# Feuille de route pour l'entraînement des modèles

10 juin 2024

# Table des matières

## 0.1 Introduction

Le but de cette documentation est d'établir une feuille de route pour la mise en œuvre d'un modèle de classification multi-label de texte. La classification multi-label est une tâche complexe du traitement automatique du langage (TAL) où chaque instance de donnée peut appartenir à plusieurs catégories simultanément.

Cette feuille de route a pour objectif de guider à travers les étapes , la mise en œuvre d'un modèle de classification multi-label en se basant sur les modules développés durant le stage portant sur les projets IA de Tenders-Page.

Nous aborderons le pré-traitement des données, l'entraînement du modèle, l'évaluation des performances et la sauvegarde du modèle dans une base de données MongoDB. En suivant ces étapes, il vous sera possible de reconstruire un modèle multi-label de texte, capable de réaliser une catégorisation automatique des textes.

Les modules et fonctions abordés se trouvent dans le dépôt git **Projet_Categories**.
Ce dépôt est organisé par dossiers selon le type de donnée (donnée en anglais, français où multilingue), et dans chaque dossier se trouve un ensemble de fichiers conçus pour entraîner un modèle dans cette langue . Les étapes décrites ci-dessous sont valables pour pour les 3 types de modèle ( anglais, français où multilingue) que vous souhaitez réentraîner avec de nouvelles données.

## 0.2 Pré-traitement des données

Cette étape consiste à importer les données brutes depuis ElasticSearch vers MongoDB.

— **Fonction à utiliser :** DeElasticSearch_A_MongoDB
— **Module :** importEtInsert_data

### 0.2.1 Mise au format multi-label

Après l'importation, les données doivent être transformées en un format multi-label. Cela permet de gérer des données où chaque instance peut appartenir à plusieurs classes simultanément.

— **Fonction à utiliser :** Creation_data_frame
— **Module :** importEtInsert_data.py

### 0.2.2 Importer les données depuis MongoDB

— **Fonction à utiliser :** Importation_dataFrame
— **Module :** importEtInsert_data.py

### 0.2.3 Analyse des données

Avant l'entraînement du modèle, il est essentiel de visualiser et analyser les données pour identifier et corriger les anomalies telles que le déséquilibre des classes ou les données manquantes. Vous pouvez réaliser autant de visualisations que vous le souhaitez avec la librairie *pandas* de Python. Une des visualisations nécessaire est la distribution des données par catégorie.

— **Fonction à utiliser :**plot_distribution, et nb_exemple_categorie
— **Module :** utility.py

## 0.3 Entraînement du modèle

### 0.3.1 Division des données

Les données doivent être divisées en trois ensembles distincts : entraînement, validation et test. Cela permet de prévenir le sur-apprentissage et de garantir la généralisation du modèle sur des données non vues.

### 0.3.2 Amélioration de l'équilibre des données

Si les classes sont déséquilibrées, il est possible d'améliorer l'équilibre des données d'entraînement en sur-échantillonnant les catégories minoritaires (oversampling) et en sous-échantillonnant les catégories majoritaires (undersampling).

— **Fonction à utiliser :** Undersampling, Oversampling
— **Module :** equilibrer_les_donnees.py

Conseils supplémentaires : Si vous choisissez d'équilibrer les données, il est recommandé de commencer par sur-échantillonner les catégories minoritaires avec Oversampling, puis de sous-échantillonner les catégories majoritaires avec Undersampling. Ensuite, reconstituez les données d'entraînement en concaténant les résultats d'Undersampling et d'Oversampling.

### 0.3.3 Stockage des ensembles de données

Chaque ensemble de données (entraînement, validation, test) doit être stocké dans des collections distinctes dans MongoDB.

— **Fonction à utiliser :** insertion_MDB
— **Module :** importEtInsert_data.py

### 0.3.4 Entraînement du modèle

Entraînez le modèle en utilisant les ensembles de données préparés. Le fichier d'entraînement nécessite les liens vers les collections des données d'entraînement et de validation.

— **Fonction à utiliser :** main

— **Module :** entrainementDuModeleBert.py

## 0.4 Évaluation du modèle

Une fois le modèle entraîné, évaluer ses performances pour s'assurer qu'il répond aux attentes. Cela inclut la mesure de la précision, du rappel, de la F-mesure, etc.

— **Fonction à utiliser :** main

— **Module :** evaluation_modele.py

## 0.5 Sauvegarde du modèle

Après l'évaluation, sauvegardez le modèle entraîné dans MongoDB pour une utilisation ultérieure ou un déploiement en production.

— **Module :** model_insert_mongodb.py

En suivant ces étapes, vous pouvez ré-entraîner et évaluer un modèle de catégorisation d'annonces avec de nouvelles données en utilisant les fonctions et modules disponibles dans le dépôt Projet_Categories.