



الْمَهْدِيَّ الْوُطْنِيُّ لِلْبَرِيدِ وَالْمَوَاسِلِ  
مَهْدِيَّةُ الْمَوَاسِلِ وَالْبَرِيدِ  
Institut National des Postes et Télécommunications



agence nationale de réglementation  
des télécommunications  
الوكالة الوطنية لتقنين المواصلات

FILIÈRE DATA ENGINEERING

PROJET (RECHERCHE & DÉVELOPPEMENT)

## Rapport du Projet

---

# La Détection des communautés virtuelles

---

*Préparé par :*

NOUALI TAHA  
EL MAHFOUD RADOUANE  
MESBAH ABDERAHMANE  
SAOUD YASSINE

*Encadré par :*

Mme AYACHE Meryeme  
Mme BELLA Imane

Année Universitaire 2022/2023

# DÉDICACE

“

*Nous dédions ce modeste travail,*

*À nos chers parents qui ont tant donné,*

*Pour leur immense soutien, leur grand amour, leurs sacrifices et  
leurs prières.*

*Qu'ils acceptent ici l'hommage de notre gratitude, qui, quelque  
grande qu'elle soit, ne sera jamais à la hauteur de leur tendresse et  
leur dévouement.*

*À notre encadrante,*

***Mme AYACHE Meryeme et Mme BELLA Imane***

*Vous avez toujours cru en nous, et c'est grâce à votre soutien que  
nous trouvons la volonté de continuer. Votre présence est une source  
d'inspiration pour nous.*

*À toutes nos familles.*

*À tous nos chers condisciples.*

*À toutes nos enseignantes et à tous nos enseignants.*

*À tous ceux que nous aimons.*

*À tous ceux qui nous aiment.*

*Et À tous ceux qui nous ont aidé de près ou de loin...*

”

# **REMERCIEMENTS**

Tout d'abord, nous remercions le Seigneur tout puissant de nous avoir donné le courage et la patience nécessaires pour mener ce travail à son terme.

Nous aimerais tout d'abord remercier nos très affectueuses encadrantes, Mme AYACHE Meryeme et Mme BELLA Imane, de nous avoir appris à être des plus autonomes tout au long de notre formation. Leur qualité de suivi, leurs avis éclairés et les judicieux conseils qu'elles nous ont prodigués ont été on ne peut plus précieux.

Nous tenons aussi à remercier tout particulièrement notre superviseur académique et notre chef de filière Mr. BAINA Amine, pour l'aide compétente qu'il nous a apportée, pour son temps, son soutien et son assistance, ses informations pratiques.

Nous savons aussi gré à l'équipe pédagogique et administrative de l'institut, et au corps professoral pour leurs efforts en vue de nous offrir une formation d'excellence.

Pour en finir, nous souhaitons remercier toute personne ayant contribué, un tant soit peu, de près ou de loin à la réalisation de ce travail.

# RÉSUMÉ

Synergeon travaille sur un projet en cours, intitulé "WellChain : the future-proof wellness enabler platform" ("WellChain : la plateforme garantissant le bien-être durable"). Ce projet est dédié au développement de solutions dans le domaine de la biotechnologie, en combinant spécifiquement les domaines de l'épigénétique et des "digital twins" (jumeaux numériques). L'objectif ultime de ce projet est d'inverser l'âge chronologique d'une personne, lui permettant de gagner au moins 20 ans d'âge biologique. Pour y parvenir, le projet adopte une approche axée sur la santé en traquant tout d'abord les pathologies courantes qui accélèrent le vieillissement cellulaire chez les individus, souvent à leur insu, comme les déséquilibres métaboliques et les maladies cardiovasculaires.

Ils visent à aborder les problématiques complexes liées au vieillissement et au bien-être, en tirant parti de la puissance de la technologie et de la créativité.

Ce rapport présente un projet visant à soutenir et développer une excellente expérience de fitness et de bien-être en créant des communautés de fitness en ligne. L'objectif principal est de regrouper les utilisateurs ayant des intérêts et des objectifs similaires afin de favoriser l'entraide, la motivation et le partage des connaissances. Le projet utilise des techniques de scraping pour collecter des données pertinentes à partir de différentes plateformes de médias sociaux. Ensuite, les données sont prétraitées pour être utilisées dans des modèles d'Intelligence Artificielle (IA) tels que le clustering. Le rapport détaille le workflow du projet, les étapes de scraping, de prétraitement et de modélisation, ainsi que les exemples d'algorithmes d'IA utilisés pour regrouper les utilisateurs.

Ce rapport donne un aperçu du projet, y compris ses objectifs et son flux de travail. Il décrit le processus de collecte de données à partir de différentes plateformes de médias sociaux, les étapes de prétraitement nécessaires et la mise en œuvre de modèles d'IA tels que le regroupement des utilisateurs. Le rapport conclut en mettant en avant les avantages des communautés de fitness en ligne, en soulignant l'importance de la motivation, du soutien et du partage des connaissances pour atteindre un bien-être optimal. Il discute également des résultats obtenus grâce à l'approche de regroupement des utilisateurs et explore les possibilités d'améliorations et d'avancées futures.

---

**Mots clés :** Communautés, Collecte des données, Algorithme de clustering, Automatisation.

# ABSTRACT

Synergeon is working on an ongoing project, titled "WellChain : the future-proof wellness enabler platform". This project is dedicated to the development of solutions in the field of biotechnology, specifically combining the fields of epigenetics and "digital twins". The ultimate goal of this project is to reverse a person's chronological age, allowing them to gain at least 20 years of biological age. To achieve this, the project takes a health-focused approach by first tracking common pathologies that accelerate cellular aging in individuals, often without their knowledge, such as metabolic imbalances and cardiovascular disease.

They aim to address complex issues related to aging and well-being, leveraging the power of technology and creativity.

This report features a project to support and grow a great fitness and wellness experience by creating online fitness communities. The main objective is to bring together users with similar interests and objectives in order to promote mutual aid, motivation and knowledge sharing. The project uses scraping techniques to collect relevant data from different social media platforms. Then the data is preprocessed for use in Artificial Intelligence (AI) models such as clustering. The report details the project workflow, scraping, pre-processing and modeling steps, as well as examples of AI algorithms used to group users.

This report provides an overview of the project, including its goals and workflow. It describes the process of collecting data from different social media platforms, the necessary pre-processing steps, and the implementation of AI models such as user clustering. The report concludes by highlighting the benefits of online fitness communities, highlighting the importance of motivation, support and knowledge sharing to achieve optimal well-being. It also discusses the results achieved through the user grouping approach and explores opportunities for future improvements and advancements.

---

**Keywords :** Communities, Data collection, Clustering algorithms, Automation.

---

# Table des matières

DÉDICACE	1
REMERCIEMENTS	2
RÉSUMÉ	3
ABSTRACT	4
TABLE DES FIGURES	8
LISTE DES ACRONYMES	9
INTRODUCTION GÉNÉRALE	10
<b>1 CONTEXTE GÉNÉRAL</b>	<b>11</b>
1.1 Organisme d'accueil . . . . .	12
1.1.1 Synergeon . . . . .	12
1.1.2 Soft Centre . . . . .	12
1.2 Contexte du projet . . . . .	13
1.2.1 Motivation et problématique du projet . . . . .	13
1.2.2 Objectifs du projet . . . . .	13
1.3 Conduite du projet . . . . .	14
1.3.1 Processus de développement adopté . . . . .	14
1.3.2 Présentation de la méthode Scrum . . . . .	14
1.3.3 Planification du projet . . . . .	14
1.3.4 Outils de suivi du projet . . . . .	15
<b>2 COLLECTION DES DONNÉES</b>	<b>16</b>
2.1 Collection des données à partir de Reddit : . . . . .	18
2.1.1 Pourquoi Reddit ? . . . . .	18
2.1.2 Explication du code : . . . . .	18
2.2 Collection des données à partir d'Instagram : . . . . .	21
2.2.1 Pourquoi Instagram ? . . . . .	21
2.2.2 Explication du code : . . . . .	21
2.3 Collection des données à partir de YouTube : . . . . .	25
2.3.1 Pourquoi YouTube ? . . . . .	25
2.3.2 Explication du code : . . . . .	25

<b>3 Preprocessing</b>	<b>30</b>
3.1 Importation des bibliothèques . . . . .	31
3.2 Prétraitement . . . . .	31
3.3 NLTK . . . . .	34
3.4 WordCloud . . . . .	36
<b>4 CRÉATION DU MODÈLE</b>	<b>39</b>
4.1 Étude benchmarking : . . . . .	40
4.1.1 Choix du modèle : . . . . .	40
4.1.1.1 Les modèles de clustering : . . . . .	40
4.1.1.2 Autres solutions que le clustering : . . . . .	41
4.1.2 Représentation des données : . . . . .	41
4.1.3 Les paramètres d'évaluation : . . . . .	42
4.2 Explication du code : . . . . .	42
4.2.1 Utilisation du modèle BERT : . . . . .	43
4.2.2 Utilisation de la matrice TF-IDF : . . . . .	46
4.2.3 Modèle finale : . . . . .	47
<b>5 communautés externes et déploiement</b>	<b>51</b>
5.1 Détection des communautés externes . . . . .	52
5.1.1 Définition et concepts . . . . .	52
5.1.2 Explication du code : . . . . .	52
5.1.2.1 Intégration du chatgpt : . . . . .	52
5.1.2.2 Extraction des liens à partir du texte : . . . . .	53
5.2 Déploiement avec Flask . . . . .	54
5.2.1 Outils et Technologies . . . . .	54
5.2.2 Implémentation du code . . . . .	56
5.2.2.1 Configuration de l'environnement virtuel . . . . .	56
5.2.2.2 Écrire l'application avec flask . . . . .	58
5.2.2.3 création des pages web . . . . .	63
5.2.2.4 Démarrage du serveur . . . . .	65
<b>CONCLUSION</b>	<b>68</b>

# Table des figures

1.1	Synergeon . . . . .	12
1.2	Softcenter . . . . .	13
2.1	Le portail du compte développeur Reddit . . . . .	18
2.2	Téléchargement de la bibliothèque PRAW . . . . .	18
2.3	Configurer l'accès à l'API de Reddit . . . . .	19
2.4	Liste des mots-clés . . . . .	19
2.5	Le code de l'extraction, la manipulation et la sauvegarde des données . . . . .	19
2.6	Instagram Scraper(1) . . . . .	21
2.7	Instagram Scraper(2) . . . . .	22
2.8	Instagram Scraper(3) . . . . .	22
2.9	Instagram Scraper(4) . . . . .	23
2.10	Instagram Scraper(5) . . . . .	24
2.11	Instagram Scraper(6) . . . . .	24
2.12	bibliothèques nécessaires pour Youtube scraping . . . . .	25
2.13	Changement de la page youtube . . . . .	26
2.14	la fonction scroll . . . . .	26
2.15	création de la liste des liens et de la dataframe finale . . . . .	27
2.16	code de scraping . . . . .	27
2.17	youtube dataframe . . . . .	28
2.18	exemples des données collectés . . . . .	28
2.19	data-set finale . . . . .	29
3.1	Bibliothèques importés . . . . .	31
3.2	Changement des données extraites . . . . .	31
3.3	Suppression des commentaires redondants, remplissages des cases vides et transformation en minuscules . . . . .	31
3.4	Nombre des cases vides . . . . .	32
3.5	les dimensions actuelles de la dataset . . . . .	32
3.6	Liste des mots clés et le code de filtre . . . . .	32
3.7	Filtre des mots indésirables et le filtre de la longueur . . . . .	33
3.8	Code de l'exécution des filtres . . . . .	33
3.9	La fonction de suppression des émojis et des caractères spéciaux . . . . .	33
3.10	la fonction pour la suppression des commentaires non anglais . . . . .	34
3.11	Importation des bibliothèques et modules . . . . .	34
3.12	La fonction preprocess_text() . . . . .	35
3.13	Le code de l'implémentation de la fonction précédente sur les commentaires . . . . .	36
3.14	Code de la jointure des commentaires selon l'auteur . . . . .	36
3.15	Code de génération du nuage de mots . . . . .	36
3.16	WordCloud généré à partir des commentaires prétraité . . . . .	37
3.17	Les dimensions des données prétraités . . . . .	38

3.18 Sauvegarde de la résultat sous le format csv . . . . .	38
4.1 nested clusters . . . . .	40
4.2 bibliothèques de l'ML . . . . .	43
4.3 bert encoding . . . . .	43
4.4 comments encoding using bert model . . . . .	44
4.5 KMeans model using Bert encoding . . . . .	45
4.6 Result for the first scenario . . . . .	45
4.7 la matrice TF-IDF . . . . .	46
4.8 Modèle DBSCAN . . . . .	46
4.9 DBSCAN results . . . . .	47
4.10 Modèle KMeans . . . . .	47
4.11 Clusters topics . . . . .	48
4.12 Exemple 1 . . . . .	48
4.13 Exemple 2 . . . . .	49
4.14 Exemple 3 . . . . .	49
4.15 Exemple 4 . . . . .	49
4.16 Exemple 5 . . . . .	49
5.1 chatgpt(1) . . . . .	52
5.2 image . . . . .	53
5.3 chatgpt(2) . . . . .	53
5.4 chatgpt(3) . . . . .	53
5.5 links extraction . . . . .	54
5.6 FLASK . . . . .	54
5.7 HTML . . . . .	55
5.8 CSS . . . . .	56
5.9 create virtual environment . . . . .	56
5.10 activation . . . . .	57
5.11 libraries . . . . .	57
5.12 ENV variables . . . . .	57
5.13 application.py . . . . .	58
5.14 process function . . . . .	59
5.15 predict function . . . . .	60
5.16 returnTopics function . . . . .	60
5.17 chatWithGpt function . . . . .	61
5.18 extractLinesWithNumbers function . . . . .	61
5.19 returnTopics function . . . . .	62
5.20 returnComments function . . . . .	62
5.21 home.html . . . . .	63
5.22 results.html . . . . .	64
5.23 running server . . . . .	65
5.24 page(1) . . . . .	65
5.25 page(2) . . . . .	66
5.26 page(3) . . . . .	66

# LISTE DES ACRONYMES

<b>API</b>	<i>Application Programming Interface</i>
<b>HTTP</b>	<i>Hypertext Transfer Protocol</i>
<b>IP</b>	<i>Internet Protocol</i>
<b>IT</b>	<i>Information Technology</i>
<b>CSV</b>	<i>Comma-separated values</i>
<b>JSON</b>	<i>JavaScript Object Notation</i>
<b>SI</b>	<i>Système d'information</i>
<b>praw</b>	<i>Python Reddit API Wrapper</i>
<b>re</b>	<i>regular expression</i>
<b>XML</b>	<i>Extensible Markup Language</i>

# INTRODUCTION GÉNÉRALE

Notre projet se concentre sur la création de communautés de fitness en ligne pour soutenir et développer une excellente expérience de bien-être. Les communautés en ligne offrent un environnement propice où les membres peuvent partager leurs connaissances, se connecter avec d'autres personnes partageant les mêmes idées et travailler ensemble pour atteindre leurs objectifs de fitness. Des marques renommées telles que Strava, Fitbit et MyFitnessPal sont des exemples réussis de telles communautés en ligne.

L'objectif de notre projet est de rassembler des personnes de tous niveaux au sein de communautés spécifiques. Ces communautés permettront aux membres de partager des connaissances, de réseauter avec leurs pairs et de s'engager dans un environnement d'apprentissage, de défis et de motivation pour améliorer leur bien-être global. Pour faciliter cela, nous avons recours à l'intelligence artificielle (IA) pour analyser les données des utilisateurs et les intégrer dans des communautés adaptées à leurs profils, objectifs, niveaux d'expérience, types d'entraînement, coachs et activités.

Il existe plusieurs méthodes d'utilisation de l'IA pour regrouper les utilisateurs au sein de communautés. Nous utilisons des techniques telles que le clustering, l'analyse de similarité et les réseaux de neurones pour créer des groupes basés sur des caractéristiques similaires des utilisateurs, telles que leurs objectifs de fitness, leur niveau d'expérience, leur profil d'entraînement, etc.

Pour réaliser ce projet, nous avons commencé par extraire des données à partir de différentes plateformes de médias sociaux telles que YouTube, Reddit et Instagram en effectuant du scraping. Nous avons ciblé des sujets pertinents tels que la nutrition, l'obésité, le diabète, le jeûne intermittent, l'exercice physique, etc. Ensuite, nous avons prétraité ces données pour les utiliser dans notre algorithme de clustering basé sur le k-means.

Ce rapport est structuré en plusieurs sections pour fournir une vue d'ensemble complète de notre projet. Dans la section suivante, nous présenterons le flux de travail général du projet. Ensuite, nous aborderons plus en détail les étapes de scraping des données, de prétraitement et de clustering. Chaque section fournira des informations détaillées sur les méthodes utilisées et les résultats obtenus.

l'objectif ultime est de développer une plateforme en ligne qui rassemble les individus partageant les mêmes intérêts et objectifs en matière de bien-être, favorisant ainsi un soutien mutuel et une expérience de fitness optimale. Nous croyons fermement que les communautés en ligne jouent un rôle essentiel dans l'amélioration de notre santé et de notre bien-être, et nous sommes impatients de présenter les résultats de notre projet.

# Chapitre 1

## CONTEXTE GÉNÉRAL

Dans ce premier chapitre, nous commencerons par présenter l'entreprise dans laquelle le projet s'est déroulé, en détaillant son contexte ainsi que ses objectifs. Nous présenterons ensuite le processus de développement que nous avons suivi afin de réaliser ce projet.

### Table des matières

---

1.1	Organisme d'accueil . . . . .	12
1.1.1	Synergeon . . . . .	12
1.1.2	Soft Centre . . . . .	12
1.2	Contexte du projet . . . . .	13
1.2.1	Motivation et problématique du projet . . . . .	13
1.2.2	Objectifs du projet . . . . .	13
1.3	Conduite du projet . . . . .	14
1.3.1	Processus de développement adopté . . . . .	14
1.3.2	Présentation de la méthode Scrum . . . . .	14
1.3.3	Planification du projet . . . . .	14
1.3.4	Outils de suivi du projet . . . . .	15

---

## 1.1 Organisme d'accueil

### 1.1.1 Synergeon

Synergeon est une startup spécialisée dans la conception et le développement de solutions disruptives, dans les domaines de la robotique, du software et de l'internet des objets ; avec une prédisposition particulièrement portée sur les segments technologiques que sont l'intelligence artificielle et la blockchain (nft). Sa mission statement repose principalement sur le développement et la mise en orbite de projets innovants qualifiés de “Moonshot projects” ; à savoir des projets initialement exploratoires visant un impact massif pour la société (tant sur le plan économique que social) et donc capables d'aborder un méta problème, via une approche créative radicale, en employant des technologies de pointe. Le présent challenge s'inscrit dans le cadre d'un projet technologique actuellement mis en œuvre par Synergeon, sous la dénomination “WellChain : the future proof wellness enabler platform” ; qui se veut être dédié au développement de solutions dans le domaine de la biotech, notamment à travers une approche combinatoire mixant l'épigénétique et les technologies dites “digital twins”

L'objectif ultime de ce projet consiste à renverser l'âge chronologique d'une personne pour lui permettre de gagner au moins 20 ans d'âge biologique. Pour ce faire, l'approche suivie du point de vue de la santé consiste à tracker, dans un premier temps, les types de pathologies sources qui touchent la plupart des personnes et qui contribuent à accélérer le vieillissement cellulaire, et le plus souvent sans qu'elles le sachent (déséquilibres métaboliques, pathologies cardio-vasculaires, ....). Pour ce faire, dans le cadre de son développement, Synergeon cherche à identifier ,via la présente démarche d'Open Innovation, ses futur(e)s collaborateurs(trices) qui pourraient disposer du “mind set” requis pour adresser ces enjeux à venir.



FIG. 1.1 : Synergeon

### 1.1.2 Soft Centre

Le Soft Centre, dont la Présidence est assurée par l'ANRT ([www.anrt.ma](http://www.anrt.ma)), est un Centre d'innovation et de RD logiciel mis à disposition des opérateurs du secteur de l'Industrie des Technologies de l'Information ; dans le but de leur permettre de produire du logiciel innovant en faisant appel aux compétences universitaires de recherche appliquée au sein des Universités et Ecoles d'ingénieurs. Le Business Model du Soft Centre repose à ce jour sur le développement de 3 domaines d'intervention : • La recherche appliquée et le développement logiciel ; à savoir la génération de projets de recherche et développement logiciel “à la demande”. • Le Skill Center for Mobile Applications : centre de services partagés, via la mise à disposition de ressources mutualisées au profit des opérateurs du secteur des TI, dans le but de favoriser l'essor du tissu des PME sur le segment des services et applications mobiles. • L'Open Innovation, via la mise en œuvre de programmes d'accélération technologiques, co-organisés de concert avec les donneurs d'ordres nationaux et à destination des startups IT locales.



FIG. 1.2 : Softcenter

## 1.2 Contexte du projet

### 1.2.1 Motivation et problématique du projet

Le contexte de notre projet est axé sur le domaine du fitness et du bien-être, qui est devenu de plus en plus populaire ces dernières années. De nombreuses personnes cherchent à adopter un mode de vie sain, à améliorer leur condition physique et à atteindre leurs objectifs de bien-être. Dans ce contexte, les plateformes en ligne et les réseaux sociaux ont joué un rôle essentiel en offrant des ressources, des conseils, des programmes d'entraînement et des communautés où les individus partageant les mêmes intérêts peuvent se connecter et s'encourager mutuellement. Cependant, malgré la disponibilité de ces plateformes, il peut être difficile pour les utilisateurs de trouver les bonnes informations et de se connecter avec des personnes partageant les mêmes objectifs et besoins.

Dans ce contexte que notre projet intervient. Nous cherchons à exploiter le pouvoir de l'intelligence artificielle et des communautés en ligne pour créer une expérience de fitness et de bien-être optimale. Notre objectif est de rassembler des individus ayant des intérêts et des objectifs similaires au sein de communautés virtuelles, où ils peuvent échanger des connaissances, partager des conseils, s'entraider et se motiver mutuellement.

Nous utilisons des techniques de scraping de données à partir de plateformes de médias sociaux telles que YouTube, Reddit et Instagram pour collecter des informations pertinentes sur des sujets tels que la nutrition, l'exercice, les régimes, les maladies chroniques, etc. Ensuite, en utilisant des méthodes d'analyse de données et d'intelligence artificielle, nous regroupons les utilisateurs en fonction de leurs caractéristiques et objectifs communs.

Notre projet vise à créer une plateforme en ligne conviviale et engageante, où les membres peuvent non seulement trouver des informations pertinentes et de qualité, mais aussi se connecter avec des personnes partageant les mêmes intérêts, participer à des défis, des compétitions et des programmes d'entraînement collectifs, et bénéficier d'un soutien mutuel tout au long de leur parcours de bien-être.

En résumé, notre projet s'inscrit dans le contexte croissant du fitness et du bien-être, en tirant parti des plateformes en ligne, de l'intelligence artificielle et des communautés virtuelles pour offrir une expérience de fitness et de bien-être améliorée. Nous visons à créer une plateforme inclusive, informative et interactive pour les individus cherchant à améliorer leur santé et leur qualité de vie.

### 1.2.2 Objectifs du projet

Les objectifs du projet sont les suivants :

Créer une plateforme en ligne : L'objectif principal est de développer une plateforme en ligne conviviale et interactive dédiée au fitness et au bien-être. Cette plateforme permettra aux utilisateurs de trouver des informations pertinentes, des conseils, des programmes d'entraînement et des ressources pour les aider à atteindre leurs objectifs de bien-être.

Faciliter la création de communautés de fitness en ligne : L'objectif est de permettre aux utilisateurs

de se connecter avec d'autres personnes partageant les mêmes intérêts et objectifs en matière de fitness et de bien-être. Les utilisateurs pourront rejoindre des communautés virtuelles où ils pourront échanger des connaissances, partager des expériences, s'encourager mutuellement et participer à des activités collectives.

Utiliser l'intelligence artificielle pour la personnalisation : L'objectif est d'utiliser des techniques d'intelligence artificielle pour analyser les données des utilisateurs, comprendre leurs besoins, leurs préférences et leurs objectifs, afin de leur fournir des recommandations et des contenus personnalisés. Cela permettra aux utilisateurs de bénéficier d'une expérience adaptée à leurs besoins spécifiques.

Encourager l'engagement et la motivation : L'objectif est de créer une plateforme qui encourage l'engagement actif des utilisateurs. Cela peut se faire en proposant des défis, des compétitions, des programmes d'entraînement collectifs, des récompenses et des incitations pour maintenir la motivation des utilisateurs à atteindre leurs objectifs de fitness et de bien-être.

Fournir des ressources de qualité : L'objectif est de garantir que les informations, les conseils et les ressources disponibles sur la plateforme sont de qualité et basés sur des preuves scientifiques. Cela permettra aux utilisateurs d'avoir accès à des informations fiables et pertinentes pour prendre des décisions éclairées concernant leur santé et leur bien-être.

Favoriser l'apprentissage et le partage des connaissances : L'objectif est de créer un environnement propice à l'apprentissage et au partage des connaissances entre les utilisateurs. Cela peut inclure des forums de discussion, des sessions de questions-réponses, des webinaires, des blogs, etc. L'objectif est de créer une communauté d'apprentissage où les utilisateurs peuvent acquérir de nouvelles connaissances et s'inspirer mutuellement.

En résumé, les objectifs du projet sont de créer une plateforme en ligne qui facilite la création de communautés de fitness, utilise l'intelligence artificielle pour personnaliser l'expérience des utilisateurs, encourage l'engagement et la motivation, fournit des ressources de qualité et favorise l'apprentissage et le partage des connaissances.

## 1.3 Conduite du projet

### 1.3.1 Processus de développement adopté

Le projet adopte une approche de développement agile pour favoriser la flexibilité, la collaboration et l'adaptabilité aux changements. Le processus de développement est itératif et incrémental, ce qui permet de livrer des fonctionnalités utilisables à intervalles réguliers tout au long du projet.

### 1.3.2 Présentation de la méthode Scrum

Le projet suit la méthode Scrum, qui est l'une des approches agiles les plus populaires. Scrum se base sur une approche itérative et incrémentale, avec une équipe auto-organisée et pluridisciplinaire. Les principaux éléments de Scrum comprennent les sprints (itérations de développement de courte durée), les rôles clés tels que le Product Owner et le Scrum Master, ainsi que les cérémonies telles que la réunion de planification du sprint, la revue de sprint et la rétrospective.

### 1.3.3 Planification du projet

La planification du projet est effectuée en utilisant une approche itérative. Les objectifs, les livrables et les échéances sont définis pour chaque itération du projet. L'équipe de développement travaille en étroite collaboration avec le Product Owner pour prioriser les fonctionnalités, définir les exigences et évaluer les efforts nécessaires pour chaque itération. La planification est régulièrement révisée et ajustée en fonction de l'évolution des besoins et des circonstances.

### **1.3.4 Outils de suivi du projet**

Pour suivre et gérer le projet, plusieurs outils sont utilisés, tels que : Un outil de gestion de projet : Il permet de planifier les tâches, de suivre leur avancement, d'affecter des ressources et de gérer les dépendances. Des outils courants comme Jira, Trello ou Asana peuvent être utilisés pour cela.  
on a utilisé Trello pour la gestion de notre projet.

Des outils de communication et de collaboration : Ils facilitent la communication au sein de l'équipe de développement et avec les parties prenantes du projet. Ces outils comprennent des plateformes de messagerie instantanée comme Slack, des outils de vidéoconférence comme Zoom et des outils de partage de documents comme Google Drive ou Microsoft Teams.

on a utilisé Google meet pour vidéoconférence et la communication avec l'encadrante .

Ces outils aident à organiser, suivre et documenter le travail effectué tout au long du projet, en permettant une collaboration efficace et une transparence des informations entre les membres de l'équipe de développement et les parties prenantes.

## **Conclusion**

Tout au long de ce chapitre, nous avons évoqué le contexte général du projet pour se situer dans la problématique sur laquelle on travaille.

Le chapitre suivant sera consacré tout d'abord à la collection des données.

# Chapitre 2

## COLLECTION DES DONNÉES

Dans cette section, nous abordons la collecte de données, une étape fondamentale de notre projet. Étant donné l'absence d'un ensemble de données existant adapté à notre étude, nous avons dû créer notre propre ensemble de données à l'aide de techniques de web scraping. Cependant, cela a soulevé des questions essentielles telles que les variables à extraire, les thèmes à rechercher et les plateformes à cibler. Nous discuterons de nos réponses à ces questions, ainsi que de la conception de notre dataset final, dans cette section.

### Table des matières

---

2.1	Collection des données à partir de Reddit : . . . . .	18
2.1.1	Pourquoi Reddit ? . . . . .	18
2.1.2	Explication du code : . . . . .	18
2.2	Collection des données à partir d'Instagram : . . . . .	21
2.2.1	Pourquoi Instagram ? . . . . .	21
2.2.2	Explication du code : . . . . .	21
2.3	Collection des données à partir de YouTube : . . . . .	25
2.3.1	Pourquoi YouTube ? . . . . .	25
2.3.2	Explication du code : . . . . .	25

---

# Introduction

Pour entamer notre projet, la première étape consistait à collecter des données. Étant donné l'absence d'un ensemble de données préexistant adapté à notre étude, nous avons décidé de créer notre propre ensemble de données en utilisant des techniques de web scraping. Cependant, cette approche a présenté certaines difficultés, nécessitant des réponses aux questions suivantes :

- Quelles variables devons-nous extraire pour constituer notre ensemble de données ?
- Quels sont les thèmes ou sujets que nous devons rechercher ?
- Comment créer cette dataset (autrement dit quels sites web ou plateformes à utiliser) ?

En ce qui concerne la première question, notre objectif était de nous baser sur des informations relatives aux clients, telles que leur âge, leur sexe, leur pays, ainsi qu'une description personnelle décrivant leurs intérêts dans le domaine du bien-être. Cependant, il est difficile de trouver les données personnelles (tels âge, sexe, pays) sur Internet, et même si elles sont disponibles, elles sont souvent limitées à un nombre restreint de personnes cependant nous souhaitions constituer un ensemble de données plus large pour obtenir des résultats plus pertinents (on a pensé à générer ces donner aléatoirement mais on a trouvé que cela va juste biaiser les résultats or que ces données sont juste complémentaires). C'est pour cela on a décidé de collecter juste les commentaires (car c'est la variable la plus importante pour nous donner une vision claire sur les centres d'intérêts d'un client afin de lui affecter des communautés pertinentes) ainsi que les noms des auteurs (pour regrouper les commentaires créés par le même auteur et éviter tout biais). Ainsi, notre ensemble de données final comprendra deux colonnes : "auteur" et "commentaire".

Concernant les thèmes à rechercher, nous avons identifié trois sujets principaux dans le domaine du bien-être : la santé, le fitness et la nutrition. Cependant, au fur et à mesure de nos recherches, nous avons découvert d'autres sous-sujets intéressants. Voici une liste finale des principaux thèmes que nous avons utilisés : [- Fitness - nutrition - obesity - diabetes - smoking - intermittent fasting - vegetables and fruits - drink water - sugar challenge - hear loss - teeth health - heart disease - weak up early - lower back pain - eye disease - dry eye - blood pressure - nerves - Exercise routines for beginners - Yoga - Healthy meal plans and recipes - mental health - Running and endurance training - Strength training and muscle building - Sleep hygiene and the importance of rest - healthy lifestyle].

Maintenant que nous avons défini la structure de notre ensemble de données et les sujets à rechercher, nous devons sélectionner les plateformes à cibler pour collecter les données. Cette étape a été particulièrement difficile, car nous devions trouver des plateformes contenant des espaces de partage où les gens peuvent exprimer leurs points de vue à travers des commentaires. De plus, ces commentaires devaient être pertinents, car souvent nous trouvions des contenus inutiles tels que "ok" ou "merci". Nous avons donc dû chercher des plateformes où des commentaires pertinents étaient disponibles pour les sujets mentionnés précédemment. Après de nombreuses recherches, nous avons finalement choisi quatre plateformes : Reddit, Instagram et YouTube.

Maintenant, nous sommes prêts à passer à la collecte de données, et nous discuterons les codes utilisés dans l'axe suivant.

## 2.1 Collection des données à partir de Reddit :

### 2.1.1 Pourquoi Reddit ?

Reddit est un choix privilégié pour notre projet de détection de communautés virtuelles en raison de sa diversité de communautés couvrant de nombreux sujets, notamment la santé et le fitness. Avec sa vaste base d'utilisateurs actifs, nous aurons accès à une abondance de commentaires provenant de différentes communautés. Les discussions sur Reddit sont publiques, nous permettant de collecter des données sans violer la vie privée des utilisateurs. De plus, les commentaires sur Reddit sont riches en informations et opinions, fournissant une source de données textuelles précieuses pour l'analyse et le clustering. L'engagement communautaire élevé sur Reddit garantit des discussions dynamiques et en temps réel, exactement ce qu'on cherche pour notre projet.

### 2.1.2 Explication du code :

Pour commencer, il faut créer dans un premier temps un compte développeur Reddit : Rendez-vous sur le site web de Reddit et créez un compte développeur. Cela nous permettra d'obtenir les informations d'identification nécessaires pour accéder à l'API Reddit.



FIG. 2.1 : Le portail du compte développeur Reddit

Après, vous installez la bibliothèque PRAW (Python Reddit API Wrapper) pour interagir avec l'API Reddit dans le langage python

```
%pip install praw

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting praw
  Downloading praw-7.7.0-py3-none-any.whl (189 kB)
  189.4/189.4 kB 5.0 MB/s eta 0:00:00
Collecting prawcore<3,>=2.1
  Downloading prawcore-2.3.0-py3-none-any.whl (16 kB)
Requirement already satisfied: websocket-client>=0.54.0 in /usr/local/lib/python3.10/dist-packages (from praw) (1.5.1)
Collecting update-checker>=0.18
```

FIG. 2.2 : Téléchargement de la bibliothèque PRAW

Utilisez maintenant les informations d'identification (client ID et client secret) pour vous authentifier auprès de l'API Reddit. Cela vous permettra d'effectuer des requêtes et d'accéder aux données. J'ai entré mes informations crédentielles avec le code suivant :

```

import praw
reddit = praw.Reddit(client_id = "z-tw9lI1w20nqzaKOZGq3A",
                     client_secret = "l1dE79ayjmTHLWC77KRW Ao1Mvxm50g",
                     user_agent = "review")

```

FIG. 2.3 : Configurer l'accès à l'API de Reddit

Faire de l'extraction est plutôt facile à ce stade. J'ai spécifié après une liste search\_queries contenant les différentes requêtes de recherche qu'on souhaite utiliser pour récupérer les commentaires sur Reddit.

```

search_queries = ["smoking ", "intermittent fasting", "eating vegetables and fruits only",
                  "drink water", "sugar challenge", "intermittent fasting", "hear loss",
                  "teeth health", "heart disease", "weak up early", "lower back pain",
                  "eye disease", "dry eye", "blood pressure", "nerves"]

```

FIG. 2.4 : Liste des mots-clés

Avec une boucle for on itère sur chaque requête dans la liste search\_queries et on effectue les opérations suivantes.

```

for querie in search_queries:
    submissions = reddit.subreddit("all").search(querie, sort="relevance", time_filter="all", limit=30)
    comments_list = []

    # Loop through each submission and its comments, and add them to the list
    for submission in submissions:
        submission.comments.replace_more(limit=None)
        for comment in submission.comments.list():
            comments_list.append([submission.title, comment.body])

    # Create a Pandas dataframe from the comments list
    search_df = pd.DataFrame(comments_list, columns=["Author", "Comment"])
    print("Scraping for the querie : ", querie)
    print(search_df.info())
    df = pd.concat([df, search_df], ignore_index=True)

```

FIG. 2.5 : Le code de l'extraction, la manipulation et la sauvegarde des données

- La ligne submissions = reddit.subreddit("all").search(querie, sort="relevance", time\_filter="all", limit=30) effectue une recherche sur le subreddit "all" (tous les sousreddits) en utilisant la requête spécifiée. Les paramètres sort="relevance", time\_filter="all", et limit=30 indiquent que les résultats de la recherche doivent être triés par pertinence, inclure tous les résultats dans toutes les périodes de temps et limiter le nombre de résultats à 30 car au-delà de 30 les commentaires perdent leur pertinence et leur connexion au terme spécifié.
- La liste comments\_list est créée pour stocker les commentaires extraits.
- La boucle for itère sur chaque soumission (post) dans les résultats de la recherche.
- submission.comments.replace\_more(limit=None) permet de charger tous les commentaires d'une soumission, y compris les commentaires cachés derrière le bouton "load more comments".
- La boucle interne itère sur chaque commentaire dans la soumission et ajoute le titre de la soumission et le contenu du commentaire à la liste comments\_list.

- Une fois que tous les commentaires ont été extraits pour une requête donnée, la liste comments\_list est utilisée pour créer un DataFrame Pandas appelé search\_df. Les colonnes du DataFrame sont nommées "Autor" pour l'auteur de la soumission et "Comment" pour le contenu du commentaire.
- Les informations sur le DataFrame search\_df sont affichées à l'aide de la méthode print(search\_df.info()), montrant le nombre de lignes, les colonnes et les types de données.
- La ligne df = pd.concat([df, search\_df], ignore\_index=True) concatène le DataFrame search\_df avec un autre DataFrame existant appelé df (qui doit être défini auparavant). L'option ignore\_index=True garantit que les index sont réinitialisés après la concaténation.

## 2.2 Collection des données à partir d'Instagram :

### 2.2.1 Pourquoi Instagram ?

J'ai choisi d'utiliser Instagram pour scraper des commentaires dans le cadre de mon projet. Instagram est une plateforme populaire où les utilisateurs partagent leurs expériences, leurs opinions et interagissent avec du contenu via des commentaires. En collectant ces commentaires, je peux obtenir des informations précieuses sur les réactions des utilisateurs, leurs points de vue et les discussions autour des sujets qui m'intéressent.

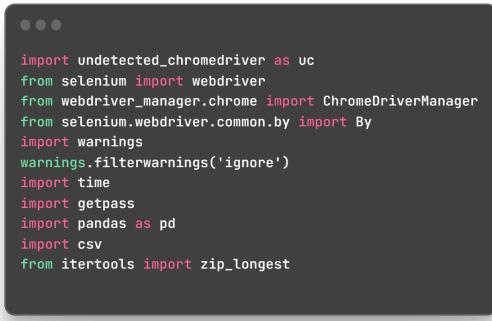
En utilisant des techniques de scraping, j'ai automatisé le processus de collecte des commentaires à partir des publications Instagram. J'ai utilisé des bibliothèques comme Selenium pour naviguer sur les pages, extraire les commentaires et les stocker dans une base de données ou un fichier pour une analyse ultérieure.

Cette approche me permet de recueillir un grand nombre de commentaires provenant de diverses publications, ce qui me donne une vue d'ensemble des opinions et des discussions des utilisateurs sur les sujets liés à mon projet.

### 2.2.2 Explication du code :

Importation des modules nécessaires :

La bibliothèque `undetected_chromedriver` est importée sous le nom de `uc` pour gérer le pilote Chrome. `webdriver` du module `selenium` est importé pour automatiser les actions du navigateur. `ChromeDriverManager` du module `webdriver_manager.chrome` est importé pour gérer le pilote Chrome. `By` du module `selenium.webdriver.common.By` est importé pour localiser les éléments en utilisant différentes stratégies. Le module `warnings` est importé pour supprimer les avertissements éventuels. D'autres modules requis tels que `time`, `getpass`, `pandas`, `csv` et `zip_longest` sont importés.



```
import undetected_chromedriver as uc
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common import By
import warnings
warnings.filterwarnings('ignore')
import time
import getpass
import pandas as pd
import csv
from itertools import zip_longest
```

FIG. 2.6 : Instagram Scraper(1)

Nous commençons d'abord par une condition pour nous assurer qu'il s'exécute uniquement lorsque le fichier est exécuté directement, et non lorsqu'il est importé en tant que module. Une instance de `undetected_chromedriver` est créée en tant que navigateur. La fenêtre du navigateur est maximisée et la page de connexion Instagram est ouverte. Un délai de 3 secondes est ajouté pour attendre le chargement de la page. Les champs nom d'utilisateur et mot de passe sont localisés à l'aide de leur XPath et remplis avec les valeurs fournies. Le bouton de soumission est localisé et cliqué pour se connecter. Un délai de 3 secondes est ajouté après la connexion. Le code tente de localiser et de fermer deux fenêtres contextuelles facultatives qui peuvent apparaître après la connexion. Un délai de 2 secondes est ajouté après la fermeture de chaque fenêtre contextuelle. Les lignes commentées (lignes précédées par le symbole `#`)

```

    ...
if __name__=="__main__":
    browser=uc.Chrome()
    browser.maximize_window()
    browser.get("https://www.instagram.com/")
    time.sleep(1) # waiting for three seconds
    username = browser.find_element(By.XPATH, ('//input[@name="username"]'))
    password = browser.find_element(By.XPATH, ('//input[@name="password"]'))
    username.send_keys("username")
    password.send_keys("pwd")
    submit = browser.find_element(By.XPATH, ('//button[@type="submit"]'))
    submit.click()
    time.sleep()
    try:
        browser.find_element(By.XPATH, ('//button[@class=_acan _acap _acas _aj1- "j"]')).click()
    except:
        pass
    time.sleep()
    try:
        browser.find_element(By.XPATH, ('//button[@class=_a9-- _a9_0"]')).click()
    except:
        pass
    time.sleep()

```

FIG. 2.7 : Instagram Scraper(2)

ne sont pas pertinentes pour l'exécution actuelle et peuvent être ignorées. La variable URL (url) est définie sur

"<https://www.instagram.com/explore/tags/workoutmotivation/>", ce qui représente la page Instagram pour le tag "workoutmotivation". Le navigateur accède à l'URL fournie. Un délai de 10 secondes est ajouté pour permettre le chargement de la page.

```

    ...
# notnow = browser.find_element(By.XPATH, ("//button[contains(text(), 'Plus
tard'))]).click()
# #turn on notif
# time.sleep(10)
# notnow2 = browser.find_element(By.XPATH, ("//button[contains(text(), 'Plus
tard'))]).click()
url ="https://www.instagram.com/explore/tags/workoutmotivation/"
browser.get(url)

time.sleep(10)

items_link=[]

last_height=browser.execute_script("return document.body.scrollHeight")

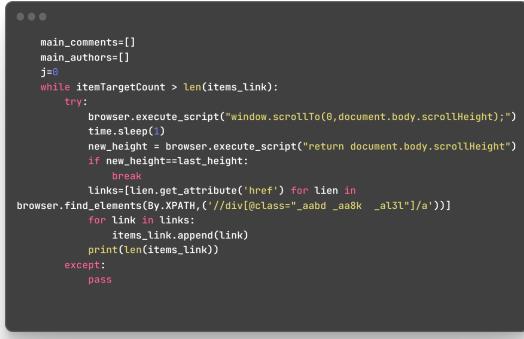
itemTargetCount =200
authors=[]
followers=[]
comments=[]

```

FIG. 2.8 : Instagram Scraper(3)

Les listes "main comments" et "main authors" sont initialisées comme des listes vides pour stocker les commentaires principaux et les auteurs de chaque publication. La variable "j" est initialisée à 0, ce qui sera utilisé comme compteur pour écrire la ligne d'en-tête dans le fichier CSV. Une boucle "while" est utilisée pour faire défiler la page et collecter les liens des publications jusqu'à ce que le nombre souhaité (itemTargetCount) soit atteint ou qu'aucun autre lien ne soit trouvé. Le navigateur exécute du code JavaScript pour faire défiler la page jusqu'en bas. Un délai de 1 seconde est ajouté après le défilement pour permettre le chargement du nouveau contenu. La nouvelle hauteur de la page est calculée en utilisant du code JavaScript. Si la nouvelle hauteur est identique à la hauteur précédente, cela indique qu'il n'y a plus de nouvelles publications à charger et la boucle est interrompue. Les liens des publications sont extraits en localisant les balises <a> dans un XPath spécifique. Chaque lien est ajouté à la liste "items link". La longueur actuelle de "items link" est affichée pour suivre l'avancement. Toutes les exceptions survenues pendant l'exécution sont ignorées.

Ce bloc effectue les actions suivantes pour chaque lien :



```

main_comments=[]
main_authors=[]
j=0
while itemTargetCount > len(items_link):
    try:
        browser.execute_script("window.scrollTo(0,document.body.scrollHeight);")
        time.sleep(1)
        new_height = browser.execute_script("return document.body.scrollHeight")
        if new_height==last_height:
            break
    except:
        pass
    links=[lien.get_attribute('href') for lien in
browser.find_elements(By.XPATH,'//div[@class="_abbd _aa8k _a33t"]/a')]
    for link in links:
        items_link.append(link)
    print(len(items_link))
except:
    pass

```

FIG. 2.9 : Instagram Scraper(4)

- Affiche le lien.
  - Navigue le navigateur vers le lien en utilisant browser.get(link).
  - Ajoute un délai de 1 seconde avec time.sleep(1) pour permettre le chargement de la page.
  - Trouve l'élément du commentaire principal en utilisant son XPath et l'assigne à la variable "comment main".
  - Trouve l'élément de l'auteur du post en utilisant son XPath et l'assigne à la variable poster.
  - Extrait le texte des abonnés en localisant les éléments avec des XPaths spécifiques et les stocke dans la liste followers.
  - Extrait le texte des commentaires en localisant les éléments avec un XPath spécifique et les stocke dans la liste "comments".
  - Affiche le commentaire principal et les abonnés pour chaque post.
  - Itère sur chaque commentaire dans la liste "comments", supprime les espaces vides au début et à la fin, et l'ajoute à la liste comments.
  - Itère sur chaque abonné dans la liste "followers", supprime les espaces vides au début et à la fin, et l'ajoute à la liste "followers".
  - Itère sur la plage de la longueur de "followers", ajoute le commentaire principal à la liste "main comments" et l'auteur (à partir de poster) à la liste "main authors".
  - Affiche les commentaires et les auteurs pour chaque post.
  - Crée une liste appelée "info" qui contient les listes "main authors", "main comments", followers et comments.
  - Utilise "zip longest" pour créer un itérateur qui agrège les éléments de chaque liste ensemble.
  - Ouvre un fichier CSV nommé "fitness advice.csv" en mode ajout et le encode en UTF-8.
  - Crée un objet écrivain CSV et écrit la ligne d'en-tête dans le fichier si j est égal à 0.
  - Écrit les données agrégées de B data dans le fichier CSV.
  - Dans l'ensemble, ce code est responsable de la visite de chaque lien, de l'extraction du commentaire principal, de l'auteur, des abonnés et des commentaires des publications Instagram, et de l'écriture de ces informations dans un fichier CSV nommé "fitness advice.csv".
- \* Les commentaires, les commentaires principaux (main comments), les auteurs principaux (main

```

j=0
for link in items_link:
    try:
        print(link)
        browser.get(link)
        time.sleep(1)
        # wait = WebDriverWait(browser, 10)
        comment_main = browser.find_element(By.XPATH, '//*[contains(concat(" ", @class, " " ), concat(" ", "a9zn", " " ))]/[*[contains(concat(" ", @class, " " ), concat(" ", "_aade", " " ))])')
        poster = browser.find_element(By.XPATH, '/h2[@class="a9zc"]')
        # poster=browser.find_element(By.XPATH,'/h2[@class="a9zc"]')
        followers_= [follower.text for follower in
browser.find_elements(By.XPATH,'//*[contains(concat(" ", @class, " " ), concat(" ", "#x70psk2", " " )) and contains(concat(" ", @class, " " ), concat(" ", "#x566063", " " ))])']
        comments=[comment.text for comment in
browser.find_elements(By.XPATH,'//div[@class="a9zs"]')]
        print(comment_main)
        print(followers_)
        for comm in comments_:
            comments.append(comm.strip())
        for follo in followers_:
            followers.append(follo.strip())
            # followers.extend(followers_)
            # comments.extend(comments_)
        for i in range(len(followers_)):
            main_comments.append(comment_main.text.strip())
            main_authors.append(poster.text.strip())
    except:
        pass

```

FIG. 2.10 : Instagram Scraper(5)

authors), les followers et les commentaires sont imprimés à des fins de débogage. \* Les listes main authors, main comments, followers et comments sont combinées dans la liste info. \* La fonction zip longest est utilisée pour regrouper les éléments de chaque liste de manière à ce que chaque élément soit associé aux autres éléments correspondants. \* Un fichier CSV nommé "fitness advice.csv" est ouvert en mode ajout (mode 'a') avec l'encodage UTF-8. \* Un objet writer est créé pour écrire dans le fichier CSV. \* Si j est égal à 0 (première itération), une ligne d'en-tête contenant les noms des colonnes est écrite dans le fichier. \* Les données de la liste B data sont écrites dans le fichier CSV en utilisant la méthode writerows(). \* Enfin, les listes items link, followers et comments sont imprimées pour le débogage.

Cette partie du code extrait les informations des commentaires, des auteurs et des followers à partir des liens de chaque publication Instagram et les enregistre dans un fichier CSV.

```

print(comments)
print(main_comments)
print(main_authors)

info= [main_authors, main_comments, followers, comments]
B_data = zip_longest(*info)
with open('fitness_advice.csv', 'a',encoding="utf-8") as myFile:
    wr = csv.writer(myFile)
    if j==0:
        wr.writerow(['main_authors',
"main_comments","followers","comments"])
    wr.writerows(B_data)
    j=j+1

except:
    pass

print(items_link)
print(followers)
print(comments)

```

FIG. 2.11 : Instagram Scraper(6)

## 2.3 Collection des données à partir de YouTube :

### 2.3.1 Pourquoi YouTube ?

Nous avons opté pour YouTube comme plateforme de collecte de données en raison de sa capacité à répondre pleinement à nos critères. En effet, YouTube offre une grande variété de vidéos couvrant presque tous les sujets possibles, y compris ceux qui nous intéressent. De plus, chaque vidéo dispose d'un espace dédié aux commentaires, accompagnés des noms des auteurs. Lors de nos tests, nous avons constaté un pourcentage intéressant de commentaires pertinents. Cependant, le défi qui se pose maintenant est de sélectionner les vidéos appropriées.

Au départ, nous avons envisagé de rechercher des chaînes YouTube spécialisées dans chaque sujet, puis de parcourir les vidéos de ces chaînes pour collecter les commentaires. Cependant, nous avons réalisé que cette approche n'était pas pertinente, car les chaînes peuvent contenir des vidéos éloignées du thème principal de la chaîne (par exemple, une vidéo célébrant l'atteinte d'un million d'abonnés).

Une autre approche consiste à rechercher des influenceurs dans chaque domaine et à collecter les commentaires de ses vidéos les plus populaires. Bien que cette méthode soit efficace, elle ajoute une complexité supplémentaire, car la recherche d'influenceurs n'est pas toujours simple.

Par conséquent, nous avons décidé d'adopter une approche plus simple. Nous effectuons donc des recherches sur chaque sujet, en sélectionnant les vidéos les plus consultées (avec des millions de vues) et contenant un nombre important de commentaires (des centaines ou des milliers). Nous créons ensuite une liste de liens vers ces vidéos, que nous utilisons ensuite dans notre code de web scraping. Ce code parcourt chaque lien, scroller plusieurs fois pour collecter tous les commentaires, puis extrait les liens et les commentaires.

### 2.3.2 Explication du code :

Pour commencer, nous avons créé un fichier contenant les liens vers les vidéos que nous avons sélectionnées manuellement. Ensuite, nous avons utilisé le code ci-dessous pour extraire les données nécessaires.

- importation des bibliothèques :

```
● ✓ from selenium import webdriver
  from webdriver_manager.chrome import ChromeDriverManager
  import undetected_chromedriver as uc
  from time import sleep
  from selenium.webdriver.common.keys import Keys
  import pandas as pd
  from selenium.webdriver.common.by import By
  from tqdm import tqdm
```

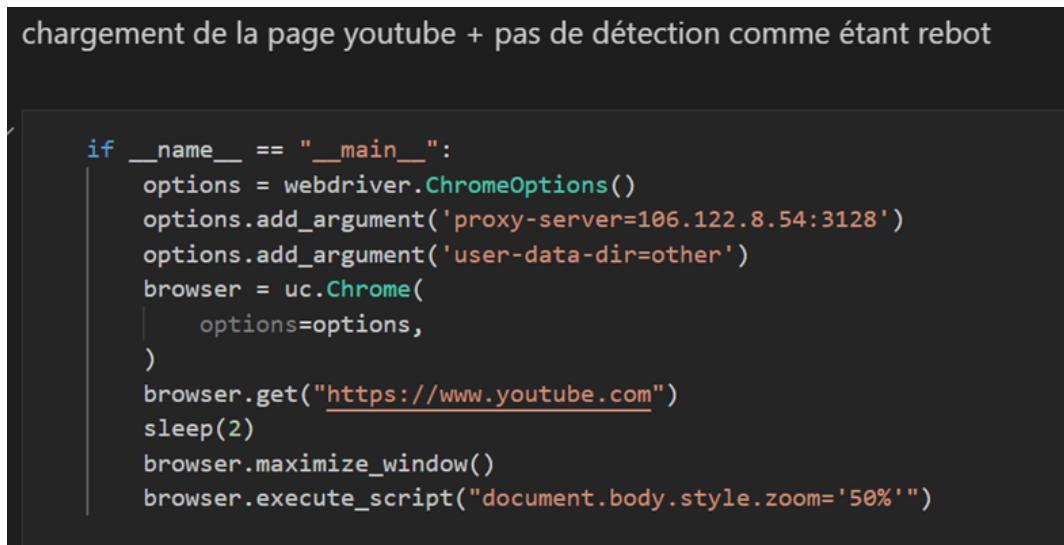
FIG. 2.12 : bibliothèques nécessaires pour Youtbe scraping

Tout d'abord, nous commençons par importer les bibliothèques nécessaires pour le scraping. Pour ce code, nous avons choisi de travailler avec Selenium.

Nous nous concentrons sur quelques bibliothèques spécifiques. Pour commencer, nous utilisons '*undetected chromedriver*' afin d'éviter d'être détectés comme des robots. Utiliser uniquement le pilote Selenium WebDriver normal peut entraîner notre blocage sur certains sites web.

En ce qui concerne la bibliothèque *tqdm*, elle offre une fonctionnalité très intéressante et utile, en particulier pour les codes de scraping. Lorsque nous travaillons sur des projets importants, l'exécution de ce type de code peut prendre beaucoup de temps (par exemple, pour ce script, il a fallu plus de 5 heures). Si nous exécutons le code directement, nous n'avons pas de visibilité sur son avancement. Nous ne savons pas s'il fonctionne correctement ou s'il est bloqué quelque part. Souvent, nous avons des doutes et nous envisageons même de relancer le code. Cependant, en utilisant *tqdm* nous pouvons suivre l'exécution en temps réel. Nous avons des informations telles que le pourcentage d'avancement, le temps écoulé et une estimation du temps restant. Ce qui est également pratique, c'est que son utilisation est très simple. Il suffit d'intégrer le mot "tqdm" dans notre boucle.

- Chargement de la page youtube :



```

if __name__ == "__main__":
    options = webdriver.ChromeOptions()
    options.add_argument('proxy-server=106.122.8.54:3128')
    options.add_argument('user-data-dir=other')
    browser = uc.Chrome(
        options=options,
    )
    browser.get("https://www.youtube.com")
    sleep(2)
    browser.maximize_window()
    browser.execute_script("document.body.style.zoom='50%'")

```

FIG. 2.13 : Chargement de la page youtube

Le rôle principal de ce code est, tout d'abord, de charger le navigateur Chrome en évitant d'être détecté comme un robot grâce à la bibliothèque *undetected\_chromedriver*. Ensuite, il cherche la page "<https://www.youtube.com>" en utilisant la méthode *.get()*. Enfin, il agrandit la fenêtre du navigateur et fixe le zoom à 50% (nous avons utilisé cette option pour afficher plus de contenu dans la fenêtre et réduire le nombre de scrolls nécessaires).

- La fonction scroll :



```

def scroll():
    while(True):
        height = browser.execute_script("return document.body.scrollHeight")
        sleep(1)
        browser.find_element_by_tag_name('body').send_keys(Keys.END)
        if int(height)==0:
            break

```

FIG. 2.14 : la fonction scroll

Nous utiliserons cette fonction pour faire défiler vers le bas de la page afin d'attendre le chargement des nouveaux commentaires. Nous devons répéter cette opération de défilement (scrolling) plusieurs fois pour récupérer tous les commentaires.

- Création de la liste des liens et de la dataframe finale :

```
with open ('pfa_links.txt','r') as f :
    videos_list = f.readlines()
len(videos_list)

162

# df_final= pd.read_csv('final.csv',index_col=0)
df_final = pd.DataFrame(columns=['author','comment'])
df_final

author  comment
```

FIG. 2.15 : création de la liste des liens et de la dataframe finale

Comme mentionné précédemment, nous avons créé manuellement un fichier contenant les liens des vidéos sélectionnées. Nous devons donc lire ce fichier et stocker son contenu (c'est-à-dire les liens) dans une liste que nous utiliserons ultérieurement. De plus, nous avons créé un dataframe dans lequel nous stockerons les données extraites.

- Code de scraping :

```
for link in tqdm(videos_list) :
    browser.get(link)
    sleep(2)
    browser.execute_script('videos = document.querySelectorAll("video"); for(video of videos) {video.pause()}' )
    title = browser.find_elements(by=By.XPATH, value='//h1/yt-formatted-string')[0].text
    youtuber = browser.find_element(by=By.XPATH, value='//div[@id="owner"]//div[@id="text-container"]//a').text

    last_position = browser.execute_script("return window.pageYOffset ;")
    while True:
        fin = True
        for i in range(3):
            scroll()
            sleep(2)
            curr_position = browser.execute_script("return window.pageYOffset ;")
            if (curr_position!=last_position):
                fin = False
                break
        if fin :
            l = browser.find_elements(by=By.XPATH, value='//div[@id="contents"]/yt-comment-thread-renderer//yt-formatted-string[@id="content-text"]')
            m = browser.find_elements(by=By.XPATH, value='//a[@id="author-text"]/span')
            authors = [n.text for n in m]
            comments = [p.text for p in l]
            break
        else: last_position = curr_position
    authors.insert(0,youtuber)
    comments.insert(0,title)

df=pd.DataFrame({'author':authors , "comment":comments})
df["author"] =df['author'].apply(lambda x: None if x == '' else x)
df["comment"] =df['comment'].apply(lambda x: None if x == '' else x)

df_final=pd.concat([df_final,df],axis=0).reset_index(drop=True)
```

FIG. 2.16 : code de scraping

Nous itérons sur chaque lien de notre liste, accédons à chaque lien, puis attendons 2 secondes pour permettre le chargement complet de la page et éviter les problèmes liés à une connexion lente. Ensuite, nous mettons la vidéo en pause en utilisant la méthode execute\_script('videos = document.querySelectorAll("video")'; for(video of videos) video.pause()) (car nous allons accéder à chaque vidéo et effectuer plusieurs scrolls, et pendant ce temps, la vidéo est en cours de lecture, ce qui pourrait entraîner des problèmes supplémentaires). Ensuite, nous extrayons le titre de la vidéo et le nom de la chaîne (nous pensons que

ces informations sont pertinentes et peuvent être intégrées dans notre dataframe). Maintenant, nous passons à la partie principale de notre code :

Le but ici est de répéter le scrolling jusqu'à récupérer tous les commentaires (fin == true). Pour ce faire, tout d'abord, nous récupérons notre position actuelle à l'aide de la méthode .execute-script("return window.pageYOffset ;") et la stockons dans la variable last-position. Ensuite, nous effectuons le scrolling et à chaque itération, nous comparons la position actuelle avec last-position. Si elles sont égales, cela signifie que nous sommes arrivés à la fin de la page (fin == true).

Donc, si nous sommes arrivés à la fin de la page, c'est le moment d'extraire les données. La liste "l" contient les web-elements pour chaque commentaire, de même pour la liste "m" qui contient les web-elements des auteurs. Ensuite, nous récupérons le texte présent dans ces web-elements et le stockons dans les deux listes "comments" et "authors". Enfin, nous sortons de la boucle "while" en utilisant "break". Sinon (fin == false), nous mettons à jour notre variable "last-position" et continuons la boucle. Ensuite, nous insérons le titre et le nom de la chaîne (déjà extraits) au début des deux listes. Ensuite, nous créons une dataframe intermédiaire qui stockera les données pour chaque vidéo (avant d'insérer les valeurs des listes dans la dataframe on a tout d'abord remplacer les champs vides dans les deux colonnes par la valeur None pour faciliter un peu le preprocessing). À la fin de chaque itération de la première boucle, nous concaténons cette dataframe avec la dataframe principale qui contiendra toutes les données collectées.

Remarque : On a utilisé la boucle (for i in range(3)) car il se peut qu' à cause d'une mauvaise connexion le scroll ne se fait pas correctement et par la suite le code va se comporter comme si c'est la fin, c'est pour cela on a utilisé cette boucle pour donner 3 chances au cas de ces problèmes.

A la fin de cette étape on a pu créer un dataset qui contient plus de 150000 lignes avec 2 colonnes (author, comment)

```
df_final.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 150573 entries, 0 to 150572
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   author    150081 non-null   object 
 1   comment   149566 non-null   object 
dtypes: object(2)
memory usage: 3.4+ MB
```

FIG. 2.17 : youtube dataframe

Et voici un exemple des données collectés :

df_final.head()		
	author	comment
0	Dr. Eric Berg DC	How Your Feet Are Warning You About Your Liver...
1	Rukhsana Begum	Someone actually teaching us about our health...
2	Adel Qutob	3 months ago I watched the episode about intim...
3	Liz Zy	I wished every med doctor would spread this kn...
4	Galavarter	I was a truck driver 40lbs over weight.\nFollo...

df_final.tail()		
	author	comment
150568	John Amato	Adaptation. Not evolution. Evolution isn't a re...
150569	She's Niki	It's sad when you talk on the negative effects...
150570	Bosco	this guy look like he has been fasting for a ...
150571	kricku	tl;dr\nDead
150572	M TABARIK ASIF	I'm muslim and proud to say we fast for one mo...

FIG. 2.18 : exemples des données collectés

# Conclusion

Au cours de ce chapitre, nous avons présenté nos choix concernant la collecte des données et la création de notre dataset. Nous avons discuté des informations à extraire, des thèmes que nous avons recherchés et des plateformes que nous avons utilisées. Ensuite, nous avons expliqué les codes que nous avons employés pour atteindre notre objectif.

Enfin, nous avons réussi à collecter plus de 300 000 commentaires.

```
comments_df=pd.read_csv("our_data.csv")

comments_df.shape

(337162, 3)

comments_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 337162 entries, 0 to 337161
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Unnamed: 0    337162 non-null   int64  
 1   author       336525 non-null   object  
 2   comment      298312 non-null   object  
dtypes: int64(1), object(2)
memory usage: 7.7+ MB
```

FIG. 2.19 : data-set finale

On peut constater que c'est une très grande dataset mais cela est nécessaire car un grand pourcentage des commentaires est inutile donc on doit passer par des filtres strictes pour réduire la dataset et ne laisser que les lignes pertinentes, ce qui sera le sujet du prochain axe qui va se concentrer dans le pré-processing

# Chapitre 3

## Preprocessing

Dans ce chapitre, nous allons entamer le prétraitement de notre dataset, c'est est une étape cruciale et joue un rôle essentiel dans la préparation de nos données brutes pour l'étape suivante

### Table des matières

---

3.1	Importation des bibliothèques . . . . .	31
3.2	Prétraitement . . . . .	31
3.3	NLTK . . . . .	34
3.4	WordCloud . . . . .	36

---

### 3.1 Importation des bibliothèques

On importe les bibliothèques nécessaire pour le traitement de la dataset :

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

FIG. 3.1 : Bibliothèques importés

On charge nos données et on supprime complètement l'ancien index

```
data = pd.read_csv('our_data.csv', index_col=0)
data.reset_index(drop=True, inplace=True)
data.info()
✓ 1.3s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 337162 entries, 0 to 337161
Data columns (total 2 columns):
 #   Column   Non-Null Count   Dtype  
 ---  --       --           --      
 0   author    336525 non-null  object  
 1   comment   298312 non-null  object  
dtypes: object(2)
memory usage: 5.1+ MB
```

FIG. 3.2 : Chargement des données extraites

### 3.2 Prétraitement

Avec la cellule suivante, nous supprimons les commentaires redondants, remplissons les cases vides des auteurs avec des noms uniques de la forme "auteurX" où X est un indice qui s'incrémenté à chaque instance d'une case vide, puis convertissons les commentaires en minuscules.

```
data.dropna(subset="comment",inplace=True)
data.drop_duplicates(inplace=True)
length=data[data["author"].isna()].shape[0]
data.loc[data["author"].isna(),"author"]=[f"author{i}" for i in range(length)]
##lower case
data["comment"]=data["comment"].str.lower()
✓ 0.5s
```

FIG. 3.3 : Suppression des commentaires redondants, remplissages des cases vides et transformation en minuscules

Maintenant, il n'y a aucune valeur manquante dans notre ensemble de données.

```
data.isna().sum()
✓ 0.0s

author      0
comment     0
dtype: int64
```

FIG. 3.4 : Nombre des cases vides

Après avoir effectué les opérations mentionnées, il nous reste 188,786 commentaires.

```
data.shape
✓ 0.0s

(188786, 2)
```

FIG. 3.5 : les dimensions actuelles de la dataset

Pour une filtration plus pertinente, nous avons rassemblé une liste de mots clés auxquels chaque commentaire doit correspondre.

```
topics=["workout","body","exercise","working","work","leg","legs","protine","beginner","gym","fitness","gym guide",
'muscle','chest','shoulders','triceps','eat','weight','cardio','build','push','pull','deit','exercises',
'food','nutrition','meal','advice','healthy','fat','training','train','ab','routine','stretch','calorie','fast',
'fasting','intermittent','fruit','sugar','brain','energy','breakfast','morning','eating','health','surgery','drink',
'water','carb','wake','blood','problem','pressure','cold shower','cigarette','heart','bread','egg','sleep','symptom',
'lower back','salt','eye','feeling','pain','quit smoking','smoker','stress','teeth','yoga','liver','dry eye','hair',
'diabete']
```

```
topic_filter=data["comment"].str.contains(r"\b(" + "|".join(topics) + r")\b")
data.drop(data[~topic_filter].index, axis=0, inplace=True)
data.shape
✓ 2.0s
```

```
C:\Users\W\AppData\Local\Temp\ipykernel_36432\1597321870.py:10: UserWarning: This pattern is interpreted as a regular expression, and has match groups. To actually get the groups, use str.extract.
topic_filter=data["comment"].str.contains(r"\b(" + "|".join(topics) + r")\b")
```

```
(80236, 2)
```

FIG. 3.6 : Liste des mots clés et le code de filtre

Après cela, nous avons créé une liste de mots non pertinents par rapport à notre objectif, puis nous avons supprimé chaque commentaire qui contient au moins l'un de ces mots et dont la longueur est inférieure ou égale à 10 caractères.

```

irrelevant=["thank","interesting","top","wonderful","fun","like","subscribe","love","awesome","great",
"amazing","cool","interesting","beautiful","nice","fantastic","wonderful","favorite","thanks"]

irrelevant_filter=data["comment"].str.contains(r"\b(" + "|".join(irrelevant) + r")\b")
length_filter=data["comment"].apply(lambda x: len(x.split())<10

data[irrelevant_filter & length_filter].to_csv('ee.csv')
✓ 0.8s

C:\Users\M.A\AppData\Local\Temp\ipykernel_36432\3991720067.py:4: UserWarning: This pattern is interpreted as a regular expression
    irrelevant_filter=data["comment"].str.contains(r"\b(" + "|".join(irrelevant) + r")\b")

```

FIG. 3.7 : Filtre des mots indésirables et le filtre de la longueur

```

data.drop(data[irrelevant_filter & length_filter].index, axis=0, inplace=True)
data.shape
✓ 0.0s
(79282, 2)

```

FIG. 3.8 : Code de l'exécution des filtres

Avec le code suivant, on supprime les emojis et les autres caractères spéciaux des commentaires. Cela permet de nettoyer les commentaires en éliminant les éléments indésirables et en conservant uniquement les caractères alphanumériques et les espaces.

```

import re

def remove_special_characters(text):
    # Remove emojis
    text = re.sub(r'\\u[\dA-Fa-f]{4}', '', text)

    # Remove other special characters
    text = re.sub(r'[^w\s]', '', text)

    return text

data["comment"] = data["comment"].apply(remove_special_characters)
✓ 0.5s

```

FIG. 3.9 : La fonction de suppression des émojis et des caractères spéciaux

Le code ci-dessous effectue un processus de filtrage des commentaires non anglais dans un DataFrame. Il utilise une bibliothèque appelée "langid" pour détecter la langue de chaque commentaire. Ensuite, il crée une nouvelle colonne dans le DataFrame appelée "lang" qui contient les codes de langue détectés pour chaque commentaire. En filtrant le DataFrame en ne conservant que les lignes où la langue détectée est l'anglais, un nouveau DataFrame est créé qui contient uniquement les commentaires en anglais. Enfin, la colonne temporaire "lang" est supprimée du DataFrame. Cela permet de conserver uniquement les commentaires en anglais pour le traitement ultérieur.

```

import langid

def filter_non_english_comments(df):
    df['lang'] = df['comment'].apply(lambda x: langid.classify(x)[0])
    df = df[df['lang'] == 'en'].copy()
    df.drop('lang', axis=1, inplace=True)
    return df

# Example usage
filtered_df = filter_non_english_comments(data)

✓ 4m 23.6s

```

FIG. 3.10 : la fonction pour la suppression des commentaires non anglais

### 3.3 NLTK

Dans cette partie, nous effectuons des traitements préliminaires à l'aide de la bibliothèque NLTK. Ces traitements comprennent l'élimination des ponctuations, des chiffres et des stopwords, ainsi que l'application des processus de tokenisation et de lemmatisation. Toutes ces opérations sont regroupées dans la fonction preprocess\_text(). Cette fonction permet de préparer le texte ultérieure en éliminant les éléments indésirables et en normalisant le texte.

On importe les modules qu'on va utiliser par la suite.

```

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import string
import re

# Download stopwords and lemmatizer data and punkt
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

```

FIG. 3.11 : Importation des bibliothèques et modules

Maintenant on définit la fonction qui va contenir tout le process de traitement avec NLTK, re et string.

```

def preprocess_text(text):

    # Digits and Punctuation removal and lowercase
    text = text.translate(str.maketrans("", "", string.punctuation))
    text = re.sub(r"\d+", "", text)
    text = text.lower()

    # Tokenize text into words
    tokens = word_tokenize(text)

    # Remove stopwords
    additional_stopwords = list(set(['u','im','dr','berg','enough','use','want','thats','said','sometime','thank','see','much',
                                     'go','find','make','one','day','think','month','year','maybe','week','youre','going','make','thing',
                                     'especially','u','mean','hour','almost','used','ect','look','dont','doesnt','may','etc','told','another',
                                     'easy','right','well','give','age','cant','lot','ive','love','still','dr berg','last','amazing','didnt',
                                     'end','many','went','know','take','come','say','come','gon na','time','video','done','alway','little','hi',
                                     'hour','true','year','ago','become','man','even','isnt','people','everything','literally','without','keto',
                                     'really','thanks','made','seem','got','ate','let','getting','add','pretty','year','old','watch','ok','least',
                                     'found','never','thought','le','put','idea','hard','every','starting','keep','person','feeling','someone','god',
                                     'day','bad','guy','trying','started','understand','new','anone','able','lol','due','big','live','two',
                                     'month','gon','na','sometimes','best','happy','least','felt','daily','hole','last','amazing','definitely',
                                     'tried','part','hope','long','fast','please','half','kind','important','reason','instead','stuff','great',
                                     'making','away','today','using','tell','told','week','normal','something','feel','month','day','year','years',
                                     'actually','added','already','also','always','amount','anyone','anything','around','appreciate','different',
                                     'bit','bp','bro','gave','lb','need','like','oat','get','help','pm','stop']))

    stop_words = set(stopwords.words('english'))
    stop_words.update(additional_stopwords)
    tokens = [word for word in tokens if word not in stop_words]

    # Lemmatize words
    lemmatizer = WordNetLemmatizer()
    # tokens = [lemmatizer.lemmatize(word) for word in tokens]
    tokens = [lemmatizer.lemmatize(word,pos="v") for word in tokens]

    # Join tokens back into a single string
    preprocessed_text = ' '.join(tokens)

    return preprocessed_text

✓ 2.3s

```

FIG. 3.12 : La fonction preprocess\_text()

La fonction preprocess\_text est appliquée à chaque valeur de la colonne "comment" du DataFrame à l'aide de la méthode apply. Cela permet d'appliquer le traitement préliminaire à chaque commentaire individuellement, puis le résultat est assigné à la colonne "without\_stop\_words" dans le DataFrame.

```
filtered_df["without_stop_words"] = filtered_df["comment"].apply(preprocess_text) ⏎
✓ 50.3s
```

FIG. 3.13 : Le code de l'implémentation de la fonction précédente sur les commentaires

Nous regroupons tous les commentaires appartenant au même auteur en les joignant ensemble.

```
grouped_df = filtered_df.groupby('author').agg({'without_stop_words': '/n'.join})
✓ 1.8s
```

FIG. 3.14 : Code de la jointure des commentaires selon l'auteur

## 3.4 WordCloud

WordCloud est une visualisation graphique qui représente les mots les plus fréquents dans un texte donné. Son utilité réside dans sa capacité à donner une représentation visuelle concise et attrayante des mots clés ou des thèmes dominants dans un corpus de texte. Les mots sont affichés dans une image où leur taille est proportionnelle à leur fréquence d'apparition dans le texte. Cela permet de mettre en évidence visuellement les termes les plus importants et les plus récurrents, offrant ainsi un aperçu rapide et intuitif du contenu du texte.

Avec le code suivant on va générer un nuage de mots "WordCloud" à partir des commentaires prétraités dans le DataFrame "grouped\_df" et l'afficher dans une figure utilisant matplotlib. Cela permet de visualiser les mots les plus fréquents dans les commentaires et d'obtenir un aperçu visuel des thèmes ou des mots clés importants dans le texte.

```
import matplotlib
import pylab as plt

from wordcloud import WordCloud
# Generate a word cloud image
wordcloud = WordCloud(width=800, height=600).generate(" ".join(grouped_df['without_stop_words']))
plt.figure(figsize=(16,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
✓ 9.4s
```

FIG. 3.15 : Code de génération du nuage de mots

Le résultat :



FIG. 3.16 : WordCloud généré à partir des commentaires prétraité

La résultat du prétraitement :

```
grouped_df.shape
✓ 0.0s
(73678, 2)
```

FIG. 3.17 : Les dimensions des données prétraités

On sauvegarde la résultat de l'étape de preprocessing dans le fichier 'tot.csv' avec le code suivant :

```
grouped_df.to_csv('tot.csv')
✓ 0.4s
```

FIG. 3.18 : Sauvegarde de la résultat sous le format csv

## Conclusion

Ce chapitre est consacré à la description des étapes parcourues dans la partie de prétraitement et de préparation de la dataset. L'objectif de ces étapes est de préparer la dataset de manière à développer un modèle plus précis et adapté aux attentes spécifiques de notre projet. Ces étapes comprennent des opérations telles que le nettoyage des données, la suppression des informations redondantes, la normalisation du texte et d'autres transformations visant à optimiser la qualité et la pertinence des données pour l'analyse ultérieure. En effectuant ces préparations, nous nous assurons d'avoir des données de haute qualité qui serviront de base solide pour la construction et l'entraînement de notre modèle de manière plus précise et efficace.

# Chapitre 4

## CRÉATION DU MODÈLE

Une fois les données collectées et préparées, il est temps de créer notre modèle de ML, de l'évaluer, de l'optimiser et de le tester sur de nouvelles données. Étant donné la multitude de choix et d'algorithmes disponibles, il est préférable de commencer par étudier les solutions existantes afin de sélectionner la meilleure pour notre cas d'étude. Dans cette optique, nous vous invitons à explorer ces différentes options et à discuter de ces points lors de cet axe intéressant.

### Table des matières

---

4.1	Étude benchmarking : . . . . .	40
4.1.1	Choix du modèle : . . . . .	40
4.1.1.1	Les modèles de clustering : . . . . .	40
4.1.1.2	Autres solutions que le clustering : . . . . .	41
4.1.2	Représentation des données : . . . . .	41
4.1.3	Les paramètres d'évaluation : . . . . .	42
4.2	Explication du code : . . . . .	42
4.2.1	Utilisation du modèle BERT : . . . . .	43
4.2.2	Utilisation de la matrice TF-IDF : . . . . .	46
4.2.3	Modèle finale : . . . . .	47

---

## 4.1 Étude benchmarking :

### 4.1.1 Choix du modèle :

Avant de se lancer dans la création du modèle on a tout d'abord besoin de savoir quel modèle de clustering est convenable pour notre cas, et de chercher s'il existe autre solution hors du clustering.

#### 4.1.1.1 Les modèles de clustering :

- dans un premier temps on a cherché les algorithmes de clustering existants on a trouvé que principalement il y a 3 algorithmes célèbres :

1) KMeans : c'est est l'un des algorithmes les plus couramment utilisés dans une variété de cas. Son objectif principal est de regrouper les données en clusters en fonction de paramètres réglables tels que le nombre de clusters et le type de distance utilisée (comme la distance euclidienne ou de Manhattan). Le processus de fonctionnement de KMeans commence par l'initialisation aléatoire des centroïdes. Ensuite, il assigne chaque point de données au centroïde le plus proche en calculant la distance entre eux. Ce processus est répété jusqu'à ce que les centroïdes se stabilisent et que les affectations ne changent plus significativement. Finalement, nous obtenons des clusters formés par les centroïdes et les points de données qui leur sont attribués.

2) DBSCAN : DBSCAN (Density-Based Spatial Clustering of Applications with Noise) est un algorithme largement utilisé pour le clustering dans des ensembles de données de grande taille et de densité variable. L'algorithme DBSCAN se distingue par sa capacité à identifier des régions de haute densité entourées de régions de faible densité (nested clusters), tout en étant capable de gérer efficacement les points de données aberrants.

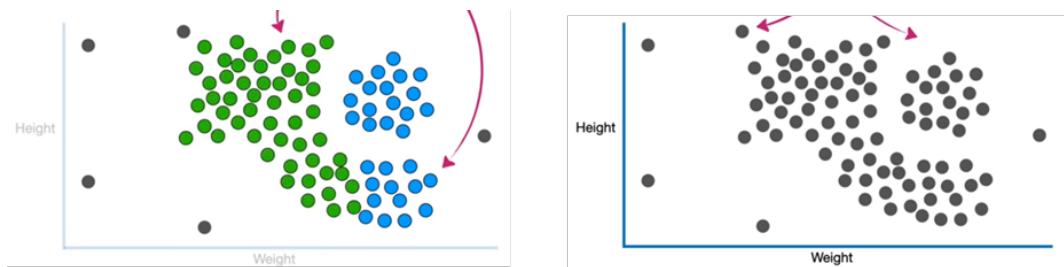


FIG. 4.1 : nested clusters

Par exemple pour ce cas de figure dans cette image il est remarquable qu'on a 2 clusters mais il sont imbriqué donc des modèles comme KMeans vont donner des fausses résultats.

Aussi, contrairement à KMeans, DBSCAN n'exige pas de spécifier le nombre de clusters à l'avance, ce qui le rend très flexible. Au début, DBSCAN choisit un point de données arbitraire et vérifie la densité locale en explorant son voisinage. Ensuite, il étend le cluster en ajoutant les points voisins qui satisfont à certains critères de densité. Ce processus se répète jusqu'à ce que tous les points pertinents aient été inclus dans des clusters ou marqués comme du bruit. Ainsi, DBSCAN permet d'identifier des clusters de formes complexes et de détecter les points aberrants dans les ensembles de données.

3) hierarchical clustering : La classification hiérarchique est un algorithme de clustering qui vise à regrouper les données en fonction de leur similarité. Contrairement à KMeans et DBSCAN, la classification hiérarchique ne nécessite pas de spécifier le nombre de clusters à l'avance. Au lieu de cela, elle

construit une structure arborescente appelée dendrogramme qui représente les relations de similarité entre les données. Le processus commence par considérer chaque point de données comme un cluster individuel, puis il fusionne progressivement les clusters similaires pour former des clusters plus grands. Il existe deux approches principales dans la classification hiérarchique : l'approche agglomérative et l'approche divisive. L'approche agglomérative commence avec des clusters individuels et les fusionne itérativement en fonction de leur proximité, jusqu'à ce qu'un seul cluster global soit formé. L'approche divisive, quant à elle, commence avec un seul cluster global et le divise en sous-clusters jusqu'à ce que chaque point de données soit assigné à son propre cluster. La classification hiérarchique offre une visualisation naturelle des relations de similarité grâce au dendrogramme, ce qui permet d'explorer la structure des données à différentes échelles.

Il existe autres modèles comme MiniBatchKMeans qui est une version de KMeans mais avec moins de calculs de distance (on calcule juste pour des biches et on généralise le résultat) il est utilisable surtout pour les larges dataset, et autres modèles mais on va se concentrer sur KMeans et DBscan

#### 4.1.1.2 Autres solutions que le clustering :

Nous avons exploré d'autres solutions que les modèles de clustering, en particulier l'utilisation du deep learning. Nous avons découvert qu'il est possible d'employer le deep learning, mais pas pour créer directement les communautés. Il peut plutôt être utilisé en tant qu'étape préliminaire, c'est-à-dire que nous devons d'abord entraîner notre ensemble de données avec un modèle DL afin d'obtenir une représentation plus pertinente des données. En particulier, nous avons constaté qu'il existe des modèles DL appelés "transformers" qui sont spécialisés dans le traitement des données textuelles. Il existe déjà des modèles implémentés et entraînés que l'on peut utiliser directement, comme le célèbre BERT de Google (nous en discuterons également dans la partie suivante). Bref, le DL peut être utilisé pour obtenir une représentation plus pertinente des données. Pour approfondir ce point, nous aborderons la prochaine partie.

#### 4.1.2 Représentation des données :

Pour entraîner des modèles de ML ou de DL en général, il est nécessaire de travailler avec des données numériques. Cependant, dans notre cas, nos données sont textuelles. Par conséquent, nous devons d'abord créer une représentation numérique de nos données. Nous avons plusieurs choix à notre disposition, mais nous nous concentrerons sur les trois méthodes les plus couramment utilisées :

1) Utilisation de CountVectorizer : Le principe de cette approche est de créer une matrice où les colonnes représentent tous les mots présents dans notre ensemble de données (nous pouvons sélectionner les mots pertinents en ajustant les paramètres du CountVectorizer). Les lignes de la matrice représentent les représentations numériques des commentaires, et les valeurs indiquent combien de fois chaque mot apparaît dans chaque commentaire. Cette approche est simple et rapide à exécuter.

2) Utilisation de la matrice TF-IDF : Le principe de cette méthode est similaire à celui du CountVectorizer, à la différence que les valeurs représentent l'importance d'un mot dans un commentaire. Un mot est considéré comme important pour un commentaire (score proche de 1) s'il est présent dans ce commentaire et rarement présent dans les autres. La formule utilisée est la suivante :  $\text{tfidf}(t, d, D) = f_t \cdot d \times \log_{10}(N/n_t)$  (où  $t$  représente le mot,  $d$  représente le document (ou commentaire),  $D$  représente l'ensemble des documents,  $n_t$  représente le nombre de fois où le mot apparaît dans tous les documents, et  $N$  représente le nombre total de documents). Cette formule comprend deux termes principaux :

$f_t, d$  : la fréquence du mot dans le commentaire, où une valeur élevée indique que le mot apparaît fréquemment dans le document (commentaire).

$\log_{10}(N/n_t)$  : plus le mot apparaît fréquemment dans les autres documents, plus  $n_t$  augmente et ce terme diminue. Remarque : Ces deux approches ne tiennent pas compte de la sémantique des textes. Par exemple, la représentation du mot "fitness" dans les phrases "Je veux commencer un programme de fitness" et "Je n'aime pas le fitness" serait la même dans ces deux approches.

3) Utilisation du modèle BERT : Pour aborder le défi de capturer la sémantique des commentaires, nous avons choisi d'utiliser le modèle BERT, qui est déjà disponible. Il suffit de le télécharger et de l'appliquer à notre ensemble de données. Cependant, il y a quelques contraintes à prendre en compte lors de l'utilisation de ce modèle. Tout d'abord, il ne prend pas en charge les textes de plus de 512 mots à encoder. De plus, il prend beaucoup de temps pour s'exécuter. Cependant, en fin de compte, il nous fournit un vecteur de taille 768 pour chaque commentaire, ce qui représente une représentation sémantique riche.

#### 4.1.3 Les paramètres d'évaluation :

La dernière chose que nous avons recherchée est comment évaluer les modèles afin de choisir le meilleur. Contrairement à l'apprentissage supervisé, nous n'avons pas de paramètres clairs tels que l'exactitude (accuracy). Après nos recherches, nous avons identifié quelques paramètres, tels que le score de silhouette (silhouette\_score). Son principe est de mesurer le degré de séparation entre les clusters. Plus la valeur est élevée, plus les clusters sont bien séparés, ce qui indique un modèle approprié. Cependant, pour notre cas, les commentaires sont souvent étroitement liés, car les sujets sont interconnectés. Par exemple, il est impossible de parler d'un programme de fitness sans évoquer un programme nutritionnel approprié. Nous avons constaté que le score de silhouette est généralement faible, quel que soit le modèle utilisé. Par conséquent, il est inutile pour la comparaison et ne fait qu'ajouter un temps d'exécution supplémentaire (car ce paramètre compare les distances entre un point et ses voisins dans le cluster par rapport aux distances avec les points des autres clusters).

Nous avons également rencontré le même problème avec d'autres paramètres. C'est pourquoi nous avons opté pour une autre méthode d'évaluation, qui peut sembler simpliste, mais qui nous aide à évaluer correctement les modèles. Nous avons décidé qu'un modèle donnait de bons résultats s'il répondait à ces critères :

- 1) Produire des clusters équilibrés (par exemple, éviter d'avoir un cluster avec 40 000 points et les autres avec seulement quelques centaines de points).
- 2) Examiner les thèmes des clusters à chaque fois et vérifier s'ils correspondent à notre étude de cas (nous expliquerons comment nous avons extrait les thèmes dans la partie dédiée au code).
- 3) Accéder aux commentaires de certains clusters de manière aléatoire et vérifier s'il existe des relations entre eux.

Maintenant, nous avons rassemblé tous les éléments nécessaires. Il est temps de passer aux codes et de tester les différents scénarios possibles afin de choisir le modèle final.

#### 4.2 Explication du code :

Afin de choisir le modèle approprié on a essayé plusieurs scénarios en changeant entre les modèles (KMeans, DBSCAN) et les méthodes utilisées pour récupérer les représentations numériques des commentaires (TF-IDF, BERT model) voici les codes utilisés et les résultats obtenus :

#### 4.2.1 Utilisation du modèle BERT :

```

from sklearn.cluster import KMeans, DBSCAN
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import torch
import transformers
from tqdm import tqdm
## This code will suppress warning messages
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')

```

FIG. 4.2 : bibliothèques de l'ML

```

Use the BERT (Bidirectional Encoder Representations from Transformers) transformer to get a semantic representation of each comment

# Load the pre-trained BERT tokenizer
tokenizer = transformers.BertTokenizer.from_pretrained('bert-base-uncased')
# Load the pre-trained BERT model
model = transformers.BertModel.from_pretrained('bert-base-uncased')

# Define a function to encode the text using BERT
def encode_text(text):
    # Tokenize the text
    tokens = tokenizer.encode(text, add_special_tokens=True) # spetial_tokens : CLS and SEP
    # Convert the tokens to PyTorch tensors
    tokens_tensor = torch.tensor([tokens])
    # Get the output of the BERT model
    with torch.no_grad(): # we don't need to update the model's parameters
        outputs = model(tokens_tensor)
        encoded = outputs[0][:, 0, :].numpy() # give us the vector fixed_size (768) representation of each token but we
                                            # select only the first one CLS because it is used as a way of summarizing
                                            # the entire input sequence into a single vector
    return encoded

```

FIG. 4.3 : bert encoding

Ce code utilise le modèle pré-entraîné BERT pour encoder du texte. Tout d'abord, la tokenizer de BERT est chargée à partir du modèle pré-entraîné 'bert-base-uncased'. Ensuite, le modèle BERT lui-même est chargé.

Ensuite, une fonction appelée encode\_text est définie. Cette fonction prend en entrée un texte et procède à son encodage en utilisant BERT. Le texte est d'abord découpé en jetons (tokens) à l'aide de la tokenizer. Les jetons obtenus sont ensuite convertis en tenseurs PyTorch. Ensuite, le modèle BERT est utilisé pour obtenir la sortie correspondante aux tenseurs de jetons.

Dans cette sortie, nous nous intéressons particulièrement au premier jeton [CLS], car il est utilisé comme une manière de résumer l'ensemble de la séquence d'entrée en un seul vecteur de représentation. Ce vecteur est extrait à partir de la sortie du modèle BERT, en prenant uniquement la première colonne [CLS]. Enfin, ce vecteur encodé est renvoyé comme résultat de la fonction.

Cette fonction sera ensuite utilisée pour encodée chaque commentaires dans notre dataset et stocker les résultats dans une nouvelle colonne 'bert\_encoding', puis on stocke cette nouvelle dataframe dans un nouveau fichier (car cette opération nécessite beaucoup de temps c'est pour cela il n'est pas logique de répéter cela à chaque fois on veut exécuter le code)

```

df['comment']=df['comment'].apply(lambda x: x[-512:] if len(x)>512 else x)

df['bert_encoding']=[0]*len(df)

# encoded_comments = np.vstack(df['comment'].apply(encode_text))
encoded_comments = []
for i in tqdm(df['comment']):
| encoded_comments.append(encode_text(i))

df['bert_encoding1'] = df['bert_encoding'].apply(lambda x: np.squeeze(x))
df.drop('bert_encoding', axis=1, inplace=True)

import pickle

# Save the trained model to disk
with open('bert_final.pkl', 'wb') as f:
| pickle.dump(df, f)

```

FIG. 4.4 : comments encoding using bert model

Pour stocker les données dans un fichier, vous avez envisagé d'utiliser le format Parquet en raison de ses nombreux avantages. Par rapport aux fichiers CSV, le format Parquet permet de récupérer les données dans leur forme initiale. En effet, les données stockées dans les fichiers Parquet sont stockées de manière optimisée par colonne, ce qui permet une meilleure compression des données et une récupération plus efficace lors des opérations de lecture. Contrairement aux fichiers CSV, où les données sont souvent stockées sous forme de chaînes de caractères, le format Parquet conserve les types de données d'origine, ce qui facilite les opérations ultérieures sur les données sans nécessiter de conversions supplémentaires. En utilisant le format Parquet, on peut bénéficier donc d'une meilleure efficacité de stockage et de lecture, ainsi que de la possibilité de conserver la structure et les types de données d'origine.

Ensuite on a traîné les 2 modèles (KMeans et DBscan) avec cette représentation mais on a eu des mauvais résultats voici par exemple le résultat en utilisant le modèle KMeans

```

trying bert encoding

np.vstack(df['bert_encoding'].dropna().tolist())

(41251, 768)

from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score

kmeans = KMeans(n_clusters=35, random_state=42)
kmeans.fit(np.vstack(df['bert_encoding'].dropna().tolist()))

C:\Users\a\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\cluster\_kmeans.py:10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warning
warnings.warn(
    "The `n_init` parameter is set to 'auto'. In version 1.4, it will be set to 10 by default. Set the value of 'n_init' explicitly to suppress the warning.", UserWarning, stacklevel=2)

KMeans
KMeans(n_clusters=35, random_state=42)

tot=df.dropna(subset='bert_encoding')
tot.shape

(41251, 4)

```

FIG. 4.5 : KMeans model using Bert encoding

```

cluster_labels = kmeans.labels_
tot['cluster'] = cluster_labels.tolist()

C:\Users\a\AppData\Local\Temp\ipykernel_5788\266753180.py:2: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row_indexer,col_indexer] = value instead
see the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
tot['cluster'] = cluster_labels.tolist()

tot[tot['cluster']==1]['comment'].to_csv('bert.csv')

```

FIG. 4.6 : Result for the first scenario

Alors on va tester ce modèle avec les critères précédemment cités, on a essayé pour le premier cluster pour voir si les commentaires sont liés mais on a eu des mauvais résultats.

C'est pour cela on a opté pour la représentation TF-IDF car elle est plus simple et elle nous donne des résultats intéressantes, donc elle nous reste juste de choisir entre KMeans et DBSCAN.

#### 4.2.2 Utilisation de la matrice TF-IDF :

Pour se faire tout d'abord on doit créer la matrice TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer
print('Tfidf ponderations')

# Initialize the "TfidfVectorizer" object.
tfidf_vect = TfidfVectorizer(max_features=300)
X_tfidf = tfidf_vect.fit_transform(total['without_stop_words'])

# Numpy arrays are easy to work with, so convert the result to an array
vectorizer_features = X_tfidf.toarray()
tfidf_frequency_matrix = pd.DataFrame(vectorizer_features,columns=tfidf_vect.get_feature_names_out())
tfidf_frequency_matrix
```

FIG. 4.7 : la matrice TF-IDF

Le paramètre "max\_features" est utilisé pour limiter le nombre de variables (mots) dans le modèle TF-IDF. Dans ce code, il est fixé à 300, ce qui signifie que seuls les 300 mots les plus importants seront inclus dans la matrice TF-IDF. Cette valeur est choisie de manière empirique en effectuant plusieurs tests et en évaluant la pertinence des mots sélectionnés. La méthode "get\_feature\_names\_out()" permet de visualiser les noms des variables sélectionnées, ce qui facilite la décision d'ajuster le nombre "max\_features". Ainsi, à la fin de cette étape, on obtient une matrice avec 300 colonnes représentant les mots les plus significatifs dans les textes d'entrée. Cette réduction de dimensionnalité permet de réduire le temps d'exécution et de se concentrer sur les caractéristiques les plus informatives pour l'analyse ultérieure.

Cette matrice sera cédée aux 2 modèles pour l'entraînement :

Voici le résultat qu'on a obtenu avec le modèle DBSCAN :

```
dbscan

from sklearn.cluster import DBSCAN
from sklearn import metrics

# Create and fit the DBSCAN model
dbscan = DBSCAN(eps=0.3, min_samples=5)
dbscan.fit(vectorizer_features)

# Evaluate the clustering results
labels = dbscan.labels_

# Evaluate the performance metrics
silhouette_score = metrics.silhouette_score(vectorizer_features, labels)
calinski_harabasz_score = metrics.calinski_harabasz_score(vectorizer_features, labels)
davies_bouldin_score = metrics.davies_bouldin_score(vectorizer_features, labels)

print("Silhouette Score:", silhouette_score)
print("Calinski-Harabasz Score:", calinski_harabasz_score)
print("Davies-Bouldin Score:", davies_bouldin_score)
```

Silhouette Score: -0.24224711637338  
Calinski-Harabasz Score: 19.0985355108623  
Davies-Bouldin Score: 1.0522472184557408

FIG. 4.8 : Modèle DBSCAN

Comme vous pouvez le voir on a eu des mauvaises résultats, mais comme on a déjà cité on ne doit pas se contenter de ces paramètres et on doit vérifier nos critères.

```
unique_values, value_counts = np.unique(labels, return_counts=True)
unique_values, value_counts
```

Output exceeds the `size limit`. Open the full output data [in a text editor](#)

```
(array([-1,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9,  10,  11,
       12,  13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,
       25,  26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,
       38,  39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,
       51,  52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,
       64,  65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,
       77,  78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,
       90,  91,  92,  93,  94,  95,  96,  97,  98,  99,  100,  101,  102,
      103,  104,  105,  106,  107,  108,  109,  110,  111,  112,  113,  114,  115,
      116,  117,  118,  119,  120,  121,  122,  123,  124,  125,  126,  127,  128,
      129,  130,  131,  132,  133,  134,  135,  136,  137,  138,  139,  140,  141,
      142,  143,  144,  145,  146,  147,  148,  149,  150,  151,  152,  153,  154,
      155,  156,  157,  158,  159,  160,  161,  162,  163,  164,  165,  166,  167,
      168,  169,  170,  171,  172,  173,  174,  175,  176,  177,  178,  179,  180,
      181,  182,  183,  184,  185,  186,  187,  188,  189,  190,  191,  192,  193,
      194,  195,  196,  197,  198,  199,  200,  201,  202,  203,  204,  205,  206,
      207,  208,  209,  210,  211,  212,  213,  214,  215,  216,  217,  218,  219,
      220,  221,  222,  223,  224,  225,  226,  227,  228,  229,  230,  231,  232,
      233,  234,  235,  236,  237,  238,  239,  240,  241,  242,  243,  244,  245,
      246,  247,  248,  249,  250,  251,  252,  253,  254,  255,  256,  257,  258,
      259,  260,  261,  262,  263,  264,  265,  266,  267,  268,  269,  270,  271],  
    dtype=int64),  
array([151716,   116,   36,   72,   10,   25,    6,   150,    5,  
      50,    17,   42,  107,   10,    5,   98,   35,    5,  
      5,     5,    8,    5,    6,    32,    5,   10,    5,
```

FIG. 4.9 : DBSCAN results

Comme vous pouvez le marquer le modèle nous donne 272 clusters mais ils ne sont pas équilibrés le premier cluster contient plus de 90% des points alors qu'il y a autre clusters qui contient juste 5 points donc ce modèle n'est pas approprié pour nous

### 4.2.3 Modèle finale :

D'après ce qui précède on a opté pour notre dernier scénario qui consiste à utiliser le modèle KMeans avec la représentation TF-IDF.

Voici le code du modèle KMeans (on a utilisé la matrice obtenu précédemment par le TfidfVectorizer pour entraîner ce modèle) :

```
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score

kmeans = KMeans(n_clusters=45, random_state=42)
kmeans.fit(vectorizer_features)

C:\Users\al\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\cluster\_kmean
10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
warnings.warn(
    "KMeans(n_clusters={0}, random_state={1})".format(n_clusters, random_state))

cluster_labels = kmeans.labels_
total['cluster'] = cluster_labels.tolist()

feature_names = tfidf_vect.get_feature_names_out()
centroids = kmeans.cluster_centers_
cluster_topics = []
for i, centroid in enumerate(centroids):
    top_indices = centroid.argsort()[-5:][::-1]
    cluster_topics.append([feature_names[ind] for ind in top_indices])
```

FIG. 4.10 : Modèle KMeans

Dans ce code, nous utilisons le modèle KMeans pour effectuer une segmentation de nos données en clusters. Nous commençons par instancier le modèle KMeans avec un petit nombre initial de clusters

(k). Ensuite, nous entraînons le modèle et récupérons les principaux thèmes associés à chaque cluster en utilisant la position du centroïde du cluster, ensuite nous évaluons si ces thèmes sont pertinents pour notre cas d'étude. Puis on augmente le k et on teste si les thèmes ajoutés sont aussi acceptable ou non et on a répété cela jusqu'à s'arrêter au nombre k=45.

Après entraînement du modèle on va extraire le cluster pour chaque commentaire dans notre dataset avec l'attribut labels\_ et on va stocker le résultat dans une nouvelle colonne qui va contenir le cluster prédit pour chaque ligne.

Ensuite on a pensé à récupérer les principaux thèmes pour chaque cluster car comme c'est cité cela nous a aidé à évaluer le choix du nombre des clusters k.

Donc comme résultat on aura une liste à 2-dimensions ‘cluster\_topics’ dont la forme est (45, 5).

Voici un exemple des éléments de cette liste :

```

1 Cluster 1 Top Words: liver, foot, problem, clean, doctor
2 Cluster 2 Top Words: eat, egg, train, routine, would
3 Cluster 3 Top Words: pressure, blood, high, doctor, low
4 Cluster 4 Top Words: morning, routine, wake, eat, night
5 Cluster 5 Top Words: hair, loss, lose, oil, grow
6 Cluster 6 Top Words: smoke, quit, cigarette, smoker, since
7 Cluster 7 Top Words: salt, water, breathe, eat, work
8 Cluster 8 Top Words: show, eat, study, body, exercise
9 Cluster 9 Top Words: beginner, workout, start, exercise, gym
10 Cluster 10 Top Words: pain, back, lower, exercise, work

```

FIG. 4.11 : Clusters topics

Dans la dernière étape, nous évaluons notre modèle en utilisant de nouveaux commentaires dont nous souhaitons prédire le cluster. Nous avons créé une fonction qui prend comme entrée le nouveau commentaire. Cette fonction applique d'abord les mêmes étapes de prétraitement que celles utilisées précédemment, telles que le retrait des ‘stop words’ et la lemmatisation.

Ensuite, nous récupérons la représentation TF-IDF de ce texte prétraité afin d'obtenir un vecteur de caractéristiques. Ce vecteur est ensuite utilisé comme entrée pour la méthode predict() du modèle KMeans, ce qui permet de prédire le cluster auquel ce nouveau commentaire appartient.

Enfin, nous affichons le cluster prédit, les principaux thèmes associés à ce cluster et également 5 commentaires appartenant au même cluster, sélectionnés de manière aléatoire. Cela nous permet de mieux comprendre le contexte et les caractéristiques des commentaires appartenant à ce cluster prédit, ce qui facilite l'analyse et l'interprétation des résultats obtenus.

Voici le code de la fonction et quelques tests qui ont donné des bons résultats :

```

prediction

def predict(my_text):
    preprocessed_text = preprocess_text(my_text)
    text_vector = tfidf_vect.transform([preprocessed_text])
    predicted_cluster = kmeans.predict(text_vector)
    print(f"cluster {predicted_cluster}")
    print(f"this cluster discuss these topics {cluster_topics[int(predicted_cluster)]}")
    print("\nthese are some comment of the same cluster\n",total[total['cluster']==int(predicted_cluster)]['comment'].sample(n=5).values)

```

FIG. 4.12 : Exemple 1

```
my_text = "i am a beginner want to start a new workout program and i don't know where to start "
predict(my_text)

cluster [8]
this cluster discuss these topics ['beginner', 'workout', 'start', 'exercise', 'gym']

these are some comment of the same cluster
['bro i am beginner and my weight is 51 can i do with proper diet',
 'this was beginner and i barely made it through took lots of breaks cant wait till im able to get through the entire thing without stopping',
 'hello sir the for the video i also want lower body workout sir and for beginners how many days do i have to follow this tutorial',
 'legs were shaking during the stretch cool down and this is the beginner workout youve got an amazing level of fitness',
 'i have a question [ni understand that for a beginner these are the only exercises you should prioritise and focus on but should you do any stretches before and after the workout']']
```

FIG. 4.13 : Exemple 2

```
my_text = "i want to start intermittent fasting what i should do"
predict(my_text)

cluster [25]
this cluster discuss these topics ['intermittent', 'lose', 'eat', 'weight', 'work']

these are some comment of the same cluster
['intermittent fasting is easy when you dont have much money',
 'anyone who is teen and still want to try intermittent fasting',
 'this guy is amazing been watching his videos for two years before i decided to give intermittent fasting a try down 27 pounds in 3 months all i did was eat and walk i still ate cardio tho and i wasnt supper strict on myself',
 'thats true i never wanted to eat early morning it was something unnatural for me but civilization brainwashed me and here i am i am on intermittent fasting already 4 days 168 and absolutely happy that i dont have to keep on eating the whole day greeting from russia to everyone',
 'thats awesome ive been intermittent fasting now for four and a half months and lost nearly 60 lb i did it once before and lost 58 lb in two and a half months and was as skinny as i was in my twenties im 52 now intermittent fasting is absolutely amazing i do weightlifting and did no cardio the first time this time im doing weightlifting and hitt training using the heavy bag i only have about 25 lbs more to lose to reach my goal and i will have lost 84 lb']
```

FIG. 4.14 : Exemple 3

FIG. 4.15 : Exemple 4

```
my_text = "i suffer from eye dry espacially when i work in my laptop"
predict(my_text)

cluster [19]
this cluster discuss these topics ['eye', 'dry', 'drop', 'doctor', 'cause']

these are some comment of the same cluster
['i am a female after i turned 51 i started having dry eyes my eye lids started to get swollen\n\nmy eye doctor said that i started having blepharitis and had eye stye know the top lids get cracked and sometimes bleed leaving a red arch on top of my lids'
'dr please is there a relation between severe eye dryness and eyelid concretions'
'and the youtube algorithm wins again i definitely needed to see this with my extremely dry eyes i think this was the motivation i needed to finally talk to my optometrist about how my eyelids stick to my eyes every morning sometimes its so bad i literally wake up in the middle of the night because its so painful maybe im sleeping with my eyes open '
'ive been doing if for only a week and lost 5 lbs i also cut out refined sugar my dark under eyes are gone and i have energy that i havent had in years this is something i can do for life '
'your mention of lagophthalmos describes what ive been experiencing this sensation after waking up that your eyelids are sticking to my eyeballs has been painful and alarming after going to an eye doctor they tested for tear production which was found normal they didnt mention the lagophthalmos or the allergy medication that you did they acted as though this stickiness really wasnt that big a deal and told me to use more drops and use eye ointment at night which i cant tolerate because of the blur!']
```

FIG. 4.16 : Exemple 5

# Conclusion

Au cours de chapitre, nous avons tout d'abord discuté les différents scénarios possibles pour choisir le plus convenable pour notre cas d'étude, en suite on a créé notre modèle finale en utilisant le modèle KMeans avec la représentation TF-IDF, et on l'a évalué et amélioré plusieurs fois et enfin on a utilisé ce modèle pour prédire des nouveaux commentaires.

Jusqu'à ce point on a pu réaliser une recommandation interne des communautés, il nous reste de faire la recommandation externe et de grouper tous cela dans une simple interface web, ce qui sera discuté dans le chapitre suivant.

# Chapitre 5

## communautés externes et déploiement

Nous allons entamer ce chapitre par une présentation des outils utilisés pour la réalisation de ce projet et leur utilité puis nous allons passer pour décrire les différentes étapes du développement des différentes interfaces de notre solution.

### Table des matières

---

5.1	Détection des communautés externes . . . . .	52
5.1.1	Définition et concepts . . . . .	52
5.1.2	Explication du code : . . . . .	52
5.1.2.1	Integration du chatgpt : . . . . .	52
5.1.2.2	Extraction des liens à partir du text : . . . . .	53
5.2	Déploiement avec Flask . . . . .	54
5.2.1	Outils et Technologies . . . . .	54
5.2.2	Implémentation du code . . . . .	56
5.2.2.1	Configuration de l'environnement virtuel . . . . .	56
5.2.2.2	Ecrire l'application avec flask . . . . .	58
5.2.2.3	création des pages web . . . . .	63
5.2.2.4	Démarrage du serveur . . . . .	65

---

## 5.1 Détection des communautés externes

### 5.1.1 Définition et concepts

Lorsqu'un utilisateur entre ses préférences dans notre plateforme, nous utilisons ces informations pour interroger l'API ChatGPT et obtenir des suggestions de communautés virtuelles externes qui correspondent le mieux à ses intérêts. L'API ChatGPT est capable de comprendre et de traiter des instructions en langage naturel, ce qui nous permet d'obtenir des recommandations plus précises et adaptées.

Nous envoyons une requête à l'API ChatGPT en lui fournissant des détails sur les préférences de l'utilisateur, tels que les sujets d'intérêt, les types de communautés recherchées, et d'autres informations pertinentes. En utilisant ces informations comme contexte, l'API ChatGPT génère des réponses contenant des recommandations de communautés virtuelles externes.

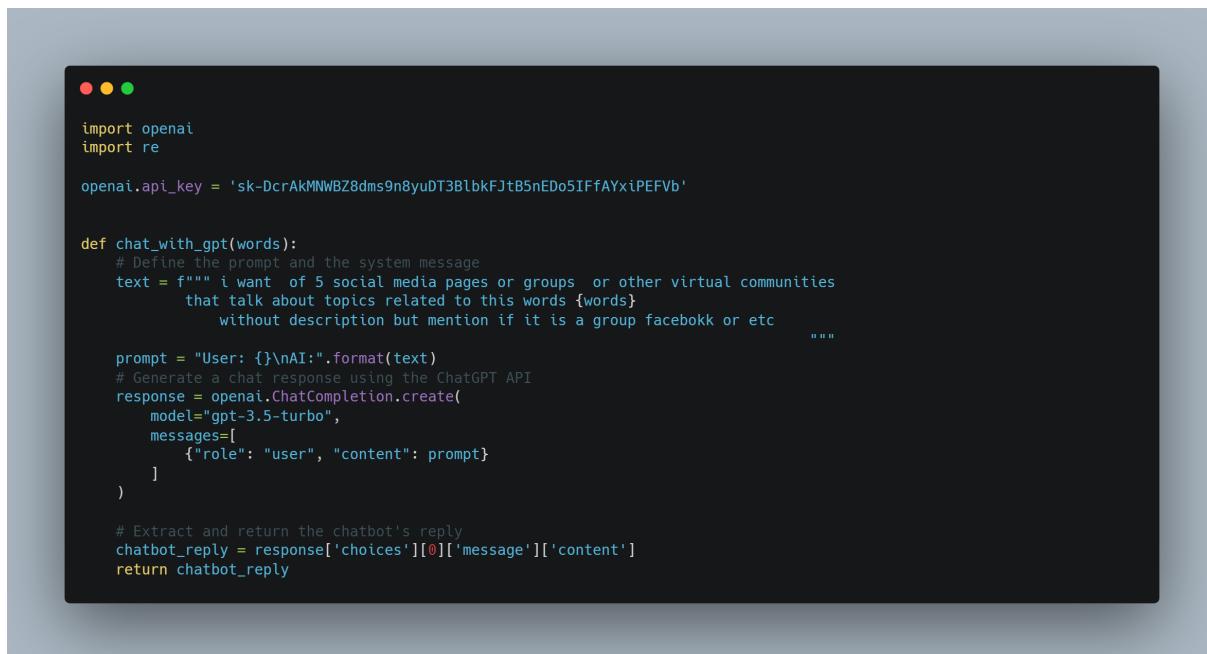
Il est important de noter que notre système repose sur un modèle de langage pré-entraîné et que les réponses générées par l'API ChatGPT sont basées sur les exemples d'entraînement qu'il a reçus. Par conséquent, les recommandations peuvent varier en fonction des exemples d'entraînement et de la qualité des informations fournies par l'utilisateur.

### 5.1.2 Explication du code :

Nous avons utilisé l'API du chatgpt pour obtenir les liens vers les communautés externes puis on utiliser les expressions régulières pour extraire juste les liens pertinentes.

#### 5.1.2.1 Integration du chatgpt :

la fonction appelée chatWithGpt qui interagit avec l'API ChatGPT d'OpenAI pour générer une réponse de chat en fonction d'une entrée donnée words.



```
import openai
import re

openai.api_key = 'sk-DcrAKMNWBZ8dms9n8yuDT3B1bkFJtB5nEDo5IFfAYxiPEFVb'

def chat_with_gpt(words):
    # Define the prompt and the system message
    text = f""" i want of 5 social media pages or groups or other virtual communities
    that talk about topics related to this words {words}
    without description but mention if it is a group facebokk or etc
    """
    prompt = "User: {}".format(text)
    # Generate a chat response using the ChatGPT API
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "user", "content": prompt}
        ]
    )
    # Extract and return the chatbot's reply
    chatbot_reply = response['choices'][0]['message']['content']
    return chatbot_reply
```

FIG. 5.1 : chatgpt(1)

la fonction appelée chatWithGpt qui interagit avec l'API ChatGPT d'OpenAI pour générer une réponse de chat en fonction d'une entrée donnée words.

```
def chat_with_gpt(words):
    # Define the prompt and the system message
    text = f""" i want  of 5 social media pages or groups  or other virtual communities
    that talk about topics related to this words {words}
        without description but mention if it is a group facebook or etc
    """
    prompt = "User: {}\nAI:".format(text)
    # Generate a chat response using the ChatGPT API
```

FIG. 5.2 : image

Dans cette partie, la fonction définit la variable text, qui est une chaîne de caractères formatée contenant le message d'accueil souhaité pour le chatbot. Elle inclut le paramètre words dans le message pour spécifier le sujet. La variable prompt combine l'entrée de l'utilisateur et le message du système, le préparant ainsi pour l'interaction avec l'API.

```
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[
        {"role": "user", "content": prompt}
    ]
)
```

FIG. 5.3 : chatgpt(2)

Le code appelle ensuite l'API ChatGPT d'OpenAI en utilisant la méthode openai.ChatCompletion.create(). Il transmet le nom du modèle "gpt-3.5-turbo" et une liste de messages. La liste des messages comprend un seul objet de message avec le rôle "utilisateur" et le contenu correspondant à la variable prompt précédemment définie.

```
chatbot_reply = response['choices'][0]['message']['content']
return chatbot_reply
```

FIG. 5.4 : chatgpt(3)

Après avoir reçu une réponse de l'API, la fonction extrait la réponse du chatbot de la réponse de l'API et l'assigne à la variable chatbotReply. Enfin, la fonction retourne la réponse du chatbot.

#### 5.1.2.2 Extraction des liens à partir du text :

La fonction extractLinesWithNumbers prend un texte en entrée et retourne une liste des lignes qui commencent par un nombre

La fonction extractLinesWithNumbers prend un texte en entrée et retourne une liste des lignes qui commencent par un nombre :

1. Divise le texte en lignes en utilisant le caractère de saut de ligne.

```

● ● ●
def extract_lines_with_numbers(text):
    lines_with_numbers = []
    lines = text.split('\n')

    for line in lines:
        if re.match(r'^\d', line): # Check if the line starts with a digit
            lines_with_numbers.append(re.sub(r'^\d+\.\s*', '', line)) # Remove the leading number

    return lines_with_numbers

```

FIG. 5.5 : links extraction

2. Initialise une liste vide appelée extractLinesWithNumbers pour stocker les lignes qui commencent par un nombre.
3. Parcourt chaque ligne dans la liste lines.
4. Vérifie si la ligne commence par un chiffre en utilisant l'expression régulière
5. Si la ligne commence par un chiffre, supprime le nombre initial en utilisant l'expression régulière.
6. Après avoir parcouru toutes les lignes, la fonction retourne la liste extractLinesWithNumbers contenant les lignes filtrées.

## 5.2 Déploiement avec Flask

### 5.2.1 Outils et Technologies

Le déploiement d'un modèle dans une application web est une étape essentielle pour mettre en valeur ses fonctionnalités. Dans le cadre de ce rapport, nous avons choisi d'utiliser Flask, un framework Python léger et puissant, pour déployer notre modèle. Flask offre une approche simple et flexible pour créer des applications web, ce qui en fait un choix idéal pour notre projet. Dans cette partie, nous détaillerons le processus de déploiement du modèle à l'aide de Flask, en expliquant les différentes étapes nécessaires pour transformer notre modèle en une API web fonctionnelle et accessible.

#### – .Flask



FIG. 5.6 : FLASK

Flask est un framework web léger et flexible écrit en Python. Il est conçu pour faciliter le développement rapide d'applications web et d'API. Flask suit l'approche du "micro-framework",

ce qui signifie qu'il offre un ensemble de fonctionnalités de base sans imposer de structures ou de dépendances spécifiques. Cela permet aux développeurs d'avoir un contrôle total sur leur application et de la personnaliser selon leurs besoins spécifiques.

L'installation et la configuration de Flask sont simples. Vous pouvez l'installer en utilisant l'outil de gestion des paquets Python appelé "pip". Une fois Flask installé, vous pouvez créer un projet Flask de base en définissant un fichier Python et en important la classe Flask. Vous pouvez également configurer un environnement virtuel à l'aide d'outils tels que virtualenv pour isoler les dépendances de votre projet Flask. Cela garantit que les dépendances spécifiques à votre projet sont gérées séparément et n'interfèrent pas avec d'autres projets Python sur votre système.

Dans Flask, les routes jouent un rôle essentiel. Les routes permettent d'associer des URL spécifiques à des fonctions de vue. Vous pouvez définir une route en utilisant le décorateur `@app.route` suivi de l'URL correspondante. Lorsqu'un utilisateur accède à cette URL dans son navigateur, Flask appelle la fonction de vue correspondante et retourne le résultat au navigateur. Les fonctions de vue peuvent renvoyer du contenu HTML, des données JSON, des fichiers statiques, etc. Flask offre également un système de routage dynamique où vous pouvez utiliser des paramètres variables dans l'URL pour capturer des valeurs spécifiques et les utiliser dans votre logique de traitement.

Flask facilite également le rendu de vues à l'aide de modèles. Vous pouvez utiliser des moteurs de modèles tels que Jinja2 pour créer des modèles HTML réutilisables. Les modèles vous permettent de séparer la logique de présentation de votre application de la logique de traitement des données. Flask facilite l'intégration des modèles dans vos vues et vous permet de passer des données dynamiques aux modèles pour générer du contenu personnalisé.

## – .HTML



FIG. 5.7 : HTML

HTML, acronyme de HyperText Markup Language, est le langage de balisage standard utilisé pour la création de pages web. Il est utilisé pour structurer le contenu d'une page web et définir la manière dont le contenu doit être affiché sur un navigateur.

Les balises HTML sont les éléments de base utilisés pour marquer et structurer le contenu d'une page web. Elles sont entourées par des chevrons `<>` et peuvent avoir des attributs pour spécifier des propriétés supplémentaires.

HTML utilise une structure en arborescence appelée DOM (Document Object Model). Chaque balise HTML est un noeud dans l'arborescence et peut contenir d'autres balises HTML à l'intérieur. Par exemple, un élément `<div>` peut contenir des éléments `<p>`, des éléments `<img>`, ou d'autres éléments `<div>`. Cette structure hiérarchique permet de définir la mise en page et la relation entre les différents éléments sur la page.

## – .CSS



FIG. 5.8 : CSS

CSS, acronyme de Cascading Style Sheets, est un langage de feuilles de style utilisé pour décrire la présentation et l'apparence des éléments HTML sur une page web. Il permet de contrôler la couleur, la police, la disposition, les marges, les bordures, les animations et d'autres aspects visuels d'un site web.

Les règles CSS sont appliquées aux éléments HTML en utilisant des sélecteurs. Les sélecteurs ciblent les balises HTML spécifiques ou des classes et des identifiants attribués aux balises. Par exemple, pour appliquer un style à toutes les balises du document, on peut utiliser le sélecteur `h1`. Pour appliquer un style à toutes les balises avec une classe spécifique, on peut utiliser le sélecteur `.nom-de-classe`.

Les propriétés CSS sont utilisées pour définir les styles des éléments sélectionnés. Chaque propriété a une valeur qui spécifie comment l'élément doit être stylisé. Par exemple, la propriété `color` définit la couleur du texte, la propriété `font-size` définit la taille de la police, et la propriété `background-color` définit la couleur de fond d'un élément.

En utilisant CSS, il est possible de créer des mises en page complexes, d'ajuster la position des éléments sur la page, d'appliquer des effets de transition et d'animation, de créer des menus déroulants, de définir des styles responsives pour s'adapter à différentes tailles d'écran, et bien plus encore. CSS permet également d'appliquer des styles conditionnels en fonction de l'état d'un élément, tel que `:hover` pour appliquer un style lorsqu'un élément est survolé par la souris.

## 5.2.2 Implémentation du code

Dans cette phase d'implémentation, nous abordons le déploiement du modèle, une étape essentielle pour mettre en œuvre notre solution de manière pratique et accessible. Le déploiement du modèle consiste à créer une interface conviviale permettant aux utilisateurs d'interagir avec notre algorithme de prédiction, en exploitant ses capacités pour résoudre des problèmes spécifiques.

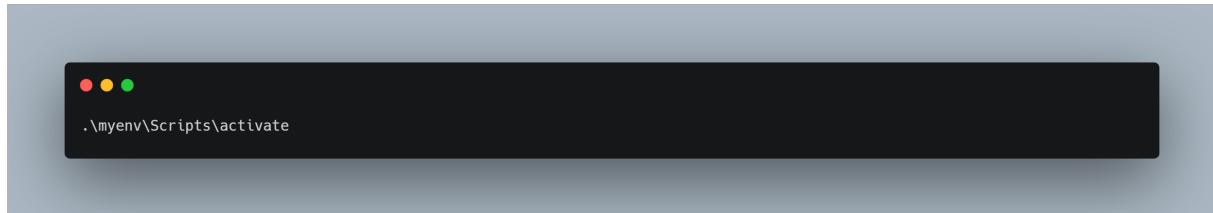
### 5.2.2.1 Configuration de l'environnement virtuel

Pour la création d'un environnement virtuel on va taper la commande dans l'invite de commande :

A screenshot of a terminal window on a Mac OS X system. The window has a dark grey header bar with three colored dots (red, yellow, green) in the top-left corner. The main area of the terminal is light grey. At the bottom of the screen, there is a dark grey footer bar. In the terminal window, the command `python -m venv myenv` is typed in and is visible in white text on the dark background.

FIG. 5.9 : create virtual environment

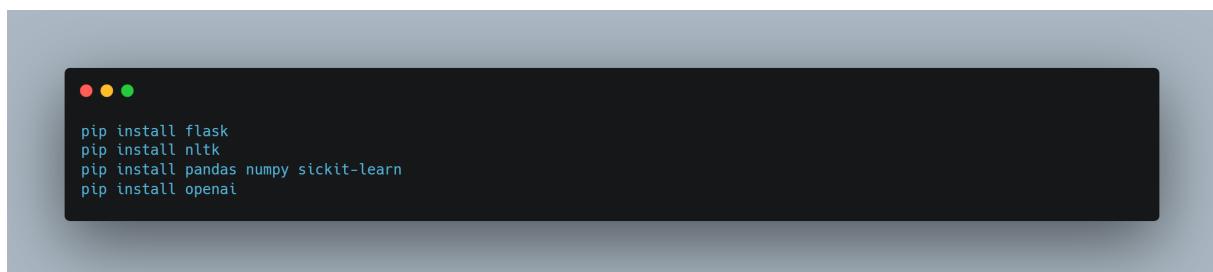
puis on va l'activer :



```
.\myenv\Scripts\activate
```

FIG. 5.10 : activation

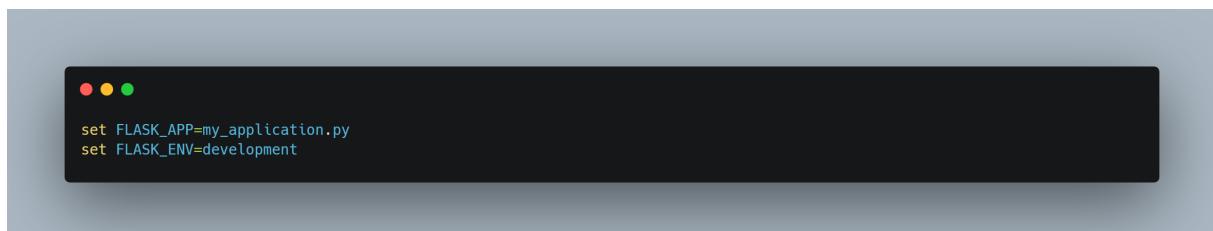
puis on va installer les bibliothèques nécessaires pour le déploiement :



```
pip install flask
pip install nltk
pip install pandas numpy scikit-learn
pip install openai
```

FIG. 5.11 : libraries

puis on va configurer les variables d'environnement :

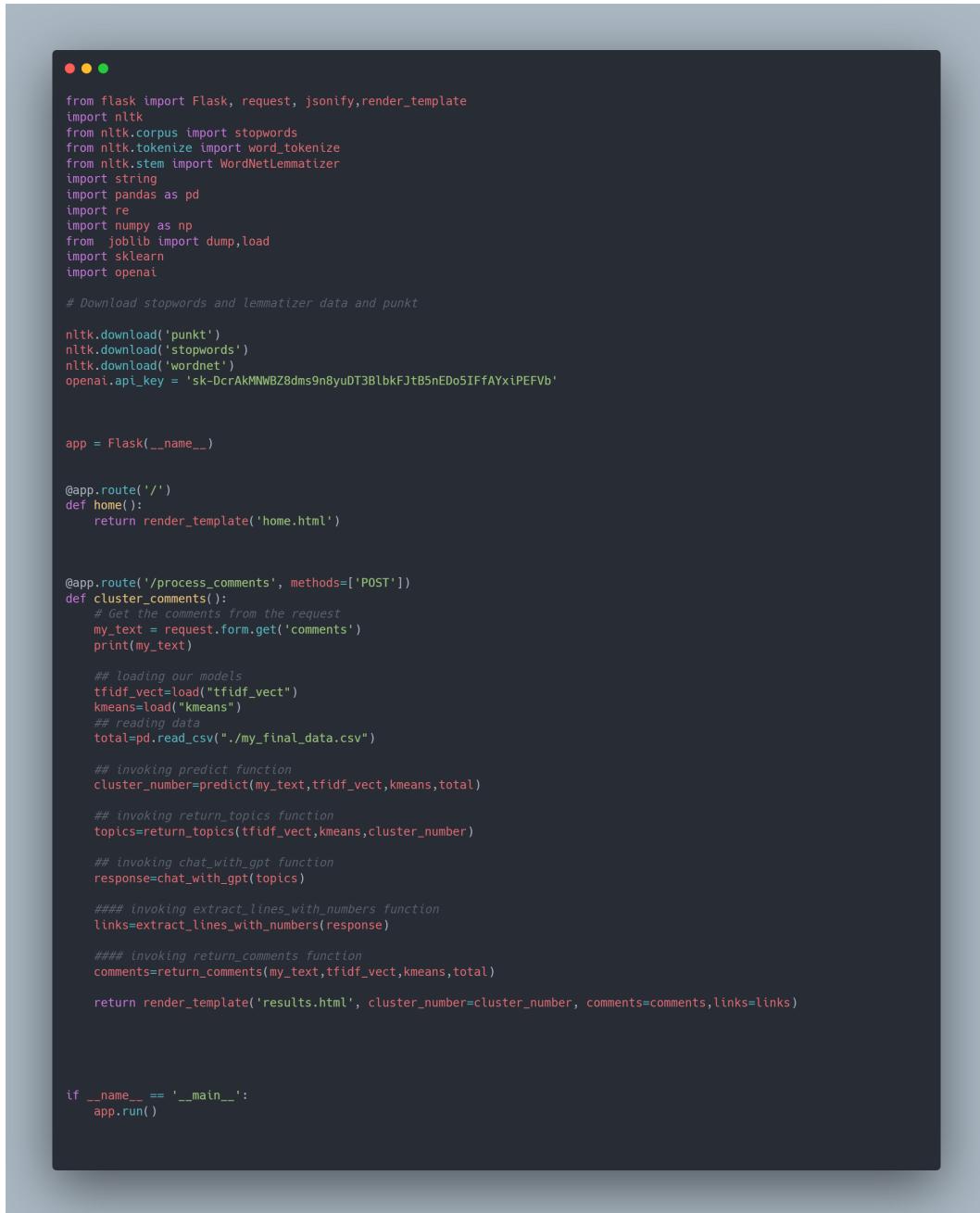


```
set FLASK_APP=my_application.py
set FLASK_ENV=development
```

FIG. 5.12 : ENV variables

### 5.2.2.2 Ecrire l'application avec flask

Pour commencer la partie du code de notre application Flask, nous allons créer un fichier Python qui servira de point d'entrée pour notre application. Nous l'appellerons généralement application.py. Dans ce fichier, nous allons importer le module Flask et initialiser notre application en utilisant la syntaxe suivante :



```
● ● ●

from flask import Flask, request, jsonify, render_template
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import string
import pandas as pd
import re
import numpy as np
from joblib import dump, load
import sklearn
import openai

# Download stopwords and lemmatizer data and punkt

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
openai.api_key = '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'

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/process_comments', methods=['POST'])
def cluster_comments():
    # Get the comments from the request
    my_text = request.form.get('comments')
    print(my_text)

    ## loading our models
    tfidf_vect=load("tfidf_vect")
    kmeans=load("kmeans")
    ## reading data
    total=pd.read_csv("./my_final_data.csv")

    ## invoking predict function
    cluster_number=predict(my_text,tfidf_vect,kmeans,total)

    ## invoking return_topics function
    topics=return_topics(tfidf_vect,kmeans,cluster_number)

    ## invoking chat_with_gpt function
    response=chat_with_gpt(topics)

    ##### invoking extract_lines_with_numbers function
    links=extract_lines_with_numbers(response)

    ##### invoking return_comments function
    comments=return_comments(my_text,tfidf_vect,kmeans,total)

    return render_template('results.html', cluster_number=cluster_number, comments=comments,links=links)

if __name__ == '__main__':
    app.run()
```

FIG. 5.13 : application.py

dans notre application on a défini plusieurs fonctions :  
on commence par la fonction **process** qui consiste à traiter les données et les nettoyer en supprimant les caractères manquantes ,les chiffres,les ponctuations etc :



```

def preprocess_text(text):
    if isinstance(text, float):
        return ""

    # Digits and Punctuation removal and lowercase
    text = re.sub(r"\d+", "", text)
    text = text.lower()
    text = text.translate(str.maketrans("", "", string.punctuation))

    # Tokenize text into words
    tokens = word_tokenize(text)

    # Remove stopwords
    additional_stopwords = list(set(['good', 'u', 'im', 'dr', 'berg', 'enough', 'use', 'want', 'thats', 'said', 'sometime', 'thank', 'see', 'much',
                                     'go', 'fnd', 'make', 'one', 'day', 'think', 'month', 'year', 'maybe', 'week', 'youre', 'going', 'make', 'thing',
                                     'especially', 'u', 'mean', 'hour', 'almost', 'used', 'ect', 'look', 'dont', 'doesnt', 'may', 'etc', 'told', 'another',
                                     'easy', 'right', 'well', 'give', 'age', 'cant', 'lot', 'ive', 'love', 'still', 'dr',
                                     'berg', 'last', 'amazing', 'didnt',
                                     'end', 'many', 'went', 'know', 'take', 'come', 'say', 'come', 'gon
                                     na', 'time', 'video', 'done', 'alway', 'little', 'hi',
                                     'hour', 'true', 'year', 'ago', 'become', 'man', 'even', 'isnt', 'people', 'everything', 'literally', 'without', 'keto',
                                     'really', 'thanks', 'made', 'seem', 'got', 'ate', 'let', 'getting', 'add', 'pretty', 'year', 'old', 'watch', 'ok', 'least',
                                     'found', 'never', 'thought', 'le', 'put', 'idea', 'hard', 'every', 'starting', 'keep', 'person', 'feeling', 'someone', 'god',
                                     'day', 'bad', 'guy', 'trying', 'started', 'understand', 'new', 'anuone', 'able', 'lol', 'due', 'big', 'live', 'two',
                                     'month', 'gon', 'na', 'sometimes', 'best', 'happy', 'least', 'felt', 'daily', 'hole', 'last', 'amazing', 'definitely',
                                     'tried', 'part', 'hope', 'long', 'fast', 'please', 'half', 'kind', 'important', 'reason', 'instead', 'stuff', 'great',
                                     'making', 'away', 'today', 'using', 'tell', 'told', 'week', 'normal', 'something', 'feel', 'month', 'day', 'year', 'years',
                                     'actually', 'added', 'already', 'also', 'always', 'amount', 'anyone', 'anything', 'around', 'appreciate', 'different',
                                     'bit', 'bp', 'bro', 'gave', 'lb', 'need', 'like', 'oat', 'get', 'help', 'pm', 'stop'])))

    stop_words = set(stopwords.words('english'))
    stop_words.update(additional_stopwords)
    tokens = [word for word in tokens if word not in stop_words]

    # Lemmatize words
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    tokens = [lemmatizer.lemmatize(word, pos="v") for word in tokens]

    # Join tokens back into a single string
    preprocessed_text = ' '.join(tokens)

    return preprocessed_text

```

FIG. 5.14 : process function

la fonction **predict** qui va prendre comme entrée un commentaire puis elle va détecter à quel cluster le commentaire appartient :



```
def predict(my_text,tfidf_vect,kmeans,total):  
  
    preprocessed_text = preprocess_text(my_text)  
    text_vector = tfidf_vect.transform([preprocessed_text])  
    predicted_cluster = kmeans.predict(text_vector)  
  
    return int(predicted_cluster)
```

FIG. 5.15 : predict function

la fonction **returnTopics** consiste à déterminer les topics principaux qui dominent dans chaque cluster :



```
def return_topics(tfidf_vect,kmeans,predicted_cluster):  
  
    feature_names = tfidf_vect.get_feature_names_out()  
    centroids = kmeans.cluster_centers_  
    cluster_topics = []  
  
    for i, centroid in enumerate(centroids):  
        top_indices = centroid.argsort()[-5:][:-1]  
        cluster_topics.append([feature_names[ind] for ind in top_indices])  
  
    topics=cluster_topics[predicted_cluster]  
    return topics
```

FIG. 5.16 : returnTopics function

la fonction **chatWithGpt** :

la fonction **extractLinesWithNumbers** :



```
def chat_with_gpt(words):
    # Define the prompt and the system message
    text = f""" i want of 5 social media pages or groups or other virtual communities
    that talk about topics related to this words {words}
    without description but mention if it is a group facebokk or etc """
    prompt = "User: {}\nAI:".format(text)
    # Generate a chat response using the ChatGPT API
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "user", "content": prompt}
        ]
    )
    # Extract and return the chatbot's reply
    chatbot_reply = response['choices'][0]['message']['content']
    return chatbot_reply
```

FIG. 5.17 : chatWithGpt function



```
def extract_lines_with_numbers(text):
    lines_with_numbers = []
    lines = text.split('\n')

    for line in lines:
        if re.match(r'^\d', line): # Check if the line starts with a digit
            lines_with_numbers.append(re.sub(r'^\d+\s*', '', line)) # Remove the leading number

    return lines_with_numbers
```

FIG. 5.18 : extractLinesWithNumbers function

la fonction **returnTopics** :



```
def return_topics(tfidf_vect,kmeans,predicted_cluster):
    feature_names = tfidf_vect.get_feature_names_out()
    centroids = kmeans.cluster_centers_
    cluster_topics = []
    for i, centroid in enumerate(centroids):
        top_indices = centroid.argsort()[-5:][:-1]
        cluster_topics.append([feature_names[ind] for ind in top_indices])
    topics=cluster_topics[predicted_cluster]
    return topics
```

FIG. 5.19 : returnTopics function

la fonction **returnComments** :



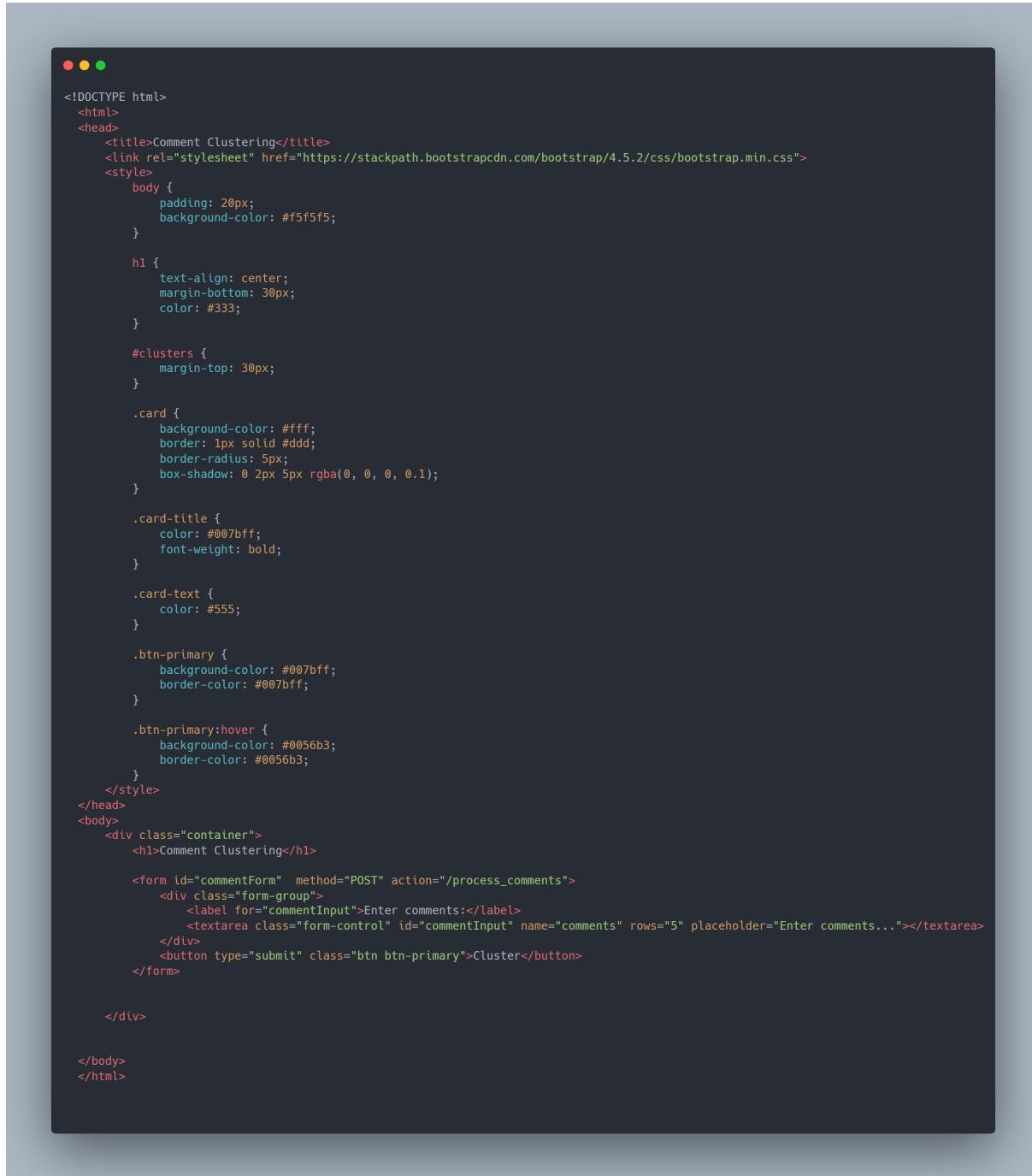
```
def return_comments(my_text,tfidf_vect,kmeans,total):
    predicted_cluster=predict(my_text,tfidf_vect,kmeans,total)
    comments=total[total['cluster']==predicted_cluster]['comment'].sample(n=5).values
    return comments
```

FIG. 5.20 : returnComments function

### 5.2.2.3 création des pages web

lorsque le client va entrer dans la plateforme il va saisir ses preferences puis on va lui recommander la communauté compatible avec les informations donné :

voici notre page home.html



```
<!DOCTYPE html>
<html>
<head>
    <title>Comment Clustering</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        body {
            padding: 20px;
            background-color: #f5f5f5;
        }

        h1 {
            text-align: center;
            margin-bottom: 30px;
            color: #333;
        }

        #clusters {
            margin-top: 30px;
        }

        .card {
            background-color: #fff;
            border: 1px solid #ddd;
            border-radius: 5px;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        }

        .card-title {
            color: #007bff;
            font-weight: bold;
        }

        .card-text {
            color: #555;
        }

        .btn-primary {
            background-color: #007bff;
            border-color: #007bff;
        }

        .btn-primary:hover {
            background-color: #0056b3;
            border-color: #0056b3;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Comment Clustering</h1>

        <form id="commentForm" method="POST" action="/process_comments">
            <div class="form-group">
                <label for="commentInput">Enter comments:</label>
                <textarea class="form-control" id="commentInput" name="comments" rows="5" placeholder="Enter comments... "></textarea>
            </div>
            <button type="submit" class="btn btn-primary">Cluster</button>
        </form>
    </div>
</body>
</html>
```

FIG. 5.21 : home.html

voici notre page results.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Comment Clustering Results</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f2f2f2;
            color: #333333;
            margin: 0;
            padding: 20px;
        }

        h1, h2 {
            color: #0056b3;
        }

        ul {
            list-style-type: none;
            padding: 0;
        }

        .cluster-number {
            display: inline-block;
            width: 40px;
            height: 40px;
            border-radius: 50%;
            background-color: #0056b3;
            color: #ffffff;
            text-align: center;
            line-height: 40px;
            font-size: 20px;
            margin-right: 10px;
        }

        table {
            width: 100%;
            border-collapse: collapse;
            margin-top: 20px;
        }

        th, td {
            padding: 10px;
            text-align: left;
            border-bottom: 1px solid #dddddd;
        }

        th {
            background-color: #0056b3;
            color: #ffffff;
        }

        .comment-row:nth-child(even) {
            background-color: #f2f2f2;
        }

        .comment-row:hover {
            background-color: #e6e6e6;
        }

        .link-row:nth-child(even) {
            background-color: #f2f2f2;
        }

        .link-row:hover {
            background-color: #e6e6e6;
        }

        .tables-container {
            display: flex;
            justify-content: space-between;
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <h1>welcome to your community</h1>
    <h2>Number of Clusters: <span class="cluster-number">{{ cluster_number }}</span></h2>

    <h2>Comments:</h2>
    <table>
        <thead>
            <tr>
                <th>Comment</th>
            </tr>
        </thead>
        <tbody>
            {% for comment in comments %}
            <tr class="comment-row">
                <td>{{ comment }}</td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
    <h2>Links:</h2>
    <table>
        <thead>
            <tr>
                <th>link</th>
            </tr>
        </thead>
        <tbody>
            {% for link in links %}
            <tr class="link-row">
                <td>{{ link }}</td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
</body>
</html>
```

FIG. 5.22 : results.html

#### 5.2.2.4 Démarrage du serveur

apres la configuration de l'environnement virtuel on va executer les commandes suivantes : **flask run**

```
C:\Users\tNouali\AppData\Roaming\nltk_data...
[nltk_data]  Package wordnet is already up-to-date!
* Serving Flask app 'application.py'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

FIG. 5.23 : running server

puis on va aller sur localhost :5000

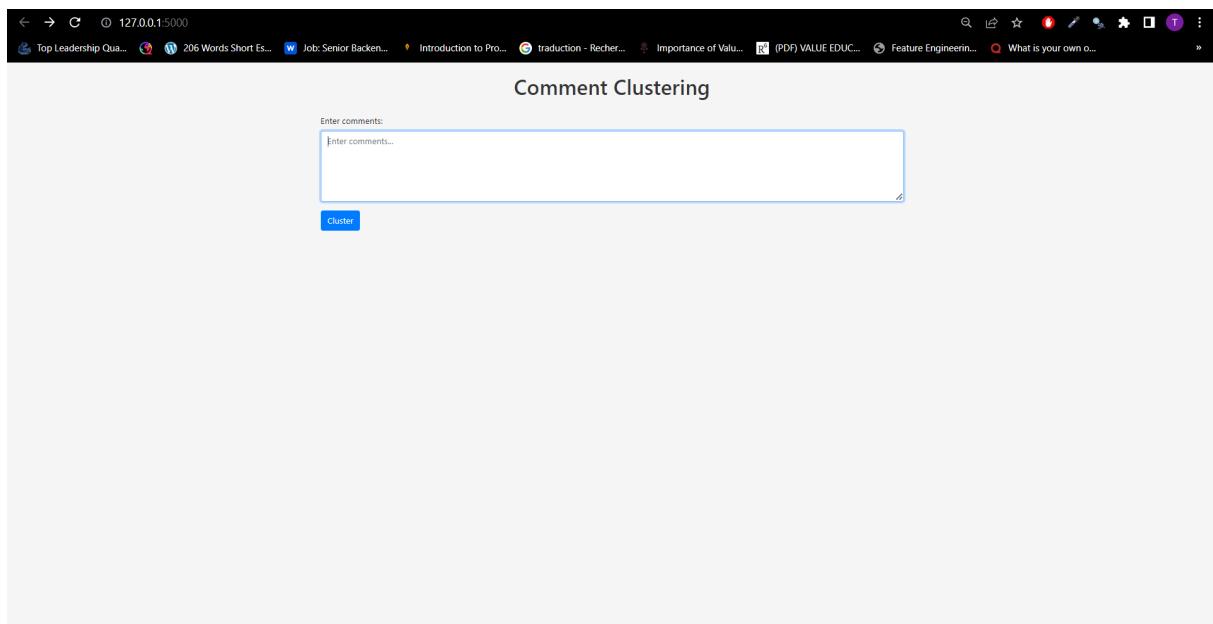


FIG. 5.24 : page(1)

puis on va taper nos préférences :

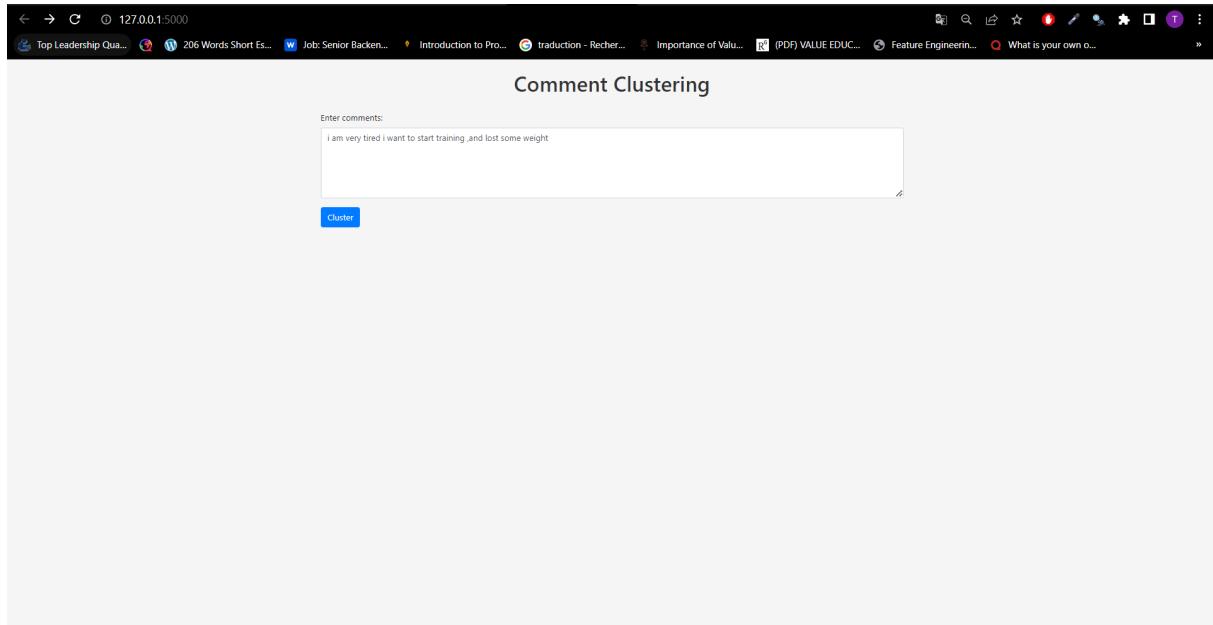


FIG. 5.25 : page(2)

puis le résultat va être affiché

A screenshot of a web page titled "welcome to your community". It displays a section for "Comments" with a count of "Number of Clusters: 40". Below this, there are four collapsed sections, each representing a cluster. The first section contains a single comment: "2100 calories work if you are really training otherwise that is too high for a woman". The second section contains a comment: "finally someone who did a video for intermediate exercises and explains it to understand without confusion after watching your video i immediately started my training thank you". The third section contains a comment: "weightstrength training causes you to burn more calories at rest". The fourth section contains a comment: "question is riding a bicycle a good active rest exercise does it train all the required muscles as well". Below the comments, there is a section titled "links:" with a link to "Train Hard/Lift Heavy - Facebook group".

FIG. 5.26 : page(3)

# Conclusion

En conclusion, la recommandation de communautés virtuelles externes constitue une étape cruciale dans notre projet de détection des communautés virtuelles. En collectant des données à partir de différentes plateformes de médias sociaux et en appliquant des techniques de clustering, nous avons pu identifier des communautés internes correspondant aux intérêts des utilisateurs. Cependant, pour offrir une expérience utilisateur complète, il est tout aussi important de recommander des communautés externes pertinentes. En utilisant des approches avancées basées sur l'analyse sémantique, l'apprentissage automatique et l'API ChatGPT, nous sommes en mesure de fournir des suggestions personnalisées qui favorisent l'engagement et l'interaction des utilisateurs avec des communautés virtuelles externes. Grâce au déploiement de notre modèle avec Flask, nous avons créé une plateforme conviviale permettant aux utilisateurs d'explorer et de rejoindre des communautés qui correspondent à leurs préférences et à leurs intérêts spécifiques. Cette approche enrichit l'expérience de nos utilisateurs en les connectant à des communautés virtuelles pertinentes, favorisant ainsi un engagement continu et une satisfaction accrue.

# CONCLUSION

Dans ce projet, nous avons utilisé des techniques de scraping sur YouTube, Reddit et Instagram pour collecter des commentaires. Ensuite, nous avons utilisé l'intelligence artificielle (clustering) pour regrouper les utilisateurs dans des communautés spécifiques en fonction de leurs profils. L'utilisation de techniques de clustering nous a permis de recommander à chaque utilisateur une communauté interne qui correspond à ses intérêts et objectifs. De plus, nous avons également intégré la possibilité d'ajouter des recommandations externes pour enrichir l'expérience des utilisateurs.

Ce projet va permettre à l'entreprise Synergeon de créer une plateforme dynamique où les utilisateurs peuvent partager leurs connaissances, se motiver mutuellement et élargir leurs réseaux dans le domaine du bien-être. En encourageant la collaboration et l'échange d'expériences, nous avons pu offrir aux utilisateurs une meilleure compréhension de tous les aspects liés à leur bien-être.

En conclusion, ce projet démontre comment l'utilisation du scraping de données combinée à l'intelligence artificielle peut être bénéfique pour créer des communautés virtuelles engagées dans le domaine du fitness et du bien-être. La mise en réseau des utilisateurs, l'échange de connaissances et la recommandation de communautés pertinentes contribuent à une expérience enrichissante pour chaque utilisateur. À l'avenir, nous espérons développer davantage ce projet en intégrant de nouvelles sources de données et en améliorant les fonctionnalités de recommandation pour offrir une expérience personnalisée encore plus poussée.