

Encadre par :

Dr. EL MADANI EL ALAMI Yasser

**Rapport de Projet BigDatA**

**Architecture Big Data pour l'Analyse en Temps Réel :** Étude de la perception médiatique du remplacement de l'homme par l'IA via News API.

Réalisé par :

Saad ESSFALI

Soufiane NHAILA

Amine BOUANANI

Pape MAGUETTE

Table des matières

[Chapitre 1 Contribution : Collecte et Ingestion des Données en Temps Réel 1](#_Toc219915669)

[1 ) Stratégie d'Acquisition (NewsAPI) 1](#_Toc219915670)

[2 ) Architecture du Pipeline de Streaming Kafka 1](#_Toc219915671)

[3 ) Stockage Persistant dans MongoDB 3](#_Toc219915672)

[4 ) Tests et Validation du Pipeline 3](#_Toc219915673)

[Chapitre 2 Nettoyage & Prétraitement des Données 4](#_Toc219915674)

[1 ) Objectifs du chapiter 4](#_Toc219915675)

[2 ) Architecture Générale du Pipeline 4](#_Toc219915676)

[2.1) Vue d’ensemble 4](#_Toc219915677)

[Flux de traitementent 5](#_Toc219915678)

[3 ) Nettoyage des Données Textuelles 5](#_Toc219915679)

[3.1) Objectifs du nettoyage 5](#_Toc219915680)

[3.2) Module cleaner.py 5](#_Toc219915681)

[4 ) Normalisation des Textes 6](#_Toc219915682)

[4.1) Rôle de la normalization 6](#_Toc219915683)

[4.2) Module normalizer.py 6](#_Toc219915684)

[5 ) Prétraitement NLP 7](#_Toc219915685)

[5.1) Techniques NLP appliquées 7](#_Toc219915686)

[5.2) Module nlp\_processor.py 8](#_Toc219915687)

[5.3) Pipeline Principal de Prétraitement 8](#_Toc219915688)

[6 ) Orchestration avec Apache Airflow 9](#_Toc219915689)

[6.1) Présentation de l'Orchestration 9](#_Toc219915690)

[6.2) Configuration du DAG 10](#_Toc219915691)

[6.3) Structure du Workflow 10](#_Toc219915692)

[Chapitre 3 Sentiment & Stance Analysis Component 12](#_Toc219915693)

[1 ) Objectifs du Module ML 12](#_Toc219915694)

[2 ) Préparation des Données et Stratégie d'Étiquetage 13](#_Toc219915695)

[2.1) Extraction et Ingénierie des Caractéristiques (Feature Engineering) 13](#_Toc219915696)

[2.2) Labellisation Automatisée via l'API Gemini (Silver Labeling) 13](#_Toc219915697)

[2.3) Gestion du Déséquilibre des Classes (Oversampling) 14](#_Toc219915698)

[2.4) Division du Dataset 14](#_Toc219915699)

[3 ) Architecture Technique du Modèle 15](#_Toc219915700)

[3.1) Framework SetFit (Sentence Fine-Tuning) 15](#_Toc219915701)

[3.2) Modèle de Base : MPNet-base-v2 16](#_Toc219915702)

[3.3) Mécanisme d'Entraînement en Deux Étapes 16](#_Toc219915703)

[3.4) Environnement de Calcul et Hyperparamètres 17](#_Toc219915704)

[4 ) Analyse des Résultats et Évaluation des Performances 18](#_Toc219915705)

[4.1) Analyse Globale des Performances 18](#_Toc219915706)

[4.2) . Analyse détaillée par classe 19](#_Toc219915707)

[4.3) Interprétation de la Matrice de Confusion 20](#_Toc219915708)

[4.4) Performance d'Inférence et Intégration Pipeline 20](#_Toc219915709)

[4.5) Conclusion sur le Module de Machine Learning 20](#_Toc219915710)

[Chapitre 4 Social Network Analysis Dashboard 21](#_Toc219915711)

[1 ) Objectif du chapiter 21](#_Toc219915712)

[2 ) Approche méthodologique 21](#_Toc219915713)

[3 ) Résultats et interpretation 21](#_Toc219915714)

[Conclusion 23](#_Toc219915715)

Introduction générale

Avec la croissance exponentielle des médias numériques, les plateformes d’actualités en ligne produisent en continu d’importants volumes de données textuelles couvrant une grande diversité de sujets. Ces données constituent une source stratégique pour analyser les tendances médiatiques, les opinions exprimées, l’évolution des thématiques émergentes et l’impact de certains sujets sur l’actualité mondiale. Cependant, leur exploitation efficace soulève des défis majeurs liés au volume, à la vélocité et à la complexité des données.

Ce projet s’inscrit dans une approche Big Data de bout en bout, visant à concevoir et mettre en œuvre une architecture complète permettant la collecte automatisée et quasi temps réel des données d’actualité via NewsAPI (newsapi.org), leur stockage, leur traitement, puis leur analyse avancée. L’architecture repose sur des technologies modernes telles que Kafka, MongoDB, Spark, Airflow et des techniques de Traitement Automatique du Langage Naturel (NLP) et de Machine Learning.

Les données collectées portent sur des thématiques spécifiques telles que l’intelligence artificielle, l’automatisation, l’emploi et l’évolution du travail, permettant ainsi d’extraire des informations pertinentes à partir de contenus textuels non structurés. Après une phase de nettoyage et de prétraitement (suppression du bruit, normalisation, tokenisation, lemmatisation), les données sont exploitées pour réaliser des analyses de sentiment, de la modélisation de sujets (topic modeling), ainsi que l’étude de l’engagement et de la fréquence des thématiques dans l’actualité.

Enfin, les résultats sont agrégés et visualisés à travers des tableaux de bord interactifs et dynamiques, offrant une vision synthétique et exploitable des tendances médiatiques observées. Ce projet illustre ainsi l’intégration cohérente des concepts de Data Streaming, Big Data Analytics, NLP, Machine Learning et visualisation décisionnelle, tout en mettant en valeur une organisation collaborative structurée autour de missions clairement définie

# Contribution : Collecte et Ingestion des Données en Temps Réel

Cette section détaille la partie que j'ai réalisée, portant sur la mise en place de la chaîne d'acquisition des données, depuis l'extraction jusqu'au stockage brut.

## Stratégie d'Acquisition (NewsAPI)

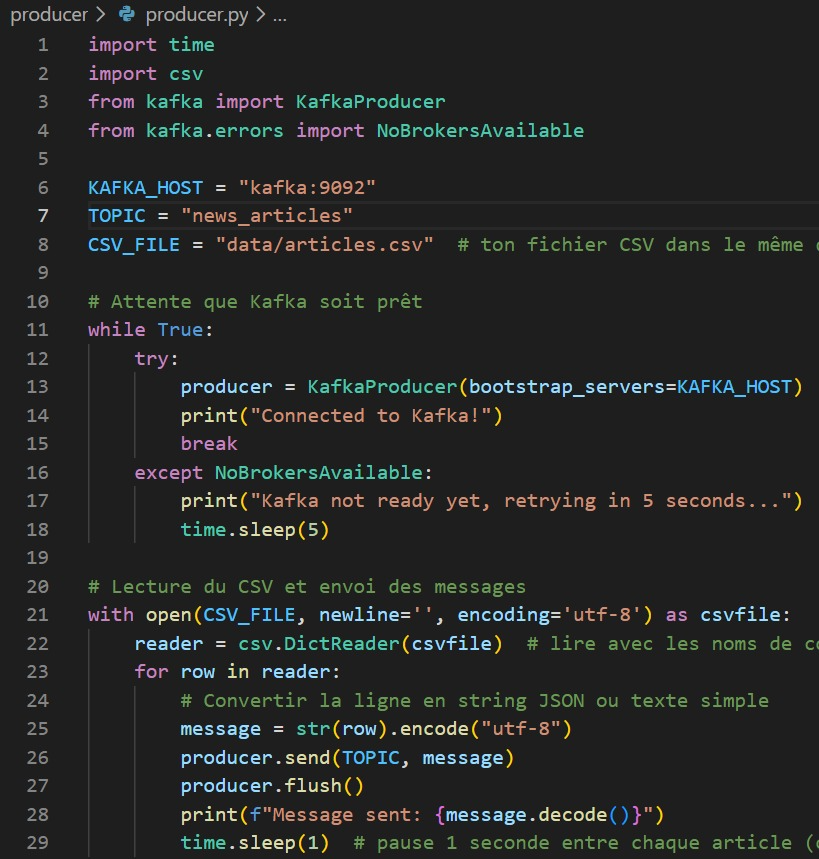
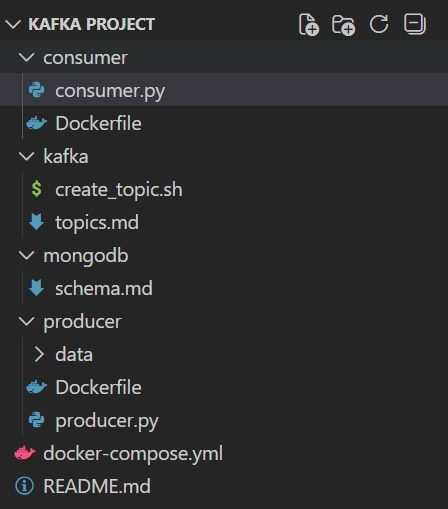
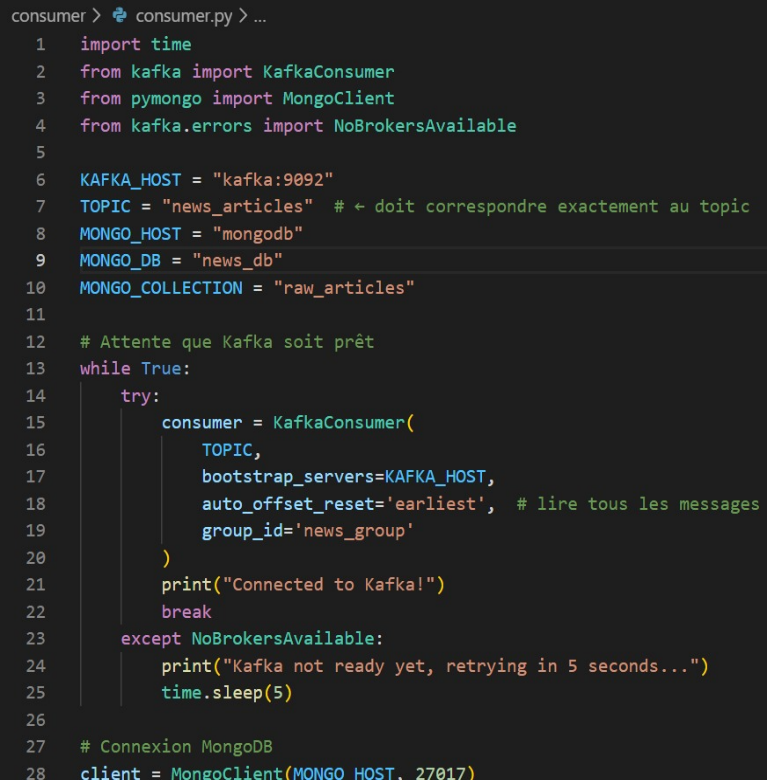
Pour alimenter notre analyse, j'ai développé un script de collecte automatisé utilisant l'interface NewsAPI.

* Ciblage thématique : J'ai sélectionné des mots-clés spécifiques tels que "AI jobs", "automation", et "future of work" pour filtrer les articles pertinents .
* Période de collecte : Le script est configuré pour extraire les données sur une fenêtre glissante de 30 jours.
* Format de sortie : Les données sont d'abord structurées dans un fichier CSV (ai\_vs\_human\_news.csv) pour simuler un jeu de données source avant l'ingestion dans le pipeline.

## Architecture du Pipeline de Streaming Kafka

Afin de garantir un traitement fluide et découplé, j'ai mis en place un pipeline de streaming utilisant Apache Kafka.

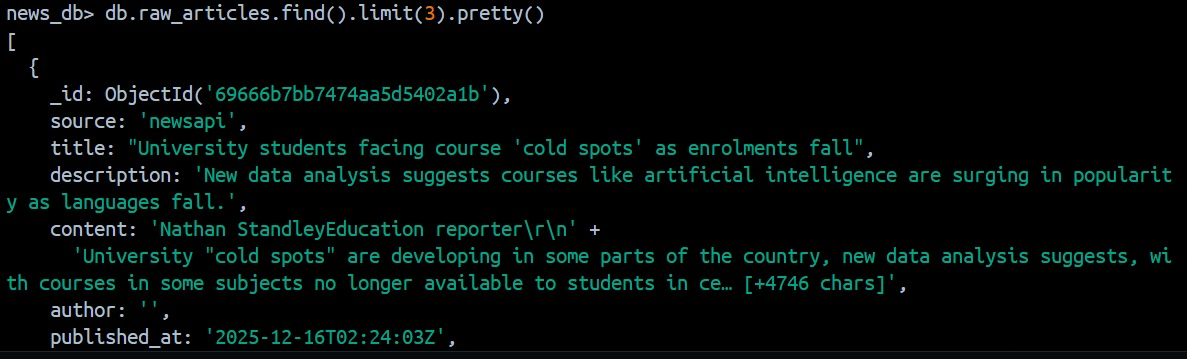
* Le Producer (producer.py) : Ce script lit le fichier de collecte et publie chaque article dans le topic Kafka nommé news\_articles. Pour simuler un flux réel, j'ai intégré un délai de 0,5 seconde entre chaque envoi.
* Le Consumer (consumer.py) : Il écoute le flux en continu, récupère les messages en temps réel et assure leur transition vers la base de données.



## Stockage Persistant dans MongoDB

Pour le stockage de la donnée brute, j'ai opté pour MongoDB en raison de sa flexibilité avec les documents JSON.

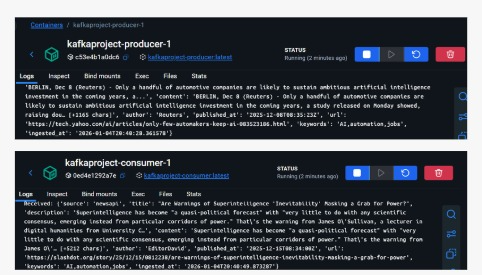
* Structure : Les données sont envoyées vers la collection raw\_articles de la base news\_db.
* Schéma des documents : Chaque entrée inclut des champs essentiels comme le titre, le contenu, l'auteur, l'URL, ainsi qu'un horodatage d'ingestion (ingested\_at) pour la traçabilité.



## Tests et Validation du Pipeline

Pour valider l'intégrité de ma partie, j'ai réalisé plusieurs tests via Docker:

* Validation des flux : Surveillance des logs des conteneurs producer et consumer pour confirmer la transmission des messages.
* Audit de base de données : Vérification du comptage et du contenu des documents via le shell mongosh directement dans le conteneur MongoDB.



# Nettoyage & Prétraitement des Données

## Objectifs du chapiter

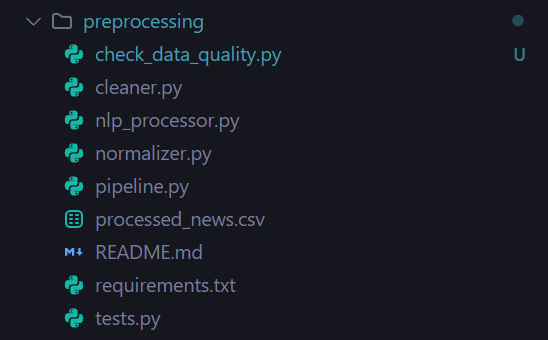
Dans le cadre du projet Big Data portant sur l’analyse de données textuelles, une phase approfondie de nettoyage et de prétraitement constitue une étape clé du cycle de traitement des données. Les données textuelles issues de sources réelles sont souvent hétérogènes, bruitées et non structurées, ce qui peut fortement dégrader la qualité des analyses statistiques et les performances des modèles de Machine Learning.

Cette contribution s’inscrit dans une logique de préparation rigoureuse des données. Elle consiste à concevoir, implémenter et automatiser un pipeline complet de prétraitement des données textuelles, depuis les données brutes jusqu’à la production de datasets propres, normalisés et directement exploitables. Ce travail vise à garantir la cohérence, la fiabilité et la reproductibilité des traitements appliqués aux données textuelles du projet.

## Architecture Générale du Pipeline

### Vue d’ensemble

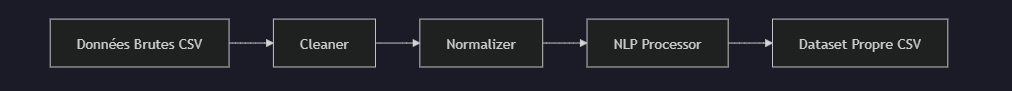
Le pipeline de prétraitement a été conçu de manière modulaire afin de faciliter la maintenance, les tests unitaires et l’évolution du projet. Chaque étape du traitement est isolée dans un module dédié.



### Flux de traitementent

Le flux global du pipeline suit l’enchaînement logique suivant :

* Chargement des données brutes depuis un fichier CSV
* Nettoyage des textes
* Normalisation des données
* Prétraitement NLP
* Export du dataset final

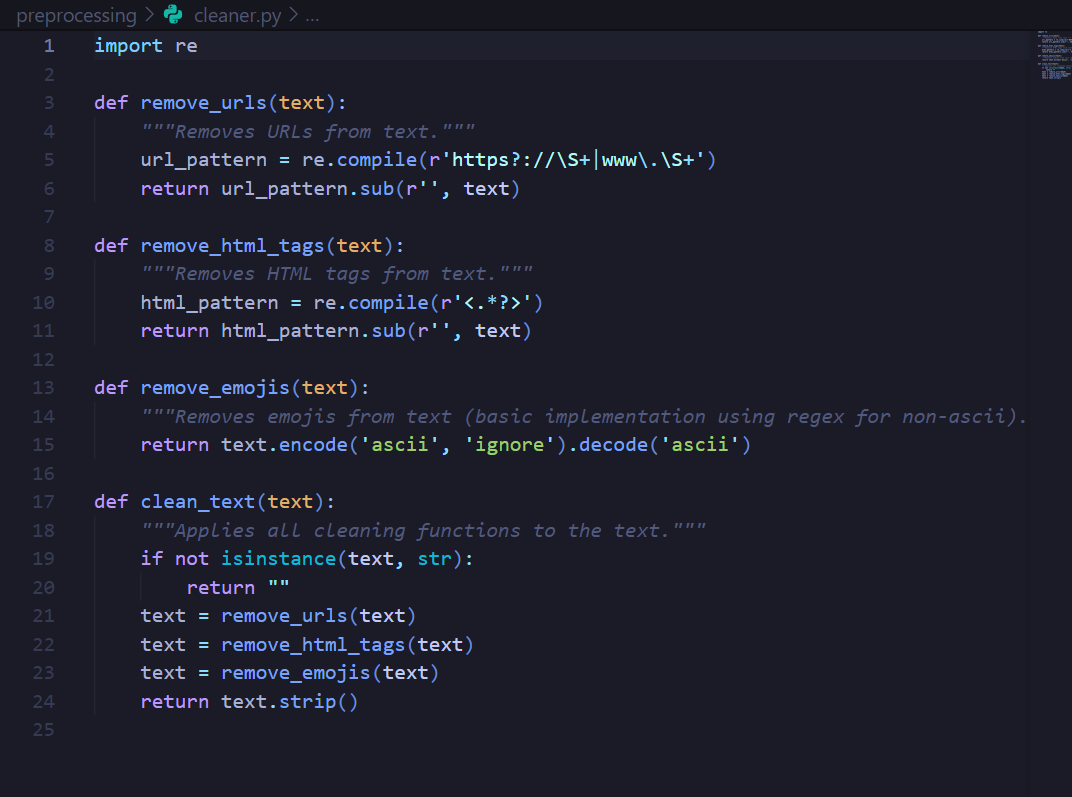


## Nettoyage des Données Textuelles

### Objectifs du nettoyage

L’étape de nettoyage vise à éliminer les éléments inutiles ou perturbateurs présents dans les textes bruts afin de réduire le bruit et d’améliorer la qualité des données.

### Module cleaner.py



## Normalisation des Textes

### Rôle de la normalization

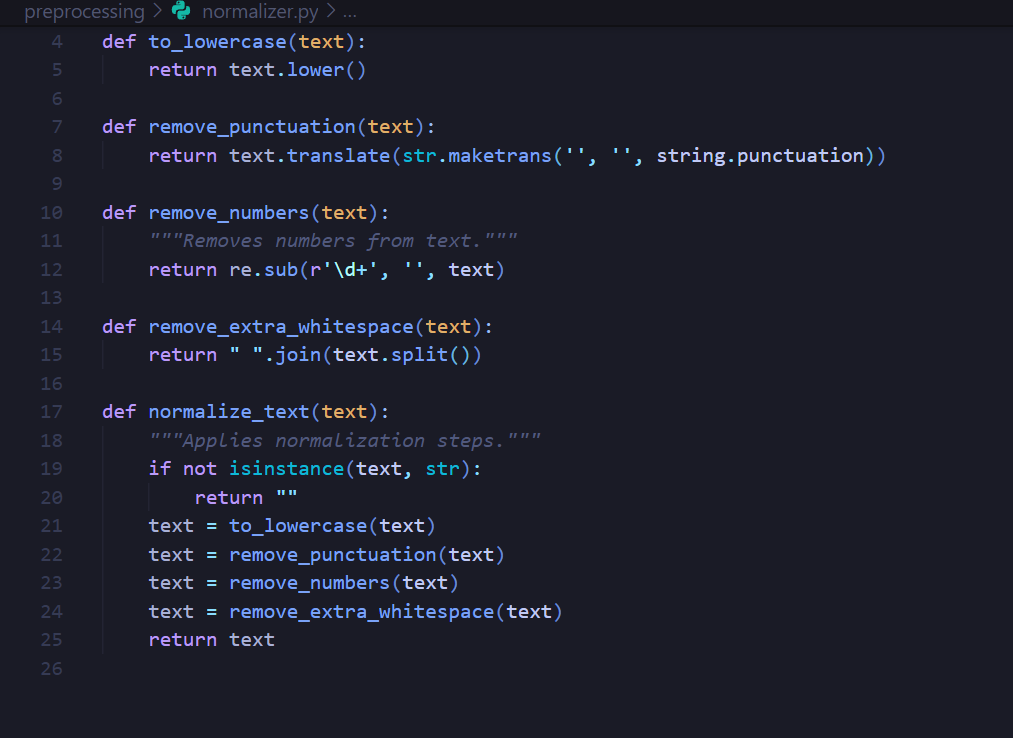
La normalisation permet d’uniformiser les textes afin de faciliter leur traitement par les algorithmes NLP et de réduire les variations inutiles.

### Module normalizer.py

Ce module applique plusieurs transformations, notamment la conversion en minuscules, la suppression de la ponctuation, des chiffres et des espaces multiples.

Fonctions principales :

* to\_lowercase()
* remove\_punctuation()
* remove\_numbers()
* remove\_extra\_whitespace()
* normalize\_text()



## Prétraitement NLP

### Techniques NLP appliquées

Le prétraitement NLP permet de transformer le texte normalisé en une représentation exploitable par les modèles de Machine Learning.

Les techniques utilisées sont :

* Tokenisation
* Suppression des stopwords anglais
* Lemmatisation

### Module nlp\_processor.py



### Pipeline Principal de Prétraitement

Le fichier pipeline.py orchestre l’ensemble des modules précédents et applique le pipeline complet sur les données brutes.

Chaîne de traitement : clean\_text() → normalize\_text() → process\_nlp()

Le résultat final est exporté dans un fichier CSV propre.



## Orchestration avec Apache Airflow

### Présentation de l'Orchestration

L'automatisation du pipeline de prétraitement est assurée par **Apache Airflow**. Cet outil garantit une exécution planifiée, reproductible et supervisée des traitements. Cette orchestration permet d'intégrer le prétraitement des données dans une logique **Big Data** orientée production.

Le workflow est configuré via le DAG défini dans le fichier preprocessing\_dag.py.

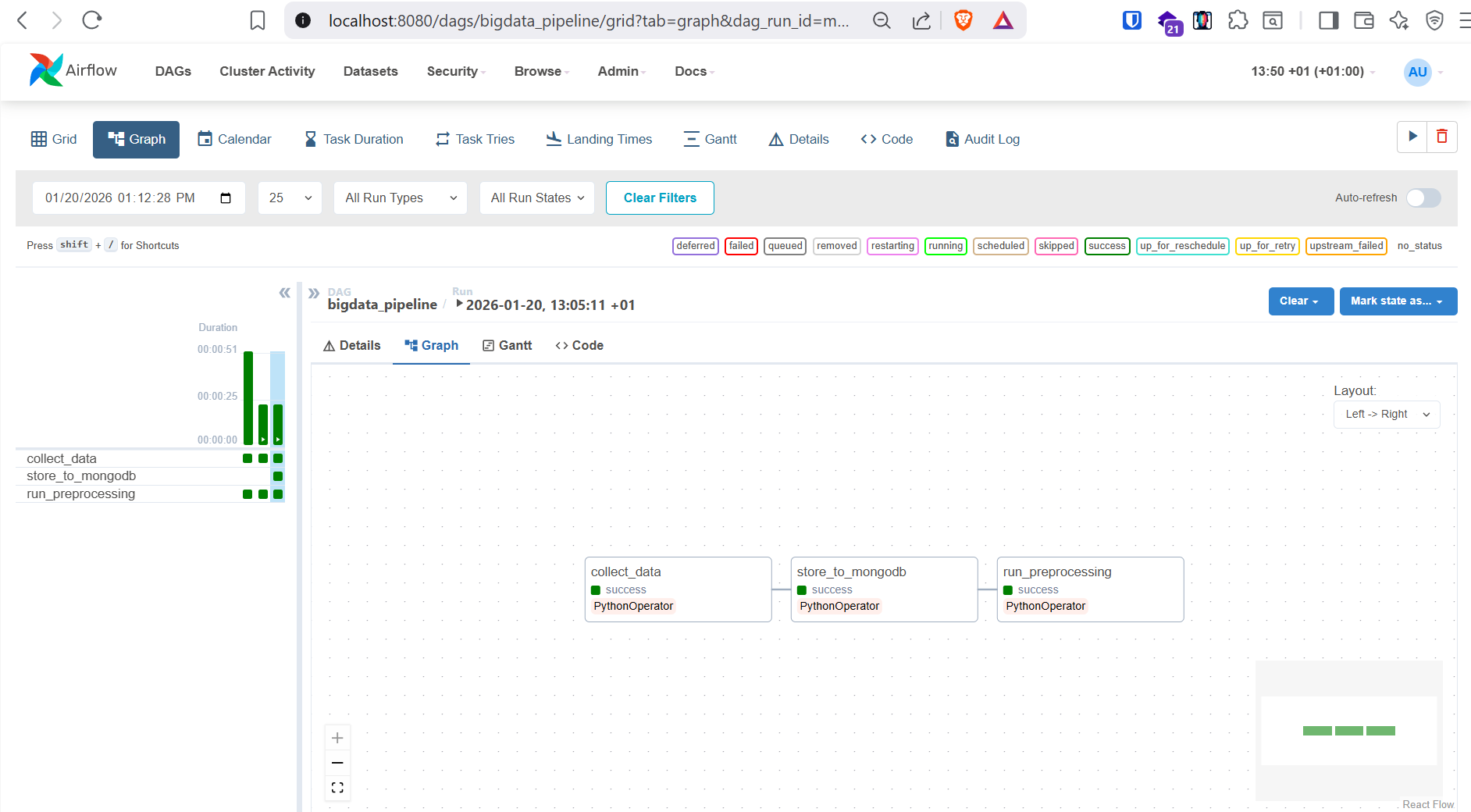
### Configuration du DAG

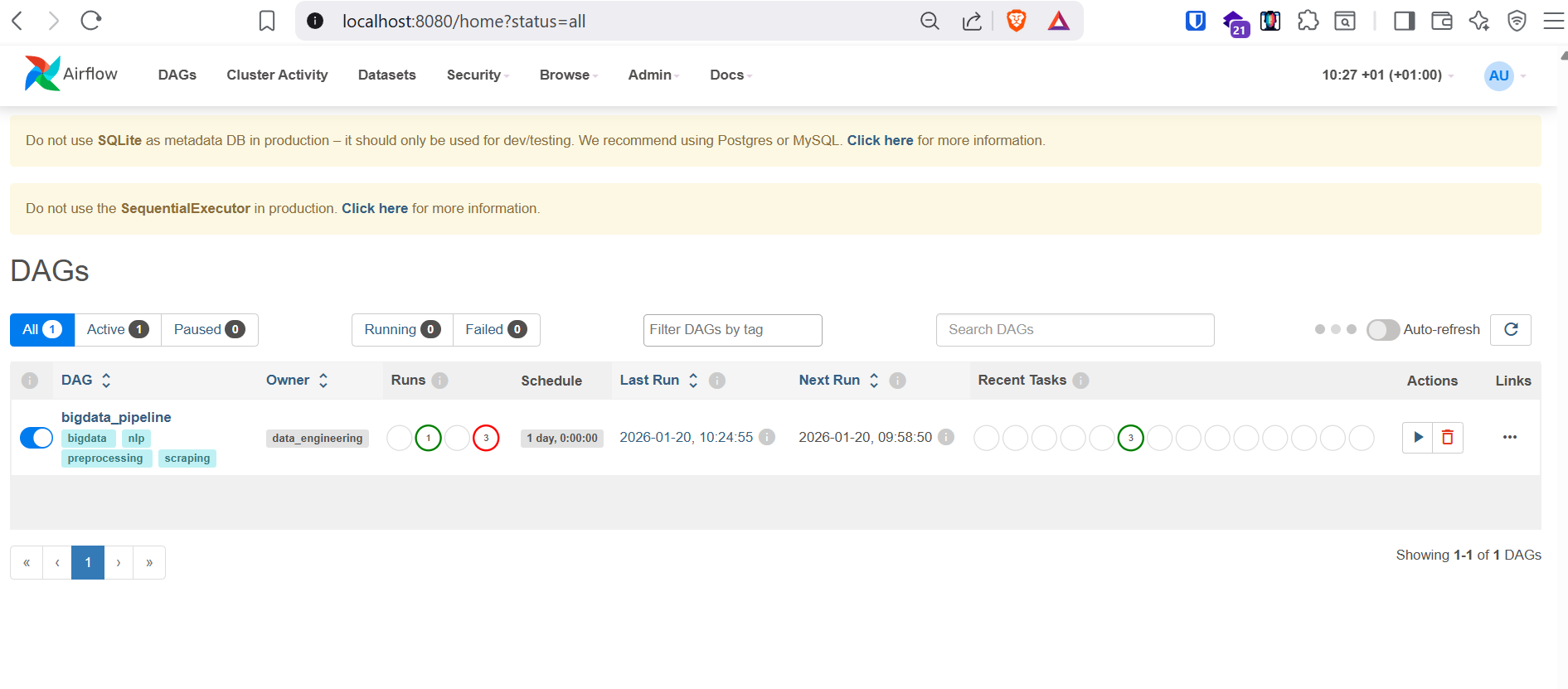
Les paramètres principaux du DAG bigdata\_pipeline sont les suivants :

* **Nom du DAG** : bigdata\_pipeline
* **Propriétaire (Owner)** : data\_engineering
* **Planification** : Exécution quotidienne (timedelta(days=1))
* **Tags** : bigdata, scraping, preprocessing, nlp

### Structure du Workflow

| **Tâche** | **Fonction** | **Description** |
| --- | --- | --- |
| **collect\_data** | Collecte | Récupère les nouvelles données (via API ou scraping). |
| **store\_to\_csv** | Stockage | Sauvegarde les données brutes dans un fichier CSV. |
| **run\_preprocessing** | Prétraitement | Applique le nettoyage et le traitement NLP (Natural Language Processing). |





# Sentiment & Stance Analysis Component

## Objectifs du Module ML

Le cœur de ce projet repose sur une question fondamentale : "La perception de l'Intelligence Artificielle est-elle globalement positive ou négative ?".

Dans le cadre de cette architecture de données (Kafka-MongoDB-PowerBI), le module de Machine Learning (ML) intervient comme le moteur d'intelligence capable de transformer une donnée brute et textuelle en une information structurée et quantifiable.

Les objectifs spécifiques de ma partie étaient les suivants :

* Analyse de "Stance" (Positionnement) : Contrairement à une analyse de sentiment classique (positif/négatif), l'objectif ici est de détecter la posture de l'auteur vis-à-vis de l'IA. Nous avons défini trois catégories de classification :
  + Agreed (Favorable) : Textes soulignant les bénéfices, les innovations et l'optimisme lié à l'IA.
  + Against (Défavorable) : Textes exprimant des craintes (éthique, emploi, vie privée) ou des critiques envers l'IA.
  + Neutral (Neutre) : Textes purement informatifs, factuels ou sans opinion marquée (articles de presse, rapports techniques).
* Enrichissement de la Pipeline de Données : Intégrer un modèle capable de traiter en temps réel (ou par lots) les données stockées dans MongoDB afin d'ajouter une "étiquette" (label) à chaque entrée.
* Automatisation de l'Étiquetage (Data Labeling) : Développer une méthode d'annotation automatique via l'API Gemini pour pallier l'absence de données labellisées au départ, permettant ainsi d'entraîner un modèle supervisé robuste.
* Aide à la Décision via Power BI : Fournir une base de données classifiée permettant de générer des indicateurs clés (KPIs) sur la tendance actuelle de l'opinion publique, visualisables par les décideurs sur le dashboard final.

## Préparation des Données et Stratégie d'Étiquetage

La qualité d'un modèle de Machine Learning dépend directement de la qualité des données en entrée. Cette étape a représenté un défi majeur : transformer des données brutes provenant de MongoDB en un dataset structuré, équilibré et annoté.

### Extraction et Ingénierie des Caractéristiques (Feature Engineering)

Les données extraites de notre base MongoDB comprenaient trois champs textuels distincts : le titre (title), la description (description) et les mots-clés (keywords).

Pour maximiser la compréhension sémantique du modèle, j'ai procédé à une fusion de ces colonnes afin de créer un bloc de texte riche en contexte. Cette approche permet au modèle de ne pas se baser uniquement sur un titre parfois trop court, mais de saisir l'intention globale de l'article ou du post.

Format final : "Title: [Titre] | Description: [Description] | Keywords: [Mots-clés]"

### Labellisation Automatisée via l'API Gemini (Silver Labeling)

L'un des obstacles principaux était l'absence de labels (étiquettes) dans les données collectées via Kafka. Un étiquetage manuel de plusieurs centaines de lignes aurait été chronophage.

J'ai donc mis en place une stratégie de "Silver Labeling" en utilisant l'API Gemini Pro.

* Méthodologie : En fournissant à Gemini des instructions précises (Prompt Engineering), l'IA a analysé chaque entrée pour déterminer si elle était en faveur de l'IA (Agreed), contre (Against) ou simplement informative (Neutral).
* Mappage : Les résultats ont ensuite été convertis en valeurs numériques pour l'entraînement
  + agreed →→ 0
  + neutral →→ 1
  + against →→ 2

### Gestion du Déséquilibre des Classes (Oversampling)

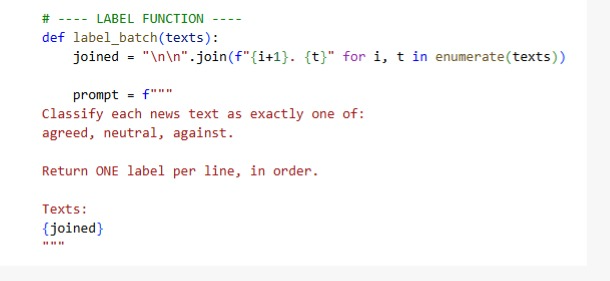
Lors de l'analyse exploratoire initiale, nous avons constaté une prédominance de données "Neutres", ce qui risquait d'introduire un biais : le modèle aurait tendance à classer chaque nouveau texte comme neutre par défaut.

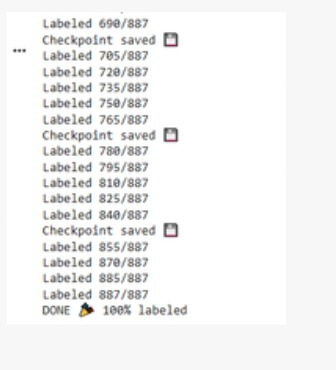
Pour corriger cela, j'ai appliqué une technique de rééchantillonnage (Resampling) :

* Sur-échantillonnage (Oversampling) : Les classes minoritaires ("Against" et "Agreed") ont été multipliées de manière aléatoire pour atteindre 400 échantillons chacune.
* Sous-échantillonnage (Undersampling) : La classe "Neutral" a été ajustée pour correspondre à ce même volume.
* Résultat : Un dataset parfaitement équilibré de 1200 échantillons, garantissant une sensibilité égale du modèle pour chaque point de vue

### Division du Dataset

Le dataset final a été divisé selon un ratio 80% pour l'entraînement et 20% pour le test, en utilisant une stratification. La stratification garantit que la proportion de chaque classe est conservée dans les deux sous-ensembles, permettant une évaluation finale honnête et rigoureuse.





## Architecture Technique du Modèle

Cette phase constitue le moteur intelligent du projet. Pour transformer des textes bruts en indicateurs de positionnement (stance), nous avons implémenté une architecture basée sur le Few-Shot Learning et les Sentence Embeddings.

### Framework SetFit (Sentence Fine-Tuning)

Le choix de l'architecture s'est porté sur SetFit, un framework de pointe développé par Intel Labs et Hugging Face. Contrairement aux modèles de deep learning classiques qui nécessitent des milliers d'exemples annotés, SetFit est conçu pour être extrêmement performant avec très peu de données.

Pourquoi SetFit plutôt qu'un BERT classique ?

* Efficacité des données : SetFit utilise l'apprentissage contrastif, ce qui lui permet d'atteindre une précision élevée avec seulement quelques dizaines d'exemples par classe.
* Rapidité d'entraînement : Le modèle n'a pas besoin de mettre à jour les milliards de paramètres de l'encodeur complet, ce qui réduit drastiquement le temps d'entraînement et l'empreinte carbone du projet.
* Stabilité : SetFit est moins sujet au phénomène "d'oubli catastrophique" que les modèles Transformers standards lors du fine-tuning sur de petits jeux de données.

### Modèle de Base : MPNet-base-v2

Le cœur sémantique du système repose sur le modèle MPNet (Masked and Permuted Pre-training for Language Understanding).

* Puissance sémantique : MPNet combine les avantages de BERT (Masked Language Modeling) et de XLNet (Permuted Language Modeling). Cela lui permet de comprendre non seulement le sens des mots, mais aussi l'ordre et le contexte global de la phrase.
* Représentation vectorielle : Ce modèle transforme chaque bloc de texte (Titre + Description) en un vecteur numérique de 768 dimensions. Dans cet espace mathématique, deux phrases ayant un sens proche (ex: "L'IA est une chance" et "L'IA révolutionne le monde") se retrouveront physiquement proches l'une de l'autre, facilitant la classification.

### Mécanisme d'Entraînement en Deux Étapes

L'entraînement du modèle a suivi un processus rigoureux en deux phases distinctes :

Étape 1 : Phase Contrastive (Fine-tuning de l'encodeur)

Pendant cette phase, le modèle génère des paires de phrases à partir de notre dataset labellisé.

* Il apprend à réduire la distance entre les phrases de même classe (ex: deux textes "Against").
* Il apprend à augmenter la distance entre les phrases de classes différentes (ex: un texte "Agreed" et un texte "Against").
* Paramètre clé : Nous avons configuré num\_iterations=10, ce qui signifie que pour chaque échantillon, le modèle génère 10 paires contrastives, renforçant ainsi sa capacité de discrimination.

Étape 2 : Phase de Classification

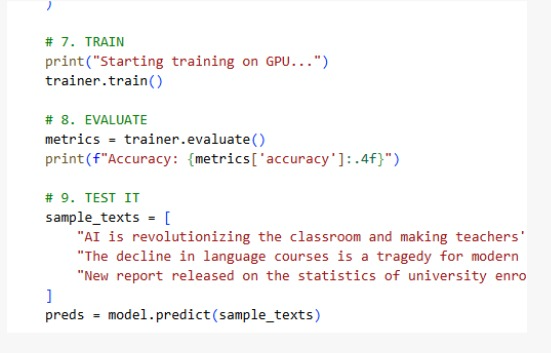
Une fois que l'encodeur sait parfaitement séparer les idées dans l'espace vectoriel, une tête de classification (Logistic Regression) est entraînée sur les embeddings produits. C'est cette couche finale qui prend la décision de classer le texte en Agreed, Neutral ou Against.

### Environnement de Calcul et Hyperparamètres

Pour garantir la reproductibilité et l'efficacité du modèle, les paramètres suivants ont été appliqués

* Hardware : Utilisation d'un processeur graphique GPU NVIDIA T4 (via Google Colab) pour accélérer les calculs matriciels.
* Batch Size (16) : Un compromis optimal entre vitesse de calcul et stabilité de la mémoire GPU.
* Époques : Fixé à 1, car l'efficacité de l'apprentissage contrastif de SetFit permet une convergence extrêmement rapide sans risque de sur-apprentissage (overfitting).
* Optimiseur AdamW : Utilisé pour sa gestion efficace de la décroissance du poids (weight decay), évitant que le modèle ne se focalise trop sur des détails spécifiques du dataset de test.

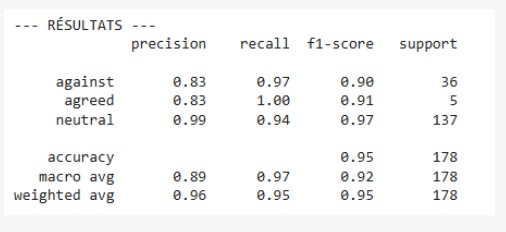


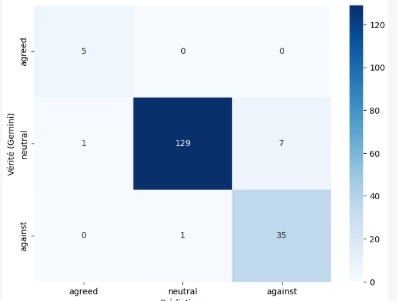


## Analyse des Résultats et Évaluation des Performances

Cette section présente l'évaluation rigoureuse du modèle de classification de stance vis-à-vis de l'IA. L'objectif est de valider si le modèle entraîné est capable de fournir des données fiables pour le dashboard final sur Power BI.

### Analyse Globale des Performances





Le modèle a été évalué sur un jeu de données de test indépendant, composé de 178 échantillons annotés. Les résultats globaux sont exceptionnels :

* Précision Globale (Accuracy) : 95.00%
* F1-Score Pondéré (Weighted Avg) : 0.95

Une précision de 95% indique que le modèle a parfaitement réussi à capturer la logique d'annotation initialement définie par l'API Gemini, tout en étant capable de généraliser ces concepts à de nouvelles données issues du flux Kafka.

### . Analyse détaillée par classe

L'examen des métriques par catégorie (Agreed, Against, Neutral) permet de comprendre la finesse du modèle :

* Classe "Neutral" (Support: 137) : Avec un F1-score de 0.97 et une précision de 0.99, le modèle excelle dans l'identification des textes purement informatifs. C'est un point crucial pour ne pas fausser les statistiques avec des bruits de fond médiatiques.
* Classe "Against" (Support: 36) : Le modèle affiche un Recall de 0.97. Cela signifie que le système détecte quasiment toutes les critiques ou craintes exprimées envers l'IA. La précision de 0.83 suggère que le modèle est légèrement "prudent", classant parfois des textes neutres comme étant contre l'IA par mesure de sécurité.
* Classe "Agreed" (Support: 5) : Malgré un faible nombre d'échantillons dans le set de test, le modèle a obtenu un Recall de 1.00. Il n'a manqué aucune déclaration positive, démontrant que les caractéristiques sémantiques de l'enthousiasme envers l'IA ont été très bien apprises lors de la phase contrastive.

### Interprétation de la Matrice de Confusion

La matrice de confusion visuelle (Heatmap) apporte des informations clés sur le comportement du modèle :

* Diagonale Principale : On observe une concentration massive des prédictions sur la diagonale (129 pour Neutral, 35 pour Against, 5 pour Agreed), confirmant la robustesse du système.
* Analyse des erreurs : Les erreurs sont marginales. Seuls 7 cas de textes "Neutres" ont été classés comme "Against". Cette confusion est fréquente en NLP car de nombreux articles factuels rapportent des risques liés à l'IA de manière objective, ce qui peut être interprété par le modèle comme une posture d'opposition.

### Performance d'Inférence et Intégration Pipeline

Au-delà de la précision, la performance temporelle a été mesurée pour valider l'intégration dans la pipeline Kafka-MongoDB :

* Latence d'inférence : Le modèle traite chaque texte en quelques millisecondes (via GPU T4).
* Consommation de ressources : L'architecture SetFit est légère, ce qui permet au script de traitement de lire les messages de MongoDB et de renvoyer les prédictions sans créer de goulot d'étranglement (bottleneck) dans le flux de données.

### Conclusion sur le Module de Machine Learning

En conclusion, ce module de Machine Learning remplit parfaitement son rôle. Il transforme avec succès une donnée textuelle brute et non structurée en une information catégorielle précise.

Grâce à ce taux de succès de 95%, nous pouvons affirmer que les visualisations Power BI produites en aval reflètent fidèlement l'état réel de l'opinion publique sur l'IA. La méthodologie employée — Labeling via Gemini API + Fine-tuning SetFit/MPNet — s'avère être une stratégie gagnante pour ce projet de Big Data.

# Social Network Analysis Dashboard

## Objectif du chapiter

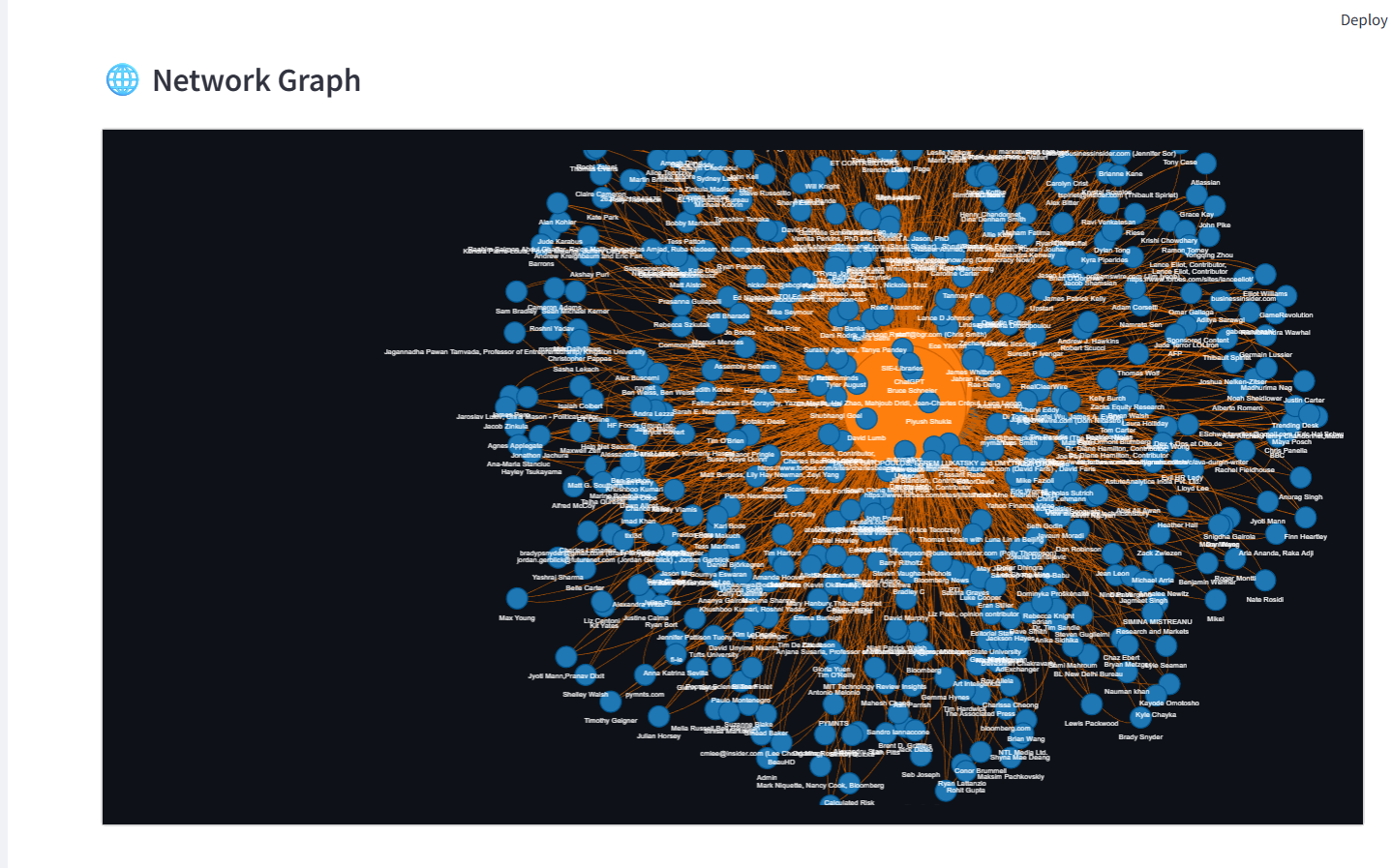
Ce chapitre présente une analyse de réseau social appliquée à des données issues d’articles d’actualité. L’objectif est d’étudier les relations entre les auteurs et les mots-clés afin d’identifier les éléments les plus influents et les principales thématiques abordées.

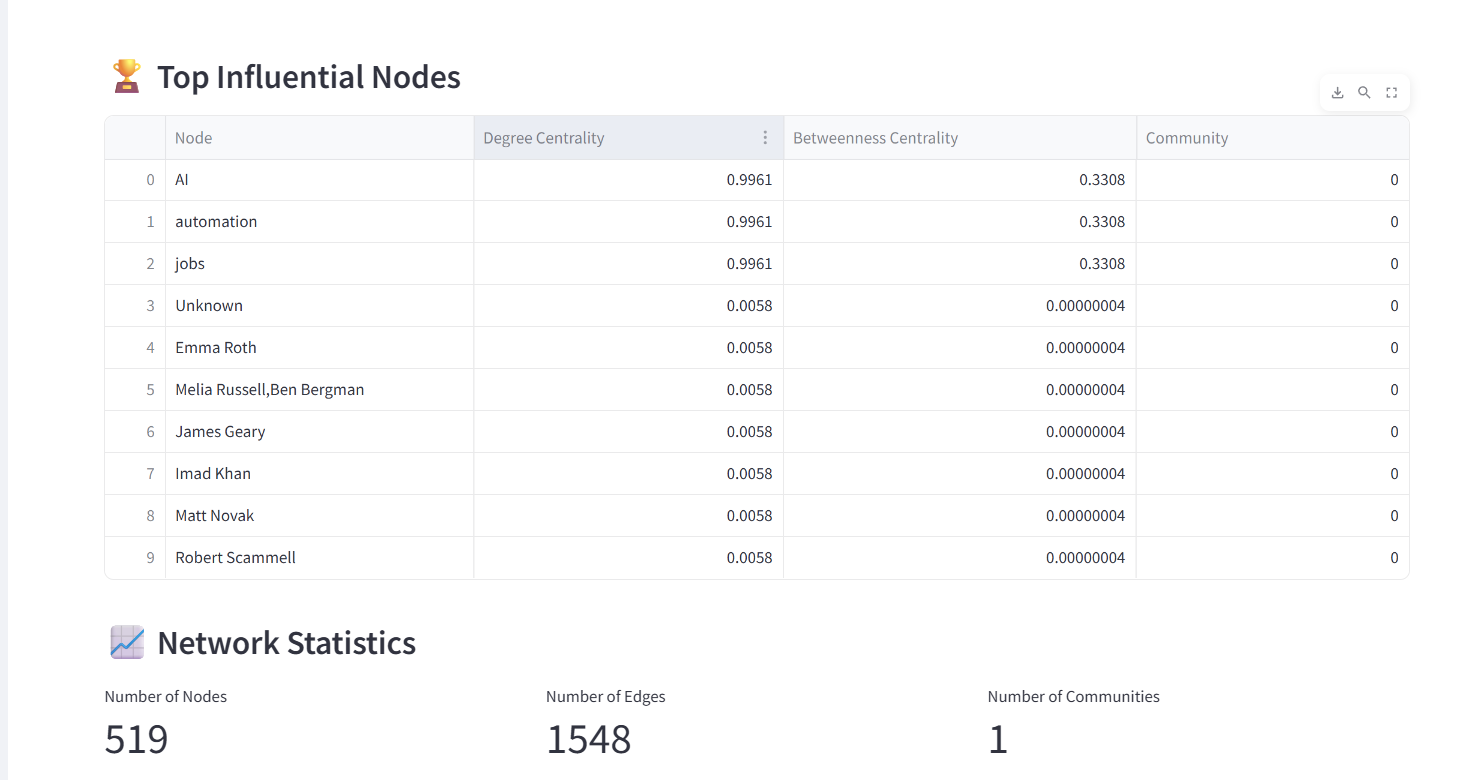
## Approche méthodologique

L’analyse repose sur la construction d’un graphe auteur–mot-clé à partir de données prétraitées. Des mesures de centralité (degré et intermédiarité) sont utilisées pour évaluer l’importance des nœuds, tandis qu’un algorithme de détection de communautés permet d’identifier des groupes thématiques cohérents.

## Résultats et interpretation

Les résultats sont présentés à l’aide d’un tableau de bord interactif permettant de visualiser le réseau, de filtrer par mot-clé et d’identifier les auteurs et thématiques les plus centraux. Cette analyse fournit une lecture synthétique et visuelle de la structure du réseau étudié.





# Conclusion

Ce projet a permis de mettre en place une architecture Big Data complète et performante pour analyser en temps réel la perception médiatique de l'intelligence artificielle. En intégrant des technologies de pointe telles qu'Apache Kafka pour le streaming, MongoDB pour le stockage flexible et Apache Airflow pour l'orchestration des pipelines, nous avons réussi à transformer des flux de données brutes issus de NewsAPI en informations structurées et exploitables. L'un des succès majeurs de ce travail réside dans le module de Machine Learning qui, grâce à l'utilisation du framework SetFit et du modèle MPNet-base-v2, a atteint une précision exceptionnelle de 95 % dans la classification des postures médiatiques. Enfin, la combinaison de l'analyse de réseaux sociaux et des visualisations interactives sur Power BI offre une aide à la décision concrète, permettant de décrypter avec finesse les thématiques dominantes et les dynamiques d'influence au sein du paysage médiatique actuel.