Document Technique : Plateforme d'Analyse de Contenus Bluesky

I. Introduction

Dans un paysage numérique où la propagation rapide de l'information est devenue la norme, la capacité à discerner le vrai du faux est un enjeu majeur. Ce projet vise à répondre à cette problématique en proposant une plateforme web dédiée à l'analyse de la véracité des contenus publiés sur le réseau social décentralisé Bluesky.

L'objectif principal est de fournir aux utilisateurs un outil simple et accessible permettant de soumettre l'URL d'un post Bluesky afin d'obtenir une classification automatisée de son contenu. Grâce à un modèle d'intelligence artificielle spécialisé dans le traitement du langage naturel, l'application évalue si le texte relève d'une information factuelle ("Vrai", "Faux") ou d'une "Opinion personnelle".

Le système est architecturé autour d'un backend robuste qui gère l'interaction avec l'API de Bluesky, l'orchestration du modèle de classification, et la persistance des données dans une base de données relationnelle. Une interface web, sécurisée par un système d'authentification, permet aux utilisateurs de soumettre leurs requêtes et de consulter l'historique des analyses. Ce document détaille l'ensemble des choix technologiques qui sous-tendent cette architecture.

II. Analyse et Choix Technologiques

1 Comparaison des Technologies

Pour chaque brique fonctionnelle du projet, plusieurs solutions ont été évaluées. Les choix finaux ont été guidés par la recherche d'un équilibre optimal entre performance, simplicité de mise en œuvre, coût et maintenabilité.

a) Modèle d'Intelligence Artificielle : Classification de Texte

- **Solution Retenue : BertForSequenceClassification (via Hugging Face Transformers & PyTorch)**
    - **Description :** Modèle de type Transformer pré-entraîné, spécialisé dans la compréhension contextuelle du langage. Son architecture

bidirectionnelle lui permet d'analyser une phrase dans son intégralité pour en saisir le sens profond. Il a été fine-tuné pour notre tâche spécifique de classification ("Vrai", "Faux", "Opinion personnelle").

- **Alternatives Écartées :**
  - *Approches statistiques (TF-IDF + Régression Logistique/SVM)* : Ces modèles, bien que très rapides à mettre en place, se basent sur la fréquence des mots et ne capturent ni le contexte ni les nuances sémantiques (ironie, sous-entendus). Leurs performances auraient été nettement insuffisantes pour une tâche aussi complexe que la détection de fausses informations.
  - *Architectures Récurrentes (LSTM/GRU)* : Bien que capables de traiter des séquences, ces modèles sont plus anciens et généralement moins performants que les Transformers pour les tâches de compréhension de texte. De plus, les entraîner à partir de zéro est un processus long, coûteux en ressources et qui nécessite une très grande quantité de données labellisées. L'approche par transfert d'apprentissage avec BERT est donc beaucoup plus efficace.

b) Framework Web et Backend

- **Solution Retenue : Flask**
  - **Description :** Micro-framework Python, léger et modulaire. Il fournit les bases solides pour le routage des requêtes, la gestion des templates et l'intégration avec d'autres bibliothèques, sans imposer une structure rigide.
  - **Alternatives Écartées :**
    - *Django* : Framework "tout inclus" ("batteries-included"). Très puissant, sa structure plus rigide et son grand nombre de composants intégrés (ORM, panneau d'administration) étaient superflus pour la portée actuelle du projet. Flask offre une plus grande flexibilité et permet de ne choisir que les composants strictement nécessaires.

c) Gestion de la Base de Données

- **Solution Retenue : PostgreSQL avec l'ORM SQLAlchemy**
  - **Description :** PostgreSQL est un système de gestion de base de données relationnelle open-source réputé pour sa robustesse, sa conformité aux standards SQL et ses fonctionnalités avancées. SQLAlchemy, en tant qu'ORM, permet de manipuler la base de

données en utilisant des objets Python, ce qui simplifie le code, réduit les erreurs et le rend agnostique vis-à-vis du SGBD sous-jacent.
    - **Alternatives Écartées :**
        - *SQLite :* Excellent pour le développement et les applications locales, mais ses limitations en matière de gestion des accès concurrents le rendent inadapté à une application web destinée à évoluer.
        - *MySQL :* Une alternative très viable. Le choix de PostgreSQL a été motivé par sa gestion native de types de données complexes comme JSONB, un atout pour stocker à l'avenir des résultats d'analyse plus structurés.

d) Interaction avec l'API Bluesky

- **Solution Retenue : SDK atproto**
    - **Description :** La bibliothèque Python officielle pour interagir avec le protocole AT et le réseau Bluesky.
    - **Alternatives Écartées :**
        - *Requêtes HTTP manuelles (avec requests) :* Cette approche aurait nécessité de gérer manuellement l'authentification (gestion des tokens), la construction des requêtes, la gestion des sessions et l'analyse des réponses JSON. L'utilisation du SDK officiel abstrait toute cette complexité, rendant le code plus propre, plus fiable et plus facile à maintenir face aux évolutions de l'API.

2 Critères de Comparaison et Pertinence Fonctionnelle

Chaque choix technologique a été justifié par son adéquation avec les spécifications fonctionnelles et les besoins du système. Le tableau ci-dessous résume cette analyse.

Technologie	Performance	Facilité de Développement	Maintenabilité	Coût	Pertinence par rapport aux Spécifications
BERT	**Élevée.** Sa compréhension sémantique de pointe est indispensa	**Élevée.** L'écosystème Hugging Face (transformers) rend le	**Moyenne à Élevée.** La maintenance du modèle pré-entraîné est	**Nul.** Modèle et bibliothèques open-source.	**Excellente.** Répond directement au besoin fonctionnel central : "Classifier le

	ble pour une classification précise et nuancée du texte.	chargement et l'utilisation de modèles complexes triviaux.	simple. Une éventuelle phase de ré-entraînement demanderait plus d'efforts.		contenu d'un post".
Flask	**Élevée.** Léger et rapide, il n'introduit pas de latence superflue et est parfaitement adapté à la charge de travail de l'application.	**Très Élevée.** Sa courbe d'apprentissage rapide et sa flexibilité sont idéales pour prototyper et développer rapidement.	**Élevée.** Sa nature modulaire permet de structurer le code de manière claire et logique (par exemple, en séparant les routes, les modèles et les services).	**Nul.** Open-source.	**Excellente.** Permet de construire l'interface web requise pour que les utilisateurs authentifiés soumettent des URL.
PostgreSQL	**Très Élevée.** Robuste et reconnu pour sa capacité à gérer une grande volumétrie de données et de multiples connexions simultanées.	**Élevée.** L'utilisation de l'ORM SQLAlchemy facilite grandement les interactions et rend le code plus lisible et sécurisé (prévention des injections SQL).	**Élevée.** Écosystème mature, excellente documentation et large support communautaire.	**Nul.** Open-source.	**Excellente.** Assure la persistance des données requises : comptes utilisateurs, posts analysés et résultats des détections.
atproto	**Optimale.** Conçue et	**Très Élevée.** Simplifie	**Élevée.** Maintenue	**Nul.** Open-sour	**Indispensable.** C'est le

	maintenue par l'équipe de Bluesky, elle garantit une interaction efficace et conforme avec l'API.	radicalement l'accès aux données de la plateforme, en masquant la complexité du protocole sous-jacent.	par les créateurs du protocole, elle garantit la compatibilité avec les évolutions futures de l'API.	ce.	moyen le plus fiable et le plus pérenne pour accomplir la tâche de récupération de posts.

III. Retour d'Expérience et Veille Technologique

- **Retour d'Expérience :**
  - **Gestion des types de données :** Le principal défi technique rencontré, bien que classique dans les pipelines de données, a été la gestion des types de données entre les bibliothèques. Le modèle PyTorch, optimisé pour le calcul numérique, produit des tenseurs et des nombres au format numpy.float32. Or, le pilote de base de données psycopg2 attend des types Python standards pour la sérialisation. Cette incompatibilité a provoqué des erreurs à l'exécution et a nécessité une conversion explicite (float(numpy_number)) avant l'insertion en base de données. Cet ajustement, bien que simple, souligne l'importance des étapes de nettoyage et de formatage des données à l'interface de chaque composant de la stack.
  - **Gestion de la session SQLAlchemy :** Une autre difficulté récurrente a été l'erreur DetachedInstanceError. Elle survenait lorsque des objets étaient récupérés de la base de données, que la session était fermée, puis que le code tentait d'accéder à des attributs liés (comme tweet.author). La solution a consisté à utiliser systématiquement des stratégies de chargement impatient (joinedload) pour s'assurer que toutes les données nécessaires sont chargées en une seule fois, avant la fermeture de la session.
- **Veille Technologique et Évolutions Futures :**
  - **Asynchronisme pour une meilleure réactivité :** Pour résoudre le problème de latence inhérent à la chaîne d'analyse synchrone, la prochaine itération du projet devra déléguer cette tâche à un

gestionnaire de tâches asynchrones comme **Celery avec un broker de messages (Redis ou RabbitMQ)**. L'utilisateur soumettrait l'URL, recevrait une réponse immédiate, et le résultat de l'analyse apparaîtrait sur son tableau de bord une fois terminée, améliorant drastiquement l'expérience utilisateur.

- **Optimisation du Modèle pour la Production :** Pour une mise en production à plus grande échelle, où le coût et la vitesse d'inférence deviennent critiques, des modèles plus légers comme **DistilBERT** seraient une alternative pertinente. Obtenu par distillation de connaissances, DistilBERT conserve une grande partie (environ 97%) de la performance de compréhension de BERT, mais avec 40% de paramètres en moins, le rendant idéal pour des déploiements sur des infrastructures moins coûteuses.
- **Conteneurisation et Déploiement Cloud :** L'architecture actuelle est parfaitement compatible avec un déploiement conteneurisé via **Docker**. La création d'un Dockerfile permettrait d'encapsuler l'application Flask et toutes ses dépendances dans une image portable et reproductible. L'utilisation de docker-compose permettrait d'orchestrer localement les différents services (application, base de données). Pour la production, cette approche ouvre la voie à des déploiements sur des services managés (Amazon RDS, Google Cloud SQL) et des plateformes d'hébergement d'applications (Heroku, Google App Engine) ou des orchestrateurs de conteneurs (Kubernetes) pour une scalabilité et une résilience maximales.

IV. Conclusion

Ce document a présenté en détail l'architecture technique de la plateforme d'analyse de contenus Bluesky, en justifiant les choix technologiques majeurs qui la sous-tendent. De l'adoption d'un modèle BERT fine-tuné pour la classification textuelle, à l'utilisation de Flask pour le backend, de PostgreSQL et SQLAlchemy pour la persistance des données, et du SDK atproto pour l'intégration avec Bluesky, chaque composant a été sélectionné pour sa robustesse, sa performance et sa pertinence fonctionnelle.

Les retours d'expérience ont mis en lumière des défis classiques de l'ingénierie logicielle, tels que la gestion des types de données et des sessions ORM, dont la résolution a renforcé la fiabilité et la maintenabilité de la plateforme. En outre, la veille technologique a permis d'esquisser des pistes d'évolution

stratégiques, notamment l'intégration de l'asynchronisme pour améliorer l'expérience utilisateur, l'optimisation des modèles d'IA pour un déploiement en production, et la conteneurisation pour une scalabilité et une portabilité accrues dans des environnements cloud.

En somme, cette plateforme représente une solution solide et évolutive dans la lutte contre la désinformation sur les réseaux décentralisés. Elle démontre une approche méthodique de la conception logicielle, combinant des technologies de pointe avec une vision pragmatique des défis de développement et des opportunités futures, ouvrant la voie à des améliorations continues et à une adaptation aux besoins croissants du paysage numérique.