Document technique

1. Introduction

a. Contexte et Motivation

Dans un monde où l'information circule à une vitesse fulgurante, les fausses informations, ou « fake news », se propagent rapidement, influençant souvent les décisions et les opinions publiques. Que ce soit sur les réseaux sociaux, les sites de nouvelles ou même dans les médias traditionnels, il est de plus en plus difficile de distinguer le vrai du faux. Cette montée des fake news pose de réels enjeux sociétaux, politiques et économiques. La désinformation peut causer des dommages importants en trompant le public, en exacerbant les divisions sociales, et en compromettant les fondements d'une démocratie saine.
Face à cette problématique, des outils permettant d'automatiser la détection des fake news sont devenus une nécessité. En combinant les technologies modernes de traitement du langage naturel (NLP) et de machine learning, il est possible de créer des systèmes intelligents capables d'analyser et de prédire la véracité des informations.

b. Objectif de la solution

L'objectif de cette solution est de fournir un outil puissant et simple d'utilisation pour détecter automatiquement les fake news à partir de contenus textuels. Cette solution se compose de deux parties : une API qui effectue les prédictions à partir des modèles d'intelligence artificielle et une application Streamlit qui offre une interface utilisateur conviviale pour interagir avec l'API.

- L'API est responsable de la gestion des requêtes et de l'analyse des textes soumis. Elle utilise un modèle de machine learning entraîné sur un large jeu de données annoté pour identifier la probabilité qu'une nouvelle soit fausse ou véridique.
- L'application Streamlit sert de couche d'interaction utilisateur, permettant à quiconque d'entrer un texte, d'obtenir une évaluation sur la véracité de l'information, et de visualiser les résultats en temps réel. Cette application rend le processus accessible à des utilisateurs non techniques.

Cette solution vise à fournir une méthode rapide et efficace pour analyser de grandes quantités d'informations, détecter la désinformation et ainsi contribuer à un écosystème informationnel plus fiable.

2. Architecture de la solution

a. Présentation générale de l'architecture

La solution de détection des fake news repose sur une architecture modulaire, composée de deux principaux éléments : une **API de détection** et une **interface utilisateur Streamlit**. Ces composants interagissent de manière fluide pour offrir une expérience utilisateur simplifiée, tout en encapsulant la complexité du traitement et de l'analyse des textes.
L'architecture est conçue de manière à être évolutive, permettant d'intégrer de nouveaux modèles de machine learning ou d'ajouter des fonctionnalités supplémentaires si nécessaire. Le cœur de l'application réside dans un modèle de machine learning spécialisé, qui analyse le texte fourni pour prédire s'il contient des informations fausses ou trompeuses.
Voici un aperçu des principaux composants de l'architecture :

- **API Backend** : Cette API constitue la partie backend de la solution. Elle reçoit des requêtes contenant le texte à analyser, traite ces données à l'aide du modèle de machine learning, et renvoie une réponse sous forme de prédiction. L'API est conçue pour être robuste et peut gérer un grand volume de requêtes.
- **Modèle de Machine Learning** : Situé au cœur de l'API, le modèle de machine learning a été entraîné à partir de vastes ensembles de données annotées, comprenant des exemples de fake news et de véritables informations. Le modèle utilise des techniques avancées de traitement du langage naturel (NLP) pour analyser le texte et produire une prédiction.
- **Application Frontend Streamlit** : L'application Streamlit fournit une interface utilisateur intuitive permettant aux utilisateurs d'interagir avec l'API. Elle offre un champ de saisie pour soumettre un texte ou un article, et affiche les résultats de la détection de fake news de manière compréhensible. Streamlit est également utilisé pour visualiser les détails du processus, comme la probabilité de fake news calculée par le modèle.

b. Flux de données

Le flux de données entre les différents composants de la solution peut être décrit comme suit :

1. **Soumission d'un texte** : Un utilisateur soumet un article, un texte ou une phrase via l'application Streamlit.
2. **Envoi à l'API** : L'application Streamlit envoie ce texte sous forme de requête HTTP à l'API de détection. L'API reçoit cette requête et l'analyse.
3. **Traitement par le modèle** : L'API passe le texte soumis au modèle de machine learning qui, grâce à des techniques de NLP, analyse le texte et évalue la probabilité qu'il s'agisse d'une fake news.
4. **Réponse de l'API** : Le modèle génère une prédiction que l'API renvoie à l'application Streamlit sous forme de réponse JSON.
5. **Affichage des résultats** : L'application Streamlit interprète la réponse de l'API et affiche les résultats à l'utilisateur.

c. Technologies utilisées

- **Python** : Utilisé pour le développement de l'API et du modèle de machine learning.
- **FastAPI** : Pour la création de l'API RESTful permettant la communication entre le modèle et l'application frontale.
- **Streamlit** : Une bibliothèque Python qui permet de créer des applications web interactives et faciles à déployer.
- **Scikit-learn** : Pour l'entraînement et la mise en place du modèle de machine learning.
- **NLTK / Transformers** : Pour le traitement du langage naturel (NLP), en particulier la tokenisation, l'analyse syntaxique, et la vectorisation des textes.
- **Docker** : Pour containeriser l'application et la rendre portable entre les différents environnements.

3. API

Fonctionnalités principales

L'API joue un rôle central dans la détection des fake news. Elle reçoit des textes soumis par les utilisateurs via des requêtes HTTP, les transmet au modèle de machine learning pour analyse, et renvoie les résultats sous forme de réponse JSON. L'API est conçue pour être légère, efficace et capable de traiter des volumes importants de requêtes.
Les fonctionnalités principales de l'API sont les suivantes :

1. Endpoint de prédiction (/predict) :
    - Méthode : POST
    - Description : Ce point de terminaison accepte un texte soumis par l'utilisateur et renvoie une prédiction sur la probabilité que ce texte soit une fake news.
    - Paramètre d'entrée :
        - text (obligatoire) : Chaîne de caractères contenant l'article.
    - Réponse : Un objet JSON contenant la probabilité que l'information soit fausse.

4. Modèle de Détection des Fake News

Description du Modèle

Le modèle de détection des fake news utilise un classificateur **Naive Bayes** associé à un pipeline de prétraitement des données textuelles et catégorielles. Le choix de **Multinomial Naive Bayes** est pertinent pour les tâches de classification de texte, en particulier avec des données transformées à l'aide de la méthode **TF-IDF (Term Frequency-Inverse Document Frequency)**, qui permet de pondérer les mots en fonction de leur importance dans le texte.

Ce modèle est structuré en plusieurs étapes :

- **Transformation des titres** : Utilisation de la vectorisation TF-IDF pour transformer les titres des articles en vecteurs numériques en tenant compte des fréquences de mots.
- **Transformation des textes** : La même approche TF-IDF est appliquée au contenu complet des articles pour capturer les mots clés les plus pertinents.
- **Encodage des données catégorielles** : Les données telles que les dates, considérées comme des variables catégorielles, sont traitées à l'aide d'un **OneHotEncoder**, qui permet de transformer ces variables en une représentation numérique interprétable par le modèle.

Prétraitement des données

Le prétraitement des données joue un rôle crucial dans la performance du modèle. Voici les étapes spécifiques intégrées dans le pipeline de transformation :

1. TF-IDF Vectorizer pour les titres :
    - Chaque titre est transformé en vecteur numérique en utilisant la méthode TF-IDF. Cela permet d'évaluer la pertinence des mots dans le titre tout en réduisant l'impact des mots communs ou insignifiants.
2. TF-IDF Vectorizer pour le contenu textuel :
    - De la même manière, le contenu complet des articles est vectorisé. Cette étape permet de capturer les relations importantes entre les termes du texte principal de l'article.

3. Encodage des variables catégorielles (dates) :
    - Les dates sont transformées à l'aide d'un encodage OneHot, où chaque date est représentée par un vecteur binaire, permettant au modèle de détecter d'éventuels motifs liés à la temporalité des fake news.

Le préprocesseur combine ces différentes transformations dans un seul pipeline, assurant une normalisation cohérente des données avant de les transmettre au modèle.

Entraînement du Modèle

Le modèle est entraîné en utilisant une technique d'apprentissage supervisé. Le jeu de données est divisé en deux sous-ensembles : X_train et X_test pour les features (caractéristiques), et y_train et y_test pour les labels (vérité terrain), avec une répartition de 80 % pour l'entraînement et 20 % pour le test.
Voici un aperçu du pipeline d'entraînement :

1. Séparation des données :
    - Le jeu de données est divisé en ensembles d'entraînement et de test à l'aide de la fonction train_test_split, garantissant une bonne généralisation du modèle et une évaluation robuste.
2. Entraînement du modèle :
    - Le pipeline complet, incluant le préprocesseur et le classificateur Naive Bayes, est ajusté sur l'ensemble d'entraînement X_train et y_train pour apprendre à classifier correctement les articles.

Évaluation

Le modèle est évalué sur l'ensemble de test (X_test et y_test) afin de mesurer sa performance sur des données non vues. Les métriques d'évaluation incluent :

- Précision (Accuracy) : `0.9624188840767175`
- Rappel (Recall) : `0.9621839337659431`
- F1-Score : `0.9621492896118596`

5. Application Streamlit

Présentation de l'interface

L'application Streamlit, déployée sur **Azure**, permet aux utilisateurs de soumettre des articles pour vérifier s'ils sont des fake news grâce à une API de prédiction. L'application inclut également une fonctionnalité de retour utilisateur, permettant d'enregistrer les résultats dans une base de données **PostgreSQL** hébergée sur Azure.
L'interface utilisateur est composée de deux sections principales :

1. **Soumission d'un article pour analyse** : L'utilisateur entre un titre, un texte, un sujet, et une date dans un formulaire.
2. **Feedback utilisateur** : Après avoir reçu une prédiction de l'API (fake ou non), l'utilisateur peut indiquer s'il est d'accord avec le résultat. Ce retour est enregistré dans une base de données PostgreSQL.

Fonctionnalités principales

1. **Formulaire de soumission d'article** :
   - **Titre** : Champ pour saisir le titre de l'article.
   - **Texte** : Une zone de texte pour le contenu complet de l'article.
   - **Sujet** : Champ pour indiquer la catégorie ou le sujet de l'article.
   - **Date** : Sélecteur de date pour la date de publication.

Une fois les champs remplis, l'utilisateur soumet le formulaire, déclenchant un appel à l'API.

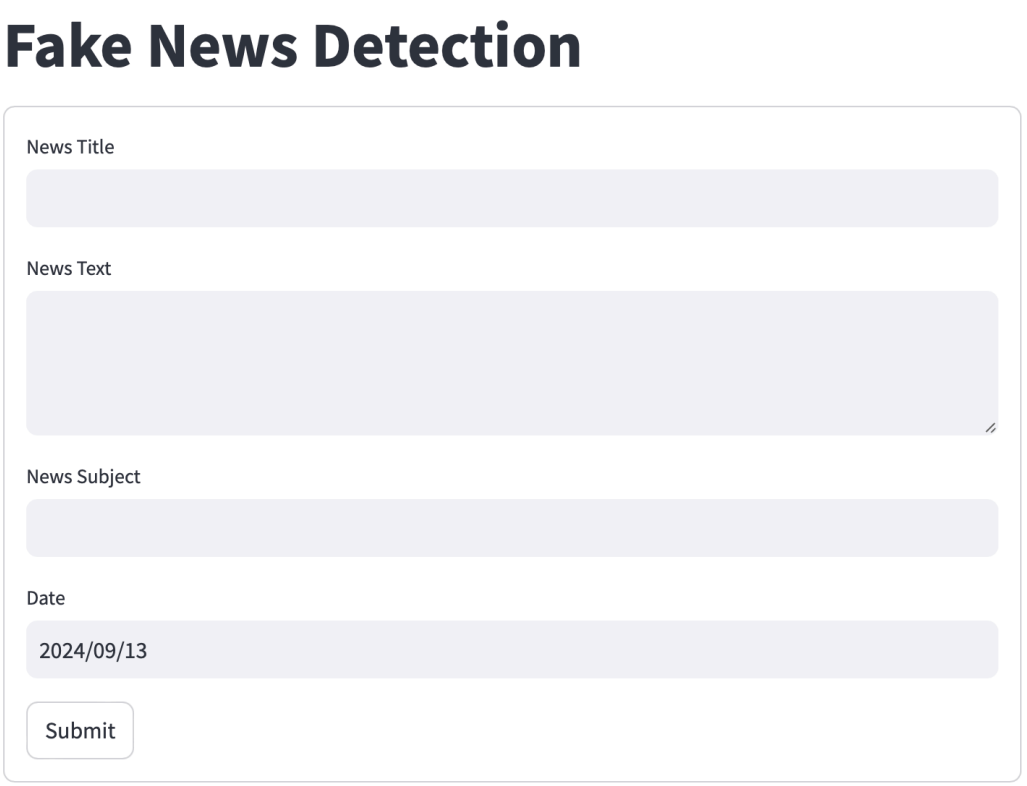
2. **Prédiction de fake news** :
   - La fonction predict_news envoie une requête POST à l'API déployée sur Azure, avec les informations de l'article.
   - L'API renvoie une prédiction (fake ou non), qui est ensuite affichée à l'utilisateur dans l'application Streamlit.
3. **Formulaire de feedback utilisateur** :
   - Après avoir reçu la prédiction, l'utilisateur est invité à indiquer s'il est d'accord avec la classification de l'API.
   - Ce retour est ensuite enregistré dans une base de données PostgreSQL sur Azure à l'aide de la fonction insert_article.

Connexion à la base de données PostgreSQL

La base de données **PostgreSQL** est hébergée sur **Azure Database for PostgreSQL**. Cette base de données stocke les articles soumis ainsi que les retours des utilisateurs. La fonction insert_article gère l'insertion des données dans une table dédiée via une connexion PostgreSQL sécurisée.
La structure de la base de données contient les champs suivants :

- **title** : Titre de l'article.
- **text** : Contenu du texte soumis.
- **subject** : Sujet ou catégorie de l'article.
- **date** : Date de publication de l'article.
- **label** : Résultat de la prédiction (fake ou non).
- **labelUser** : Retour utilisateur (accord ou désaccord avec la prédiction).

Capture d’écran de l’application streamlit



6. Déploiement

Hébergement de l'API sur Azure

L'API de prédiction des fake news est hébergée sur **Azure App Service**, une plateforme de services cloud entièrement gérée qui permet de déployer facilement des applications web et des API sans avoir à gérer l'infrastructure sous-jacente.

Azure App Service :

- **Étapes de déploiement** :
  - Créer un service **App Service** sur le portail Azure.
  - Configurer l'API à l'aide de **FastAPI** en tant qu'application Python.
  - Déployer le code source via GitHub.

Hébergement de l'application Streamlit sur Azure

L'application Streamlit est également hébergée sur **Azure App Service**, en parallèle de l'API. Cela permet à l'application frontale d'être accessible de manière fiable et évolutive.

1. **Déploiement avec GitHub** :
   - **Étapes de déploiement** :
     - Créer une nouvelle instance App Service pour l'application Streamlit.
     - Déployer l'application à l'aide de GitHub Actions.
     - Assurer la connexion avec l'API en utilisant l'URL publique fournie par Azure App Service.
2. **Connexion avec la base de données PostgreSQL** :
   - Streamlit interagit avec une base de données PostgreSQL hébergée sur **Azure Database for PostgreSQL**.

Base de données PostgreSQL sur Azure

La base de données utilisée pour stocker les articles soumis et les retours des utilisateurs est hébergée sur **Azure Database for PostgreSQL**.

- **Étapes de configuration** :
  - Créer une instance de base de données PostgreSQL sur le portail Azure.
  - Configurer les règles de pare-feu pour autoriser les connexions depuis l'API et l'application Streamlit.

7. Améliorations futures

Malgré les résultats positifs obtenus, plusieurs axes d'amélioration peuvent être envisagés pour perfectionner la solution :

1. **Amélioration du modèle de classification** :
    - **Nouveaux modèles** : Tester des modèles plus avancés comme les **transformers** (ex. BERT) ou les réseaux neuronaux pour améliorer les performances du modèle, notamment en matière de précision et de rappel.
    - **Enrichissement des données** : Augmenter la taille du jeu de données en intégrant des articles provenant de différentes sources, ou en ajoutant des données multilingues pour étendre la portée du modèle.
    - **Fine-tuning** : Affiner le modèle existant en le réentraînant régulièrement avec de nouveaux articles soumis et en intégrant les retours utilisateurs pour une amélioration continue.
2. **Fonctionnalités supplémentaires dans l'application** :
    - **Analyse multilingue** : Étendre la capacité de détection des fake news à plusieurs langues en utilisant des modèles NLP multilingues.
    - **Téléchargement de fichiers** : Permettre aux utilisateurs de soumettre des documents (PDF, Word) pour analyse, ce qui serait utile pour traiter des rapports ou des articles complets.
    - **Analyse des liens URL** : Intégrer une fonctionnalité permettant de soumettre directement des liens d'articles, avec un système d'extraction automatique du texte de l'article.
    - **Historique des prédictions** : Ajouter un tableau de bord pour que les utilisateurs puissent consulter un historique des articles soumis et des prédictions correspondantes.
3. **Optimisation de l'infrastructure** :
    - **Amélioration des performances** : Optimiser le pipeline pour réduire le temps de réponse des requêtes API et de l'application Streamlit.
    - **Gestion des logs** : Mettre en place un système de gestion des logs plus avancé avec **Azure Log Analytics** pour un meilleur suivi des erreurs et de la performance.
4. **Analyse avancée des retours utilisateurs** :
    - Utiliser les retours des utilisateurs pour ajuster les prédictions du modèle. Par exemple, en analysant les tendances dans les retours d'accord ou de désaccord, il serait possible de réévaluer certains aspects du modèle ou des règles de prédiction.
    - Intégrer des techniques d'interprétation des modèles comme **LIME** ou **SHAP** pour offrir des explications plus détaillées des prédictions et ainsi augmenter la transparence du système.