Rapport : projet de veille sur le sentiment public autour d’un produit technologique

1. Cadrage du projet

Objectif

Ce projet visait à concevoir un système complet de surveillance du sentiment en ligne, concernant un produit technologique. L’idée principale était de pouvoir analyser automatiquement les avis des internautes, d’en déterminer la tonalité(positive, neutre ou négative) et d’afficher les tendances sous forme de graphiques interactifs.

Périmètre

La solution repose sur l’écosystème ELK (Elasticsearch, Logstash, Kibana), associé à une API d’analyse développée en Python avec FastAPI.

Le prototype comprend :

- l’analyse automatique du texte grâce à un modèle d’IA,
- le stockage structuré des résultats dans Elasticsearch,
- la visualisation des données dans Kibana,
- ainsi qu’une documentation complète accompagnée d’une gestion de projet agile.

Livrables

- Une stack Docker-ELK fonctionnelle (Elasticsearch + Kibana + API FastAPI)
- Le dataset complet
- Un dashboard Kibana intitulé *“Sentiment Monitoring Dashboard”*
- L’export du dashboard au format `.ndjson`

- Le rapport de projet

Méthode de travail

Le projet a été mené en suivant une approche agile simplifiée.

Chaque étape de la configuration de l'infrastructure à la création du dashboard a été réalisée de manière incrémentale et validée avant de passer à la suivante.

Cette méthode a permis d'assurer un avancement fluide, tout en gardant une vision claire des priorités.

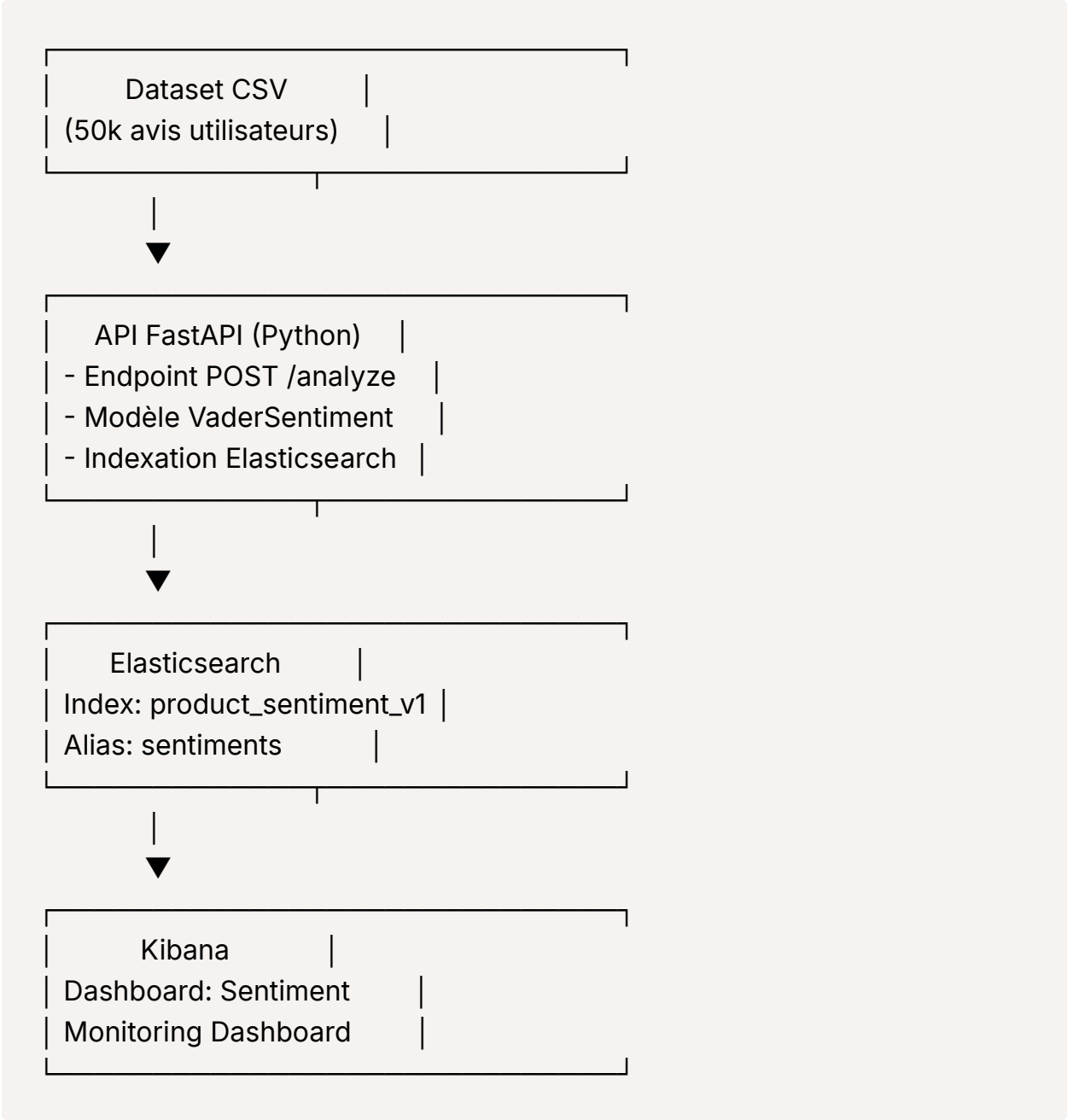
2. Planification du projet

Tâche	Description	Priorité	Durée estimée	Statut
T1	Installation et configuration de la stack Docker-ELK	Haute	0,5 jour	Terminé
T2	Préparation et importation du dataset CSV	Haute	0,5 jour	Terminé
T3	Développement du service d'analyse IA avec FastAPI et Vader	Haute	1 jour	Terminé
T4	Tests d'intégration et indexation dans Elasticsearch	Moyenne	0,5 jour	Terminé
T5	Conception du dashboard Kibana	Haute	1 jour	Terminé
T6	Rédaction du rapport et synthèse finale	Moyenne	0,5 jour	Terminé

Durée totale d'environ 4 jours pour un prototype complet, documenté et fonctionnel.

3. Architecture technique

Le pipeline mis en place repose sur une architecture modulaire claire et efficace :

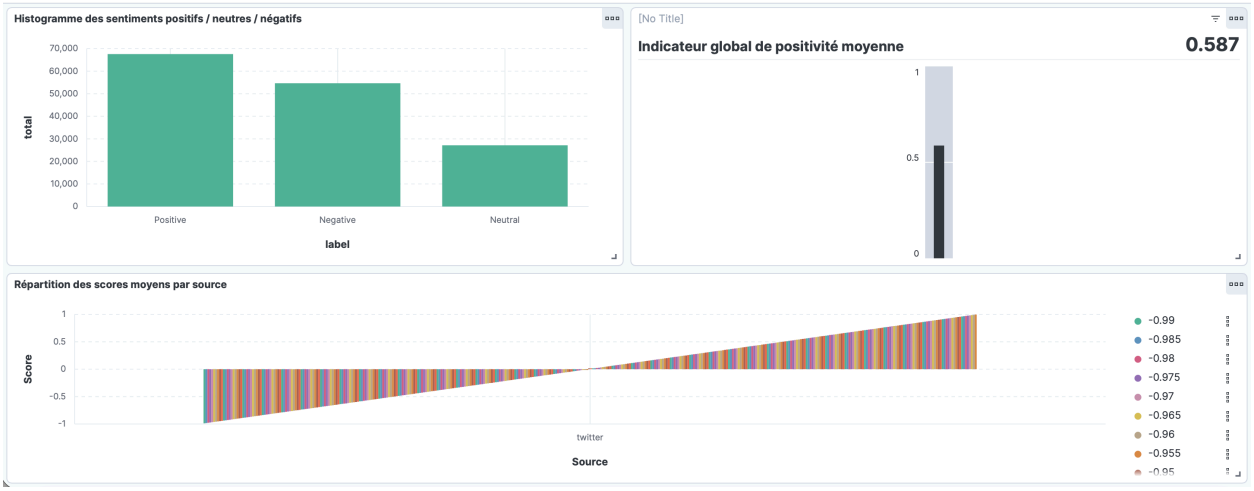


Technologies clés

- FastAPI (Python 3.11) : service d’analyse textuelle en ligne
- VADER : modèle de détection de sentiments léger et performant
- Elasticsearch 8.14 : moteur de recherche et stockage structuré des données
- Kibana 8.14 : visualisation des résultats sous forme de dashboard

- Docker Compose : orchestration des services
- Dataset source : `twitter_training.csv`

4. Résultats obtenus



Indicateur	Observation	Interprétation
Nombre total d'avis	≈ 150 000 cumulés	Volume conséquent pour une analyse fiable
Sentiment dominant	Positif (≈ 45–50 %)	Bonne image générale du produit
Score moyen global	0.587	Satisfaction plutôt stable
Écart entre positif et négatif	≈ 15 000 avis	Opinion polarisée (amour ou rejet)
Sources analysées	Principalement Twitter	Représente le ressenti en ligne spontané

Export du dashboard

- Nom : *Sentiment Monitoring Dashboard v2*
- Format : `.ndjson`
- Localisation : `kibana/sentiment_monitoring_dashboard.ndjson`

5. Rétrospective du projet

Difficultés rencontrées

- Ordre de démarrage des conteneurs : l'API FastAPI se lançait avant Elasticsearch, provoquant des erreurs de connexion. Problème résolu en ajoutant un délai d'attente dans le Dockerfile.
- Problèmes d'import de modules Python : erreur `ModuleNotFoundError` corrigée grâce à l'ajout du chemin du projet dans `sys.path`.
- Apprentissage de Kibana Lens : la découverte de l'outil a nécessité quelques essais avant de maîtriser la création et la mise en forme des visualisations.

Choix et arbitrages techniques

- Adoption du modèle VADER, privilégié pour sa légèreté et sa simplicité d'intégration.
- Utilisation de FastAPI pour sa rapidité et sa compatibilité avec Elasticsearch.
- Architecture modulaire, facilitant la maintenance et l'ajout futur de fonctionnalités.

Pistes d'amélioration

- Intégrer un modèle transformer (Hugging Face) pour améliorer la précision de l'analyse.
- Connecter la solution à un flux en temps réel (API Twitter/X) pour suivre les tendances instantanément.
- Héberger le système sur une infrastructure cloud (Elastic Cloud, AWS, Azure).
- Mettre en place un système d'alerte automatique en cas de hausse des avis négatifs.