l'erreur moyenne en valeur absolue entre la prédiction et la vérité terrain.

- **Cosine Similarity** : Évalue la similarité entre deux vecteurs (par exemple, deux embeddings d'articles) en mesurant l'angle entre eux.

Toutes ces métriques exploitent, d'une manière ou d'une autre, une **notion de distance ou de différence** dans **un espace de dimensions données** :

- Pour la MSE, la **RMSE** et la **MAE**, on mesure l'écart entre la prédiction (indicateur de popularité prédit) et la réalité (indicateur réel calculé à partir du nombre de clics effectifs).

- Pour la **Cosine Similarity**, on calcule l'angle entre les vecteurs d'embeddings, ce qui revient à évaluer leur proximité dans l'espace vectoriel.