

Objectifs du projet :

Créer une application avec FastAPI permettant de recevoir des messages textuels, puis grâce à un modèle d'analyse déterminer si la connotation du message est plutôt positive ou négative. Pour les données à analyser, on utilisera un dataset trouvé en ligne et on utilisera ELK pour stocker et analyser les résultats.

Livrables :

Lien du dépôt github : <https://github.com/LBnst/sentiments/tree/main>

Fichiers trouvables sur le github :

- docker-compose.yml : fichier de paramétrage/lancement de ma stack ELK
- dashboard_sentiments.ndjson : fichier export du dashboard créé sur ELK (capture de celui-ci plus bas)
- twitter_training.csv : dataset d'origine non modifié/nettoyé pris sur internet (lien : <https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis>)
- clean_dataset.csv : dataset nettoyé et prêt à l'emploi pour mon app fastAPI
- prepare_dataset.py : script de nettoyage du dataset
- requirements.txt : liste des packages à installer pour le fonctionnement de mon app
- main.py : structure et définition de mon app d'analyse de sentiments
- model.py : script d'analyse des messages
- es.py : indexation des messages analysés dans ma stack ELK
- Suivi projet.xlsx : excel contenant le suivi et les problèmes de mon projet

Modèle d'analyse utilisé : <https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english>

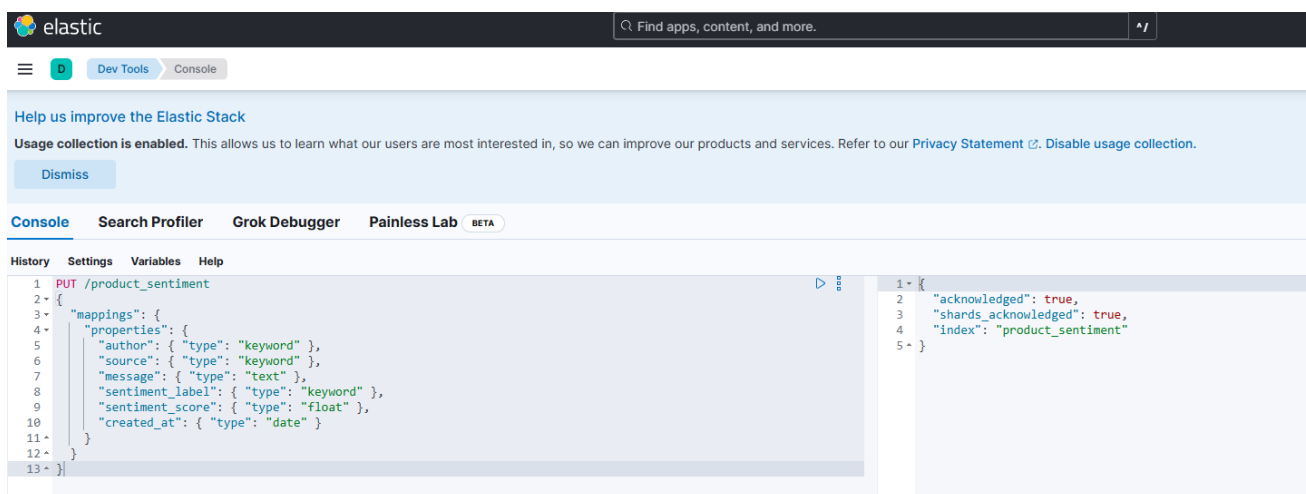
Déroulement du projet

La première étape a été de comprendre l'objectif et de préparer les outils pour atteindre ce dernier.

- Pour le dataset j'ai privilégié le premier de la liste (Twitter Sentiment Analysis sur kaggle) car j'étais déjà habitué à l'utilisation de kaggle.
- Docker et VScode étaient tous deux déjà installés sur mon ordinateur. Pour dev, j'ai donc utilisé VScode et mon powershell windows.
- En raison de mon alternance et autres occupations externes, j'ai préféré éviter de créer un planning à l'avance par crainte de ne pas être capable de le respecter. J'ai préféré créer un tableau excel pour prendre note de mon avancée et garder une trace des problèmes rencontrés.
- J'ai donc privilégié d'avancer un maximum sur le projet dès lors que j'avais du temps libre disponible.

J'ai commencé par créer ma stack ELK.

Cette étape a été plutôt rapide étant donné que j'ai eu à faire le même exercice lors d'un projet précédent. J'ai pu réutiliser la même structure pour ma stack et mes connaissances encore récentes sur le sujet.



La seconde étape a été de créer mon app d'analyse de sentiment.

J'ai souhaité laisser mon dataset de côté pour le moment pour privilégier avant des tests manuels.

J'ai donc utilisé FastAPI pour créer une app d'analyse locale. Grâce à transformers et un modèle trouvé sur Hugging Face, il me suffisait de rentrer un message en anglais et le modèle déterminait si le message est plutôt positif ou négatif, et attribue un score de précision entre 0 et 1 à sa prédiction.

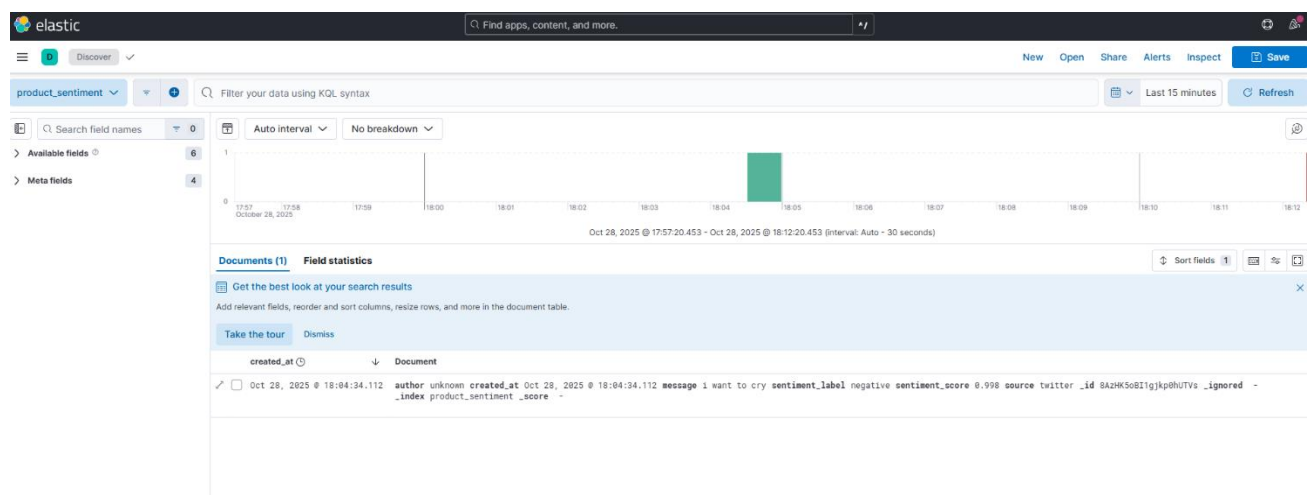
Pour que mes valeurs remplissent correctement mon index sur ELK, j'y ai rajouté un champ pour l'utilisateur et la plateforme sur laquelle l'utilisateur a posté le message. Le champ « date » étant non présent dans le dataset originel, je me suis permis de le remplacer par la date à laquelle le message a été analysé par mon app.

Ne reste plus qu'indexer mes 6 champs dans « product_sentiment ».

Problème rencontré : ma version d'elasticsearch était apparemment trop récente. J'ai alors dû désinstaller elasticsearch pour réinstaller une version antérieure.

```
(venv) PS C:\Users\Lucien\Desktop\Rattrapages 2025\app> uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['C:\\Users\\Lucien\\Desktop\\Rattrapages 2025\\app']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [20520] using StatReload
Device set to use cpu
✗ Erreur de connexion à Elasticsearch : BadRequestError(400, 'media_type_header_exception', 'Invalid media-type value on headers [Content-Type, Accept]', Accept version must be either version 8 or 7, but found 9. Accept=application/vnd.elasticsearch+json; compatible-with=9)
INFO: Started server process [26724]
INFO: Waiting for application startup.
INFO: Application startup complete.
```

Une fois le problème résolu, l'indexation des données fonctionnait et mes entrées dans elastic ressemblaient à ça :



Les tests étant concluants, j'ai donc préparé mon dataset :

- Seule les colonnes « message » et « source » ont été laissées. Une colonne prédisant l'humeur du message était déjà présente mais le but étant de les analyser moi-même, je ne l'ai pas gardée.
- Nouvelle colonnes « author ». Puisque l'auteur n'était pas indiqué dans le dataset, elle a été manuellement remplie avec « unknown ».

Mon app a été modifiée pour rajouter un script qui effectue le même processus mais sur l'intégralité de mon dataset nettoyé.

Problème rencontré : certaines données du dataset faisaient crash mon app car la colonne « message » était vide. J'ai alors repris mon script de nettoyage pour retirer les champs

vides, les champs contenant moins de 2 mots ou ne contenant pas de lettres.

```
INFO: 127.0.0.1:58567 - "POST /bulk_import HTTP/1.1" 500 Internal Server Error
ERROR: Exception in ASGI application
Traceback (most recent call last):
  File "C:\Users\Lucien\Desktop\Ratrapages 2025\app\venv\Lib\site-packages\uvicorn\protocols\http\h11_impl.py", line 403, in run_asgi
    result = await app(  # type: ignore[func-returns-value]
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    self.scope, self.receive, self.send
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

2nd problème : je n'ai pas anticipé que mon script pourrait prendre du temps à analyser mon dataset (71000 champs à analyser). Lorsque je me suis rendu compte de mon erreur, j'ai tenté de couper manuellement mon app mais cette dernière voulait aller au bout de ma requête avant son arrêt. J'ai donc dû couper manuellement python dans mon powershell pour la stopper.

```
(venv) PS C:\Users\Lucien\Desktop\Rattrapages 2025\app> tasklist | findstr python
python.exe                15844 Console                1      3?564 Ko
python.exe                10212 Console                1      44?868 Ko
python.exe                 6516 Console                1      635?072 Ko
(venv) PS C:\Users\Lucien\Desktop\Rattrapages 2025\app> taskkill /PID 6516 /F
Opération réussie : le processus avec PID 6516 a été terminé.
(venv) PS C:\Users\Lucien\Desktop\Rattrapages 2025\app> |
```

Dernière étape : créer mon dashboard kibana

Je savais déjà comment créer une stack ELK mais je ne savais pas comment faire pour créer un dashboard dessus, et son fonctionnement n'est pas très clair. J'ai donc réappris de 0 en regardant un tutoriel d'utilisation.

Je me suis également rendu compte d'un problème : le nombre d'analyses pertinentes possibles allait être restreint en raison de l'absence de diversité dans les champs `author` et `date`.

Le champ date peut-être intéressant dans un cas où des tweets sont régulièrement importés sur mon app, mais dans mon cas il présente peu d'intérêt.

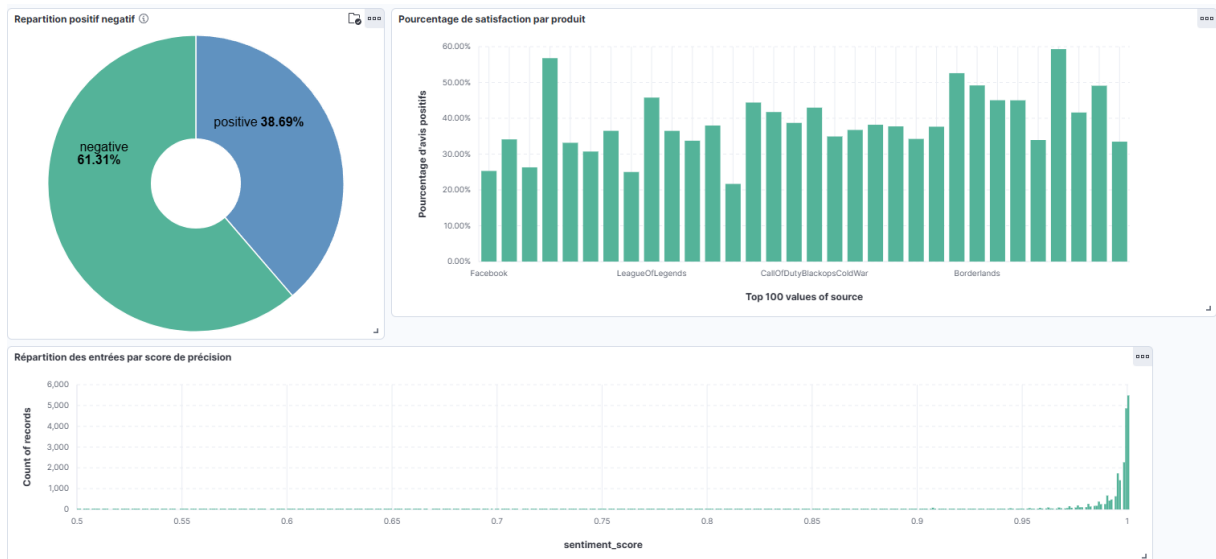
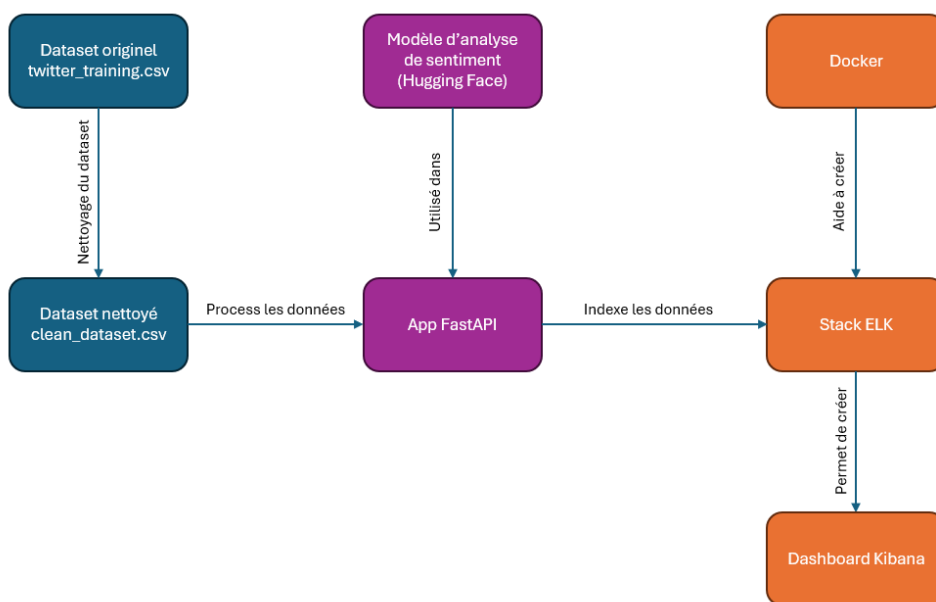


Schéma du pipeline complet :



Problèmes rencontrés :

En plus des problèmes évoqués plus haut, mon plus gros problème a été de réaliser à la fin du projet que l'objectif initial (comparer le taux de positivité par rapport a différents produits) n'avait pas été respecté. J'ai donc vidé mon index, repris mon script de nettoyage du dataset car j'avais au départ rempli la colonne « source » avec « twitter », et j'en ai profité pour réduire sa taille par 3 pour éviter de reproduire le problème lors de l'indexation. Puis j'ai modifié le dashboard pour rajouter le % de satisfaction par produit.