**Documentation - API FastAPI**

Cette documentation détaille le fonctionnement et les routes de l'API développée avec FastAPI. Elle couvre les différents endpoints, l'utilisation des modèles LDA et FastText, ainsi que les fonctionnalités de feedback et d'exportation de données.

* Modules et Librairies Importées

- json, joblib, numpy, pandas: Manipulation des fichiers, modèles et données.

- FastAPI: Framework pour créer l'API.

- HTTPException, Query: Gestion des erreurs HTTP et des paramètres de requêtes GET.

- BaseModel: Pour valider les données des requêtes.

- StreamingResponse: Gestion du streaming (ici pour exporter les CSV).

- StringIO: Manipulation des objets en mémoire de type fichier.

- mysql.connector.Error: Gestion des erreurs liées à la base de données.

- utils.db\_utils.create\_connection, utils.db\_utils.insert\_feedback: Fonctions utilitaires pour la connexion à la base de données et l’insertion de feedback.

* Chargement des Modèles

Modèle LDA et vectorizer pour la classification des thèmes :

- lda\_model: Modèle LDA (Latent Dirichlet Allocation) chargé à partir d'un fichier .pkl.

- vectorizer: Utilisé pour transformer le texte en vecteur.

Modèle d'analyse de sentiment :

- fasttext\_model: Modèle FastText pour obtenir les vecteurs des mots.

- logistic\_model: Modèle de régression logistique pour prédire les sentiments.

Thèmes des topics :

- Un fichier JSON contient les thèmes associés à chaque topic LDA (topics.json).

* Fonctions Utilitaires

- get\_document\_vector(text): Génère un vecteur moyen des tokens d'un texte à partir des embeddings FastText.

- predict\_sentiment(text): Prédiction du sentiment d'un texte avec un vecteur généré et le modèle logistique.

* Routes FastAPI

a. GET /status

Renvoie un message confirmant que l'API est en ligne.

Réponse :

{ "message": "API en ligne!" }

b. POST /predict

Prend un texte en entrée, le transforme en vecteur et renvoie la distribution des topics.

Input :

{ "text": "votre texte ici" }

Réponse :

{ "topic\_distribution": [{"theme": "Thème 1", "probability": 0.45}, {"theme": "Thème 2", "probability": 0.55}] }

c. POST /predict\_s

Prend un texte en entrée et prédit le sentiment (positif/négatif) avec le modèle logistique.

Input :

{ "text": "votre texte ici" }

Réponse :

{ "sentiment": "positif" }

d. POST /feedback\_topic

Soumet un retour utilisateur sur une prédiction de thème.

Input :

{ "text\_input": "texte fourni", "predicted\_value": "Thème prédit", "real\_value": "Thème réel" }

Réponse :

{ "message": "Feedback enregistré avec succès." }

e. POST /feedback\_sentiment

Soumet un retour utilisateur sur une prédiction de sentiment.

Input :

{ "text\_input": "texte fourni", "predicted\_value": "Sentiment prédit", "real\_value": "Sentiment réel" }

Réponse :

{ "message": "Feedback enregistré avec succès." }

f. GET /export\_data

Permet d'exporter les données d'une table spécifique sous forme de fichier CSV.

Paramètre : table\_name (nom de la table à exporter).

Réponse : Un fichier CSV contenant les données.

* Base de Données

La connexion est établie via la fonction create\_connection(). La fonction insert\_feedback() permet d'insérer des retours dans les tables monitoring\_topic et monitoring\_sentiment.

* Exemples d'Utilisation

- Prédire un thème :

URL : POST http://localhost:8000/predict

Body : { "text": "Votre texte" }

- Prédire un sentiment :

URL : POST http://localhost:8000/predict\_s

Body : { "text": "Votre texte" }

- Exporter des données :

URL : GET http://localhost:8000/export\_data?table\_name=monitoring\_topic

7. Gestion des Erreurs

Retourne des erreurs HTTP en cas de problème de connexion à la base de données ou de texte invalide.