

Projet1 : Analyse de Sentiment des Avis Clients avec Résumé Automatique

📌 **Objectif** : Développer une application capable de scraper des avis clients depuis un site e-commerce, analyser leur sentiment (**positif, neutre, négatif**) avec un **LLM** et générer un résumé automatique des avis.

🔧 Technologies et Concepts Abordés :

- ✅ **Web Scraping** : Récupération des avis clients avec **BeautifulSoup, Scrapy ou Playwright**.
 - ✅ **Analyse de Sentiment** : Utilisation d'un **LLM (GPT-4, Gemini, Mistral, DeepSeek)** ou d'un modèle comme **DistilBERT, Vader, TextBlob**.
 - ✅ **Génération de Résumé** : Utilisation d'un **LLM API (OpenAI, Gemini, DeepSeek, Hugging Face)** ou d'un modèle extractif/abstractive.
 - ✅ **API REST** : Développement avec **Flask ou FastAPI** pour exposer le service.
 - ✅ **Visualisation** : Interface web simple avec **Streamlit ou Dash/Gradio** pour afficher les résultats.
-

📌 Étapes du Projet :

1. Web Scraping des Avis Clients

- Cible : Sites comme **Amazon, Trustpilot, Yelp** (ou dataset public).
- Extraction des **noms des produits, notes, commentaires**.

2. Prétraitement des Données

- Nettoyage du texte (HTML, emojis, stopwords).

3. Stockage en bases données vectorielle

- Embedding des données et stockage avec **faiss** ou **Chromedb, Weaviate, OpenSearch**

4. Analyse de Sentiment

- Utilisation d'un modèle **pré-entraîné (DistilBERT, Vader, LLM API)**.
- Classification des avis : **Positif / Neutre / Négatif**.

5. Génération d'un Résumé Automatique

- **Méthode extractive** (ex: **TextRank, BART**).
- **Méthode abstractive** via **GPT-4, Gemini, Mistral**.

6. Déploiement d'une API REST

- Création d'un endpoint `/analyze_reviews` prenant en entrée un **produit** et retournant :
 - ✓ Répartition des sentiments (**graphique**)
 - ✓ Un **résumé des avis**

7. Interface Web (optionnelle)

- **Streamlit / Dash** pour afficher l'analyse en temps réel.

8. Conteneurisation avec Docker et Docker Compose

◆ Variantes et Améliorations :

- **Ajout de tendances** : comparaison des avis sur plusieurs mois.
- **Analyse multi-langues** avec des modèles comme **mBERT**.
- **Personnalisation** : L'utilisateur peut choisir **le nombre d'avis à scraper**.
- **Connexion à une base de données** pour stocker les avis analysés.

🎯 Résultat attendu :

Une application qui prend un **produit ou une URL**, analyse le **sentiment des avis**, et génère un **résumé clair et concis** des retours clients.

Projet 2 : Générateur de Résumés et d'Insights à partir d'Articles Web

📌 **Objectif** : Développer une application qui récupère des articles depuis le web, extrait les informations pertinentes et génère un résumé enrichi à l'aide d'un **LLM**.

🔧 Technologies et concepts abordés :


- ✓ **Web Scraping** : Utilisation de **BeautifulSoup**, **Scrapy** ou **Playwright** pour extraire les articles d'un site d'actualités.
- ✓ **API** : Utilisation d'une API LLM (ex : **OpenAI GPT**, **Gemini API**, **Mistral AI**) pour la génération de résumés.
- ✓ **LLMs** : Fine-tuning ou utilisation d'un modèle **open-source (Llama 3, Mistral, Falcon)** avec **Hugging Face** pour améliorer la pertinence du résumé.

◆ Étapes du projet :

1. **Scraping d'articles** depuis un site d'actualités (**Le Monde, BBC, TechCrunch...**).
2. **Nettoyage et prétraitement** du texte (suppression des balises HTML, normalisation).
3. **Les indexer dans une base données vectorielles (faiss , chromebd, weviate, OpenSearch**
4. **Appel à une API LLM** pour générer un résumé concis et pertinent.
5. **Ajout d'analyses complémentaires** (ex : extraction des mots-clés avec un modèle d'embeddings).
6. **Déploiement d'une API Flask ou FastAPI** pour exposer le service.
7. **Interface Web (optionnelle)** avec **Streamlit ou Gradio/Dash** pour afficher les résumés.
8. **Déploiement et conteneurisation avec Docker**

◆ **Variantes et améliorations possibles :**

- Ajouter une **analyse du sentiment** des articles.
- Comparer les résumés générés par **différents LLMs**.
- Permettre aux utilisateurs de **fournir un lien vers un article** et obtenir un résumé en temps réel.

 **Résultat attendu :** Une API qui prend une URL d'article en entrée et retourne un résumé généré par un LLM.