

# Projet 09 :

**Réalisez une application de  
recommandation de contenu**

## Contexte projet :

**My Content** est une start-up qui veut encourager la lecture en recommandant des contenus pertinents pour ses utilisateurs.



**Vous êtes le CTO** et cofondateur de la start-up avec **Samia** qui est CEO. Vous êtes en pleine construction d'un **premier MVP** qui prendra la forme d'une **application**.

Dans un premier temps, votre start-up souhaite tester une solution de recommandation d'articles et de livres à des particuliers.

Un jeu de données est fourni ainsi que deux types d'architectures (type serverless) sont proposées par Julien (développeur web)

### En résumé, votre mission est la suivante :

- développer une première version de votre système de recommandation sous forme d'Azure Functions;
- réaliser une application simple de gestion du système de recommandation (interface d'affichage d'une liste d'id utilisateurs, d'appel Azure functions pour l'id choisi, et d'affichage des 5 articles recommandés)
- stocker les scripts développés dans un dossier GitHub ;
- synthétiser vos premières réflexions sur :
  - l'architecture technique et la description fonctionnelle de votre application à date, et le système de recommandation,
  - l'architecture cible pour pouvoir prendre en compte l'ajout de nouveaux utilisateurs ou de nouveaux articles, que vous présenterez à Samia.

Plan de travail :

1- Description et exploration EDA du jeu de données.

2- Systèmes de recommandation

3- Deploiement du systeme de recommandation

Conclusion



# 1- Description et exploration EDA du jeu de données :

Le lien proposé nous renvoie vers le site Kaggle

54

New Notebook

Download (377 MB)

### News Portal User Interactions by Globo.com

A large dataset for news recommendations offline evaluation and analytics

Data

Code (7)

Discussion (2)

#### About Dataset

Usability

7.94

Les fichiers fournis sont représentés ci dessous :

| Name                       | Date modified      | Type                  | Size       |
|----------------------------|--------------------|-----------------------|------------|
| clicks                     | 4/16/2019 12:26 AM | File folder           |            |
| articles_embeddings.pickle | 4/16/2019 12:25 AM | PICKLE File           | 355,515 KB |
| articles_metadata.csv      | 4/16/2019 12:25 AM | Microsoft Excel Co... | 10,834 KB  |
| clicks_sample.csv          | 4/16/2019 12:25 AM | Microsoft Excel Co... | 132 KB     |

385 fichiers "clicks" au total au niveau du dossier clicks

| Name                | Date modified     | Type                  | Size   |
|---------------------|-------------------|-----------------------|--------|
| clicks_hour_000.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 132 KB |
| clicks_hour_001.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 100 KB |
| clicks_hour_002.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 65 KB  |
| clicks_hour_003.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 45 KB  |
| clicks_hour_004.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 42 KB  |
| clicks_hour_005.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 57 KB  |
| clicks_hour_006.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 139 KB |
| clicks_hour_007.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 241 KB |
| clicks_hour_008.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 342 KB |
| clicks_hour_009.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 391 KB |
| clicks_hour_010.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 373 KB |
| clicks_hour_011.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 341 KB |
| clicks_hour_012.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 400 KB |
| clicks_hour_013.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 366 KB |
| clicks_hour_014.csv | 3/20/2019 2:07 PM | Microsoft Excel Co... | 400 KB |

## 1- Description et exploration EDA du jeu de données :

La description qu'on retrouve sur l'article kaggle est affichée ci dessous :



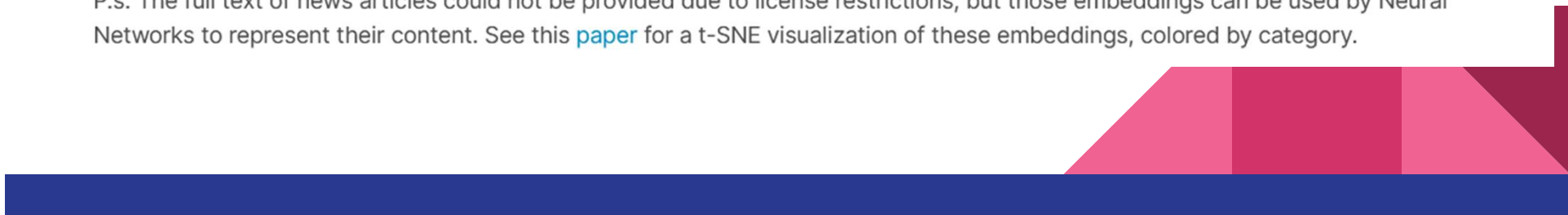
### Content

The dataset contains a sample of user interactions (page views) in [G1 news portal](#) from Oct. 1 to 16, 2017, including about 3 million clicks, distributed in more than 1 million sessions from 314,000 users who read more than 46,000 different news articles during that period.

It is composed by three files/folders:

- **clicks.zip** - Folder with CSV files (one per hour), containing user sessions interactions in the news portal.
- **articles\_metadata.csv** - CSV file with metadata information about all (364047) published articles
- **articles\_embeddings.pickle** Pickle (Python 3) of a NumPy matrix containing the Article Content Embeddings (250-dimensional vectors), trained upon articles' text and metadata by the CHAMELEON's ACR module (see [paper](#) for details) for 364047 published articles.

P.s. The full text of news articles could not be provided due to license restrictions, but those embeddings can be used by Neural Networks to represent their content. See this [paper](#) for a t-SNE visualization of these embeddings, colored by category.



## 1- Description et exploration EDA du jeu de données :

Document : articles\_metadata.csv

```
##-----##
shape of "articles_metadata.csv" file : (364047, 5)
duplicated rows on "article_id" subset : 0
Articles count from min to max: 0 364046
Max and min dates of dataset : 2018-03-13 12:12:30      2006-09-27 11:14:35
Time interval of dataset : 11.47
##-----##
File : Articles metadata.csv
```

| article_id | category_id | created_at_ts   | publisher_id | words_count | creation_time       |
|------------|-------------|-----------------|--------------|-------------|---------------------|
| 0          | 0           | 0 1513144419000 | 0            | 168         | 2017-12-13 05:53:39 |
| 1          | 1           | 1 1405341936000 | 0            | 189         | 2014-07-14 12:45:36 |
| 2          | 2           | 1 1408667706000 | 0            | 250         | 2014-08-22 00:35:06 |

```
check distribution of words count :
```

|  |        |
|--|--------|
| (2016-04-15 00:02:50.833333248, 2018-03-13 12:12:30]           | 243418 |
| (2014-05-18 11:53:11.666666752, 2016-04-15 00:02:50.833333248] | 88834  |
| (2012-06-19 23:43:32.500000, 2014-05-18 11:53:11.666666752]    | 31340  |
| (2010-07-23 11:33:53.333333248, 2012-06-19 23:43:32.500000]    | 352    |
| (2008-08-24 23:24:14.166666752, 2010-07-23 11:33:53.333333248] | 54     |
| (2006-09-23 06:48:07.524999935, 2008-08-24 23:24:14.166666752] | 49     |

```
check distribution of publishers :
```

|   |        |
|---|--------|
| 0 | 364047 |
|---|--------|

```
check distribution of words count :
```

|                              |        |
|------------------------------|--------|
| (-6.691000000000001, 1672.5] | 364022 |
| (1672.5, 3345.0]             | 22     |
| (3345.0, 5017.5]             | 2      |
| (5017.5, 6690.0]             | 1      |

# 1- Description et exploration EDA du jeu de données :

Document : clicks\_sample.csv

|   | user_id | session_id       | session_start | session_size | click_article_id | click_timestamp | click_environment | click_deviceGroup | click_os | click_country | click_region | click_referrer_type |
|---|---------|------------------|---------------|--------------|------------------|-----------------|-------------------|-------------------|----------|---------------|--------------|---------------------|
| 0 | 0       | 1506825423271737 | 1506825423000 | 2            | 157541           | 1506826828020   | 4                 | 3                 | 20       | 1             | 20           | 2                   |
| 1 | 0       | 1506825423271737 | 1506825423000 | 2            | 68866            | 1506826858020   | 4                 | 3                 | 20       | 1             | 20           | 2                   |
| 2 | 1       | 1506825426267738 | 1506825426000 | 2            | 235840           | 1506827017951   | 4                 | 1                 | 17       | 1             | 16           | 2                   |

On procède à la concaténation de l'ensemble des fichiers clicks pour avoir un historique complet

```
shape of global file : (2988181, 13)
```



## 1- Description et exploration EDA du jeu de données :

```
-----
Unique device group : [3 1 4 5 2]
count of "DeviceGroup" : 1 1823162
3 1047086
4 117640
5 283
2 10
Name: click_deviceGroup, dtype: int64
-----
Unique environment : [4 2 1]
count of "Environment" : 4 2904478
2 79743
1 3960
Name: click_environment, dtype: int64
-----
Unique session-size : [ 2  3  7  6  4  5 16  8 10  9 24 11 13 20 12 14 15
18 51 26 22 19 27 23 25 21 31 36 37 30 32 17 35 39 29
28 82 57 41 33 75 53 34 52 38 46 106 74 47 71 58 65 98
72 59 43 60 40 48 86 62 124 92 68 44 45 67 107 94 79 56]
count of "session_size" : (1.877, 26.4] 2975898
(26.4, 50.8] 9358
(50.8, 75.2] 2057
(75.2, 99.6] 531
(99.6, 124.0] 337
Name: session_size, dtype: int64
-----
Unique OS : [20 17  2 12 13 19  5  3]
count of "Os" : 17 1738138
2 788699
20 369586
12 60096
13 23711
19 6384
5 1513
3 54
Name: click_os, dtype: int64
-----
```

```
-----
Unique region : [20 16 24 21 17 25 12 18 19  6  9 13  8  7 28  5  4 26 15 11 10 27  2  1
14 22  3 23]
count of "Regions" : 25 804985
21 464230
13 320957
8 179339
16 164884
28 135793
24 130537
20 120884
5 96979
9 84693
7 64062
17 61514
6 57254
11 35934
19 34092
12 30875
4 30065
15 29535
14 25708
10 21995
26 18893
27 18711
2 16728
18 15083
22 13101
1 7110
3 3997
23 43
Name: click_region, dtype: int64
-----
```

```
-----
Unique country : [ 1 11 10  8  2  4  7  3  9  5  6]
count of "Countries" : 1 2852406
10 61377
11 29999
8 9556
6 7256
9 6746
2 6101
3 4540
5 3498
4 3389
7 3313
Name: click_country, dtype: int64
-----
Unique referrer type : [2 1 5 7 6 4 3]
count of "Referrer types" : 2 1602601
1 1194321
5 80766
7 69798
6 20455
4 19820
3 420
Name: click_referrer_type, dtype: int64
-----
```

Count of consulted articles : 46033 which represents a ratio of : 12.64 %

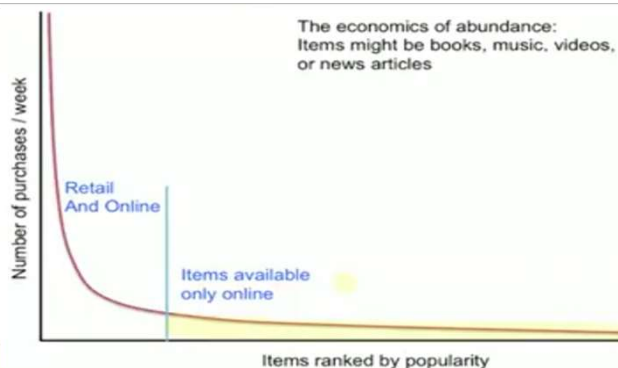


## 2- Systèmes de recommandation :

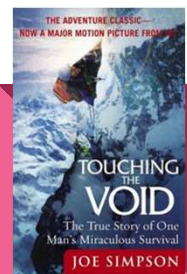
Les **systèmes de recommandation** sont une forme spécifique de filtrage de l'information (SI) visant à présenter les éléments d'information (films, musique, livres, news, images, pages Web, etc) qui sont susceptibles d'intéresser l'utilisateur.

un élément clé de l'utilité de ces systèmes est le fait d'être passé d'une ère de rareté à une ère d'abondance (disponibilité marchande)

Le coût de plus en plus réduit de la mise en ligne d'informations liées à chaque produit , ce qui donne naissance au phénomène nommé "Long-Tail"

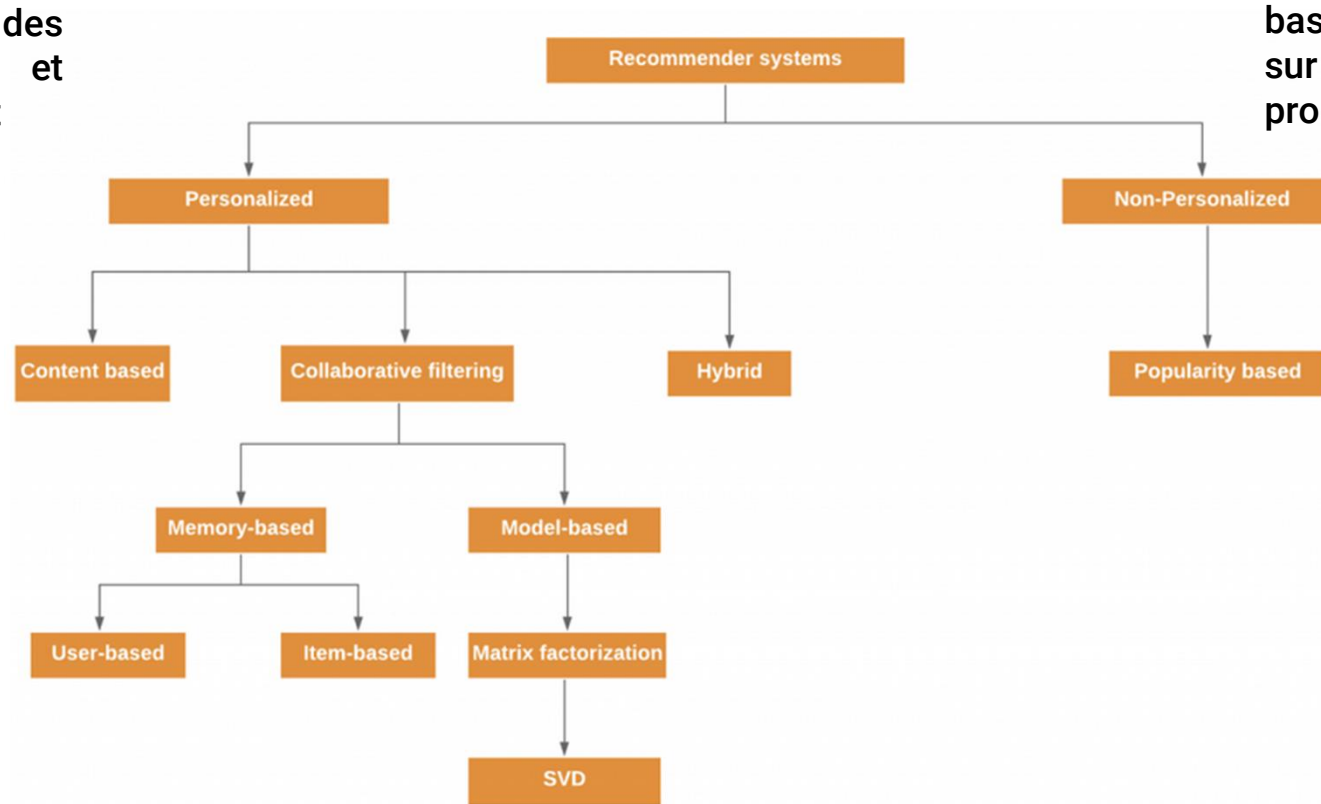


Comment le livre "Into the air" a fait du livre "Touching the void" un bestseller



## 2- Systèmes de recommandation :

Avec analyse des  
items et  
comportement  
des users



basé uniquement  
sur la popularité  
produit

## 2- Systèmes de recommandation :

- Système de popularité :  
c'est un système qui se base essentiellement sur le principe de popularité ou tout autre argument qui constitue une tendance.

On s'est basé dans ce projet sur l'estimation de nombre de click par article pour établir un classement de popularité.

```
# Create popularity model
def get_popularity_rec(clicks, n_reco=5):
    # Compute the five most popular articles
    df_popularity = clicks.groupby(by=['click_article_id'])['click_timestamp'].count().sort_values(ascending=False).reset_index()
    df_popularity.rename(columns = {'user_id':'popularity'}, inplace=True)
    return df_popularity.click_article_id.head(n_reco).to_list()
```

Le résultat reste liée à l'activité des users et changera selon leur utilisation

```
# Call the function to get the most viewed articles according to users activity
# Can be used for new users with no history records on activity
get_popularity_rec(global_df, n_reco=5)
```

```
[160974, 272143, 336221, 234698, 123909]
```

## 2- Systèmes de recommandation :

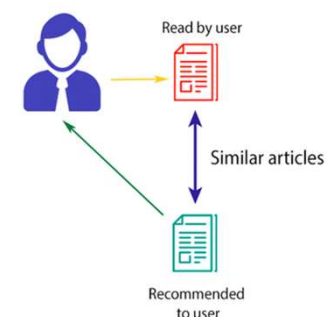
- Content based :

**Cet algorithme analyse un ensemble de contenu sans prendre en compte les utilisateurs** (en tout cas, pas dans un premier temps) **et détecte les similarités entre les contenus à des fins de recommandation en inspectant son contenu.** Pour le content-based, l'analyse de contenu consiste par exemple à **identifier le sujet d'un contenu en répertoriant tous les mots d'un article de presse** (excepté les stop words) puis en comparant tous les mots de l'article analysés aux autres articles. Plus un article aura un nombre de mots similaires, plus ces articles seront considérés comme « proches » permettant ainsi de détecter les sujets identiques ou similaires et d'en déduire des recommandations pour le lecteur.

Pour le projet :

- Une matrice embeddings a été fournie et sera utilisée pour le calcul de similarité entre article. (le processus NLP pour la création de cette matrice n'est pas pris en charge dans ce projet vu que le contenu des articles n'est pas disponible, une mise à jour par contre nécessite a coup sur une revisite complète pour la mise à jour de la matrice Embeddings)
- La fonction cosine\_similarity de la librairie sklearn a été utilisée pour le calcul de similarité.
- le principe du code établi est la création d'une liste des 5 articles similaires a l'article choisi

CONTENT-BASED FILTERING



```
chosen article : 88
```

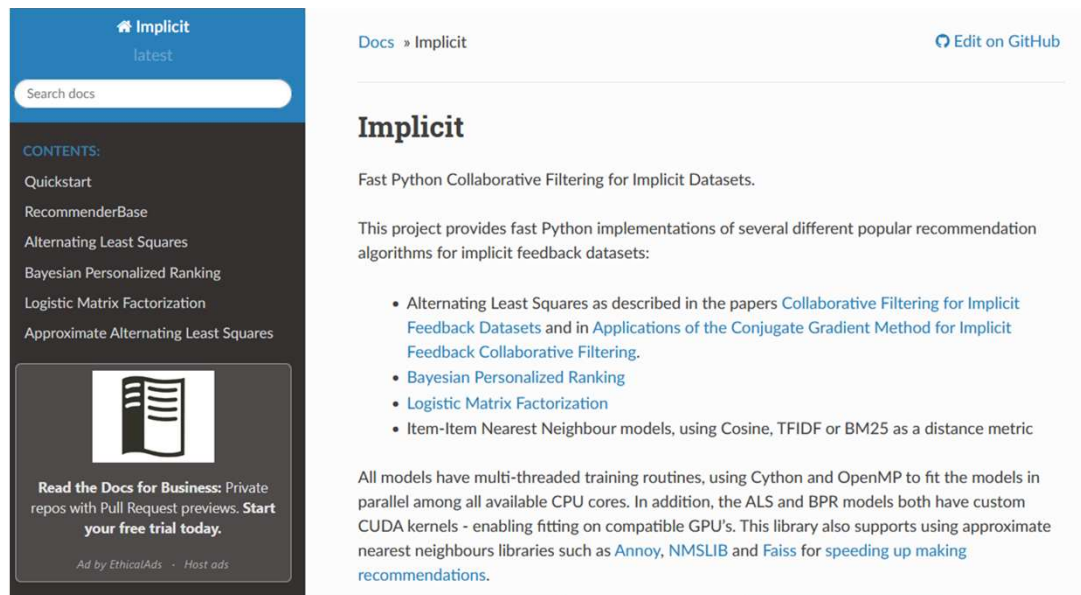
|    | article_index | cosine   |
|----|---------------|----------|
| 79 | 79            | 0.862483 |
| 9  | 9             | 0.832213 |
| 61 | 61            | 0.825840 |
| 76 | 76            | 0.817524 |
| 89 | 90            | 0.807653 |

## 2- Systèmes de recommandation :

### - Collaborative based :

Le filtrage collaboratif repose sur l'adage : Si deux personnes ont aimé des contenus identiques par le passé, elles ont une probabilité élevée d'aimer les mêmes choses dans le futur. Les recommandations personnalisées issues du filtrage collaboratif peuvent être calculées de diverses manières. Notamment en se basant sur **le profil des lecteurs (User-based)**, en utilisant **les profils de contenus (Item-based)** ou encore en faisant de **la factorisation de matrice**.

pour ce projet nous avons eu recours  
à la librairie Implicit



Implicit  
latest

Search docs

CONTENTS:

- Quickstart
- RecommenderBase
- Alternating Least Squares
- Bayesian Personalized Ranking
- Logistic Matrix Factorization
- Approximate Alternating Least Squares

Read the Docs for Business: Private repos with Pull Request previews. **Start your free trial today.**

Ad by EthicalAds · Host ads

Docs » Implicit [Edit on GitHub](#)

## Implicit

Fast Python Collaborative Filtering for Implicit Datasets.

This project provides fast Python implementations of several different popular recommendation algorithms for implicit feedback datasets:

- Alternating Least Squares as described in the papers [Collaborative Filtering for Implicit Feedback Datasets](#) and in [Applications of the Conjugate Gradient Method for Implicit Feedback Collaborative Filtering](#).
- Bayesian Personalized Ranking
- Logistic Matrix Factorization
- Item-Item Nearest Neighbour models, using Cosine, TFIDF or BM25 as a distance metric

All models have multi-threaded training routines, using Cython and OpenMP to fit the models in parallel among all available CPU cores. In addition, the ALS and BPR models both have custom CUDA kernels - enabling fitting on compatible GPU's. This library also supports using approximate nearest neighbours libraries such as [Annoy](#), [NMSLIB](#) and [Faiss](#) for [speeding up making recommendations](#).

## 2- Systèmes de recommandation :

### 1. Préparation des données input et création des ratings :

| user_id | session_id       | click_article_id | click_timestamp |
|---------|------------------|------------------|-----------------|
| 0       | 1506825423271737 | 157541           | 1506826828020   |
| 0       | 1506825423271737 | 68866            | 1506826858020   |

```
def get_ratings(clicks):  
    """ Compute the rating dataframe providing for each interaction a rating based on the number of clicks per article weighted by the total number of clicks per user  
  
    # Create a dataframe containing the number of clicks for each user and each article  
    count_clicks_by_articles_by_user = clicks.groupby(["user_id", "click_article_id"]).agg(count_clicks_by_articles_by_user=("session_id", "count"))  
  
    # Create a dataframe containing the number of clicks for each user  
    count_clicks_by_user = clicks.groupby(["user_id"]).agg(count_clicks_by_user=("session_id", "count"))  
  
    # Compute the weighted ratio of clicks  
    clicks_count = count_clicks_by_articles_by_user.join(count_clicks_by_user, on="user_id")  
    clicks_count['rating'] = clicks_count["count_clicks_by_articles_by_user"] / clicks_count["count_clicks_by_user"]  
  
    # Just rename columns  
    ratings = clicks_count.reset_index().drop(["count_clicks_by_articles_by_user", "count_clicks_by_user"], axis=1).rename(columns={"click_article_id": "article_id"})  
  
    return ratings
```

matrice de Ratings

|         | user_id | article_id | rating |
|---------|---------|------------|--------|
| 0       | 0       | 68866      | 0.125  |
| 1       | 0       | 87205      | 0.125  |
| 2       | 0       | 87224      | 0.125  |
| 3       | 0       | 96755      | 0.125  |
| 4       | 0       | 157541     | 0.125  |
| ...     | ...     | ...        | ...    |
| 2950705 | 322894  | 168401     | 0.500  |
| 2950706 | 322895  | 63746      | 0.500  |
| 2950707 | 322895  | 289197     | 0.500  |
| 2950708 | 322896  | 30760      | 0.500  |
| 2950709 | 322896  | 157507     | 0.500  |

## 2- Systèmes de recommandation :

### 2. Création de la sparse matrice et évaluation de divers modèles (Implicit) :

```
# Define models
models_list = [AlternatingLeastSquares(), BayesianPersonalizedRanking(), LogisticMatrixFactorization()]
```

```
train_csr = csr_matrix((train_df['rating'], (train_df['user_id'], train_df['article_id'])), dim)
test_csr = csr_matrix((test_df['rating'], (test_df['user_id'], test_df['article_id'])), dim)
```

Dataframe complet

|   | model                       | Precision@k | MAP@k   | nDCG@k  | train_time |
|---|-----------------------------|-------------|---------|---------|------------|
| 0 | AlternatingLeastSquares     | 0.10099     | 0.06523 | 0.09320 | 309.36034  |
| 1 | BayesianPersonalizedRanking | 0.12337     | 0.08388 | 0.11816 | 75.19970   |
| 2 | LogisticMatrixFactorization | 0.03130     | 0.01085 | 0.01985 | 52.10852   |

Dataframe ratio = 5% soit 147536 lignes

|   | model                       | Precision@k | MAP@k   | nDCG@k  | train_time |
|---|-----------------------------|-------------|---------|---------|------------|
| 0 | AlternatingLeastSquares     | 0.01732     | 0.00594 | 0.00874 | 64.59176   |
| 1 | BayesianPersonalizedRanking | 0.00288     | 0.00125 | 0.00165 | 5.96202    |
| 2 | LogisticMatrixFactorization | 0.06721     | 0.02346 | 0.03473 | 12.17533   |



## 2- Systèmes de recommandation :

### 3. Création de la fonction de prédiction :

```
def get_cf_reco(clicks, userID, csr_item_user, csr_user_item, model_path=None, n_reco=5, train=True):  
    start = time()  
    # Train the model on sparse matrix of shape (number_items, number_user)  
  
    if train or model_path is None:  
        model = LogisticMatrixFactorization(factors= 128, random_state=42)  
        print("[INFO] : Start training model")  
        model.fit(csr_user_item)  
  
        # Save model to disk  
        with open('recommender.model', 'wb') as filehandle:  
            pickle.dump(model, filehandle)  
    else:  
        with open('recommender.model', 'rb') as filehandle:  
            model = pickle.load(filehandle)  
  
    # Recommend N best items from sparse matrix of shape (number_user, number_items)  
    # Implicit built-in method  
    # N (int) : number of results to return  
    # filter_already_liked_items (bool) : if true, don't return items present in  
    # the training set that were rated/viewed by the specified user  
    recommendations_list = []  
    recommendations = model.recommend(userID, csr_user_item[userID], N=n_reco, filter_already_liked_items=True)  
  
    print(f'[INFO] : Completed in {round(time() - start, 2)}s')  
  
    #recommendations = [elt[0] for elt in recommendations]  
  
    return recommendations
```

Recommandations après  
input de données :

```
# Call the function, load the model and perform recommendations  
userID = 34  
get_cf_reco(df, userID, csr_item_user, csr_user_item, model_path="./recommender.model", n_reco=5, train=False)  
  
[INFO] : Completed in 0.35s  
  
(array([354086, 158535, 348111, 30730, 199376]),
```



### 3- Déploiement du système de recommandation :

Le modèle collaboratif a été choisi (vu sa pertinence) pour être déployé sur le cloud en utilisant aussi une Azure fonction.

Azure Functions est un service cloud disponible à la demande qui fournit l'infrastructure et les ressources mises à jour en continu qui sont nécessaires pour exécuter vos applications. Vous vous concentrez sur le code le plus important pour vous, dans le langage le plus productif pour vous, et Functions gère le reste. Functions assure un calcul serverless pour Azure. Vous pouvez utiliser Functions pour créer des API web, répondre à des modifications de base de données, traiter des flux IoT, gérer des files d'attente de messages et plus encore

#### Déclencheur HTTP Azure Functions

Article • 18/11/2022 • 27 minutes de lecture • 14 contributeurs

[Commentaires](#)

Choisir un langage de programmation

C# Java JavaScript PowerShell **Python**

Le déclencheur HTTP vous permet d'appeler une fonction avec une requête HTTP. Vous pouvez utiliser un déclencheur HTTP pour générer des API serverless et répondre aux webhooks.

La valeur de retour par défaut pour une fonction déclenchée par HTTP est :

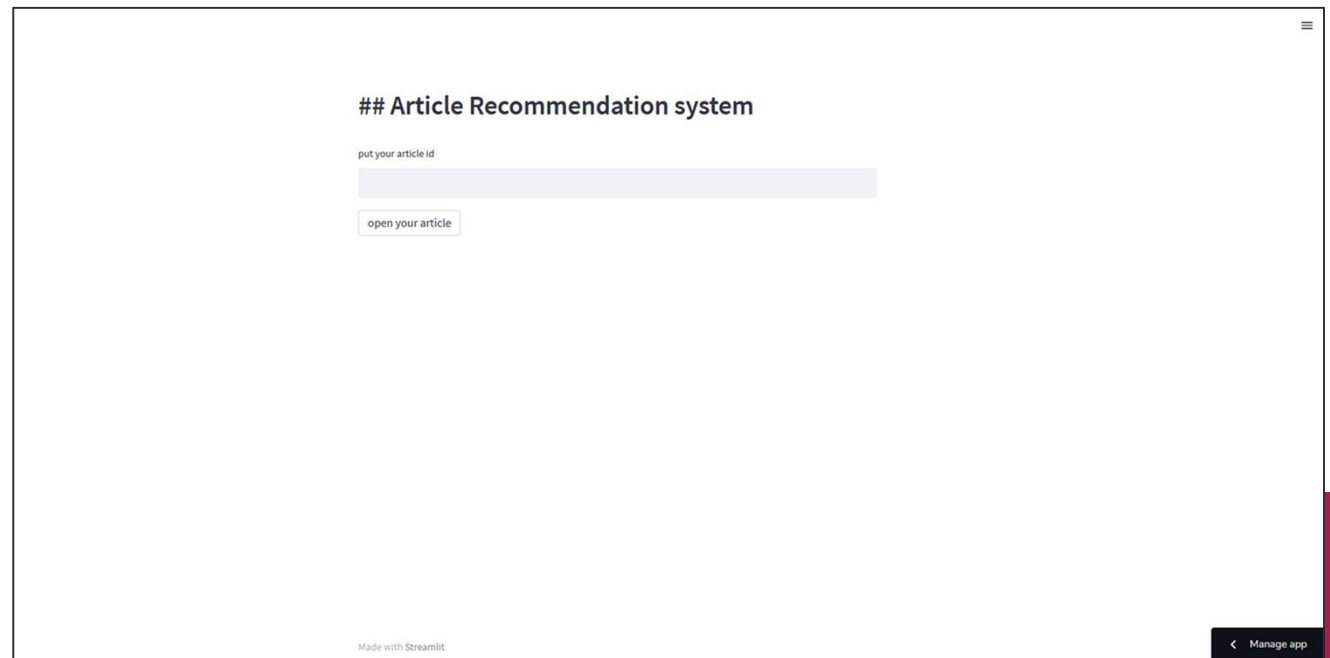
- `HTTP 204 No Content` avec un corps vide dans Functions 2.x et versions ultérieures
- `HTTP 200 OK` avec un corps vide dans Functions 1.x

### 3- Déploiement du système de recommandation :

Le modèle collaboratif a été choisi (vu sa pertinence) pour être déployé sur le cloud en utilisant aussi une Azure fonction.

<https://cher-if-p9-app-p9-dep-ue5ria.streamlit.app/>

L'image ci contre  
montre l'interface  
utilisateur de notre  
application déployée  
sur le cloud



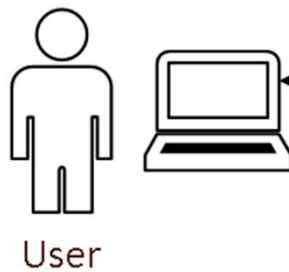
### 3- Déploiement du système de recommandation :



### 3- Déploiement du système de recommandation :

Architecture actuelle

Deployed on streamlitcloud



HTTPS



Microsoft  
Azure



Function Apps

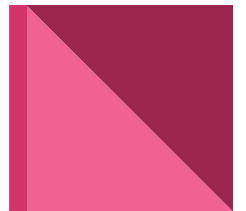


Visual Studio Code


API : Deployed on Pyhtonanywhere


HTTP  
Request

HTTP  
Response




### 3- Déploiement du système de recommandation : API sur pythonanywhere

 pythonanywhere  
by ANACONDA

/home/Ai2020/  mysite




Directories

[New directory](#)

[\\_\\_pycache\\_\\_](#) 


Files

[New file](#)

 flask\_app.py   2022-11-25 17:43 3.6 KB

[Upload a file](#)

100MiB maximum size

 /home/Ai2020/mysite/flask\_app.py

Keyboard shortcuts: [Normal v](#) [Share](#) [M Save](#) [Save as...](#) [Run](#) [C](#)

```
1 |
2 # Start importing relevant libraries
3 # Import libraries
4 import os
5 import pandas as pd
6 import numpy as np
7 from time import time
8 from random import randint
9 from sklearn.metrics.pairwise import cosine_similarity
10
11 from scipy.sparse import csr_matrix
12 from sklearn.model_selection import train_test_split
13
14 from tqdm import tqdm
15
16 from implicit.ief import LogisticMatrixFactorization
17 from implicit.evaluation import precision_at_k, mean_average_precision_at_k, ndcg_at_k, AUC_at_k
18 import pickle
19 import flask
20 from flask import jsonify
21
22
23 clicks = pd.read_csv("https://github.com/archiducamel/p9-oc/releases/download/clicks/clicks.csv")
24 MODEL_PATH = "../recommender.model"
25 if not os.path.exists(MODEL_PATH):
26     os.system("wget https://github.com/archiducamel/p9-oc/releases/download/clicks/recommender.model")
27
28 def compute_interaction_matrix(clicks):
29     # Create interaction DF (count of interactions between users and articles)
30     interactions = clicks.groupby(['user_id', 'article_id']).size().reset_index(name='count')
31     # Print(interactions DF shape, interactions shape)
32
33     # csr = compressed sparse row (good format for math operations with row slicing)
34     # Create sparse matrix of shape (number_items, number_user)
35     csr_item_user = csr_matrix((interactions['count'].astype(float),
36                               (interactions['article_id'],
37                                interactions['user_id'])))
38     # Print('CSR Shape (number_items, number_user): ', csr_item_user.shape)
39
40     # Create sparse matrix of shape (number_user, number_items)
41     csr_user_item = csr_matrix((interactions['count'].astype(float),
42                               (interactions['user_id'],
43                                interactions['article_id'])))
43     # Print('CSR Shape (number_user, number_items): ', csr_user_item.shape)
44
45
```

### 3- Déploiement du système de recommandation : Activité GitHub

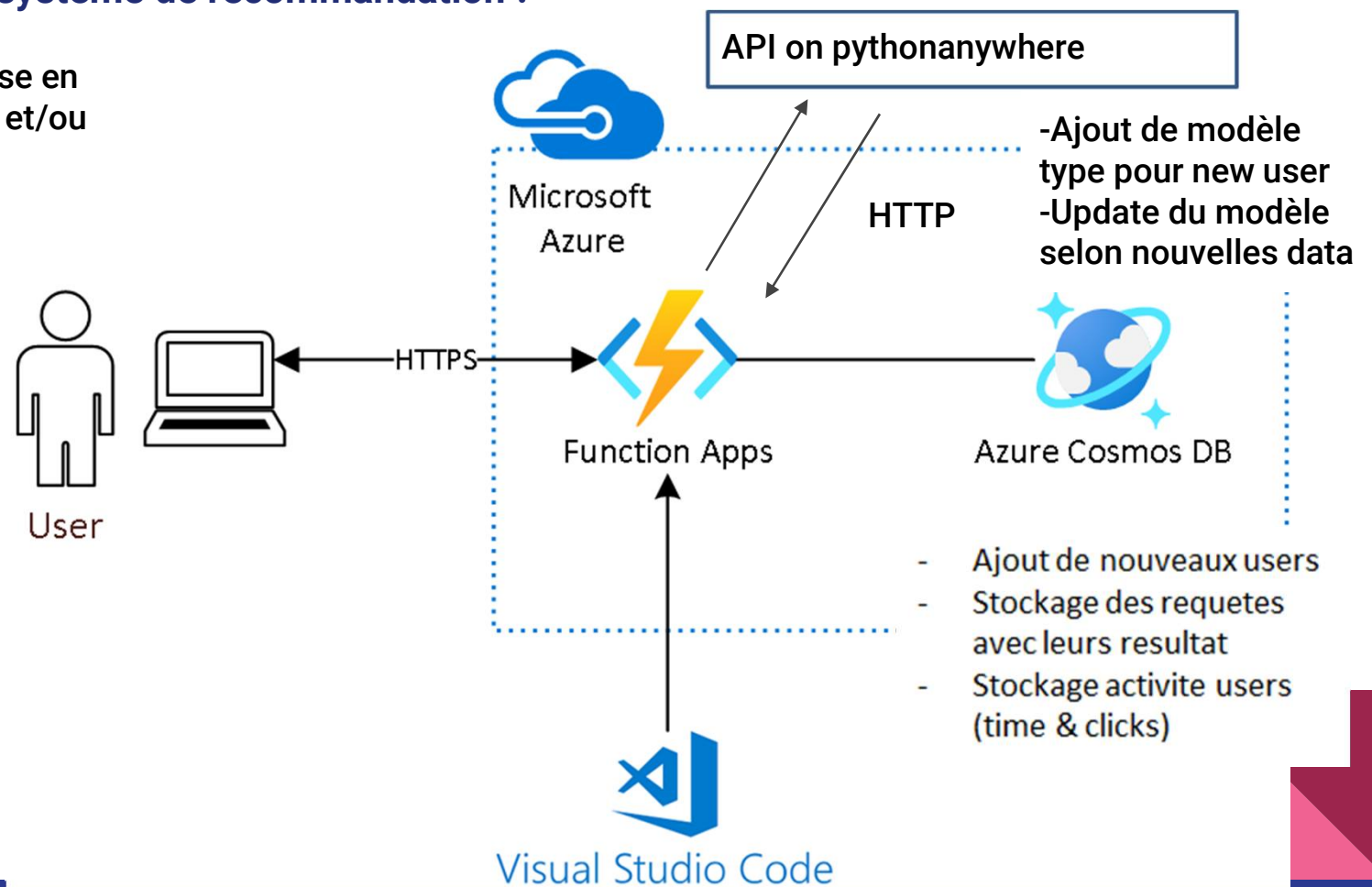
[https://github.com/Cher-iF/P9\\_app.git](https://github.com/Cher-iF/P9_app.git)

The screenshot shows the GitHub profile page for user 'Cher-iF'. The top navigation bar includes 'Overview' (selected), 'Repositories' (3), and 'Projects'. Under 'Popular repositories', three repositories are listed: 'P7\_app' (sentiment analysis, Python), 'P8\_app' (Python), and 'P9\_app' (Jupyter Notebook). Below this, a '55 contributions in the last year' section shows a calendar grid with green squares indicating contributions, primarily in September and November. A legend at the bottom indicates the intensity of contributions with a color scale from light green to dark green.

The screenshot shows the repository page for 'Cher-iF / P9\_app'. The top navigation bar includes 'Code' (selected), 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. Below the navigation bar, the repository name 'Cher-iF / P9\_app' is displayed with a 'Public' label. The main content area shows a list of files and folders: 'EDA', '1.PNG', 'p9\_dep.py', and 'requirements.txt'. Each item has a commit message and a timestamp. For example, 'EDA' was added 2 minutes ago, while '1.PNG', 'p9\_dep.py', and 'requirements.txt' were first committed 4 days ago. A green 'Add a README' button is visible at the bottom right.

### 3- Déploiement du système de recommandation :

Architecture pour prise en charge de new users et/ou articles



## Conclusion :

1- Le projet procure une introduction intéressante pour les systèmes de recommandation. Il serait intéressant d'exploiter d'autres dataset ainsi que d'autres librairies dédiées à ce topic.

2- L'option azure fonction permet d'utiliser une architecture serverless pour de tels projets, et le déploiement est facile sur Microsoft Azure

