Projet de Classification de Textes

# Introduction

## Contexte et Importance du Projet

La classification de texte est une tâche cruciale dans de nombreux domaines tels que l'analyse de sentiments, la détection de spam, et la catégorisation d'articles de presse. Avec l'augmentation exponentielle des données textuelles disponibles en ligne, il est essentiel de développer des modèles capables de classer automatiquement les textes de manière efficace et précise. Les réseaux de neurones récurrents (RNN) et leurs variantes avancées comme les LSTM et Transformers ont démontré leur efficacité pour cette tâche.

## 

## Objectif du Projet

L'objectif de ce projet est de développer et d'entraîner des modèles de réseaux de neurones LSTM, LSTM avec mécanisme d'attention, et Transformer pour effectuer une tâche de classification de texte. Vous aurez l'occasion de comparer les performances de ces architectures avancées et d'évaluer leur capacité à classifier correctement des textes en différentes catégories.

## 

## Structure du Projet

Ce projet se déroulera en plusieurs étapes clés :

1. Préparation des Données : Prétraitement et transformation des données textuelles pour l'entraînement des modèles.
2. Entraînement de Modèles : Implémentation et entraînement de trois modèles différents.
3. Évaluation des Modèles : Test et évaluation des performances des modèles sur un jeu de données de test.

# I) Manipulation de la Donnée

## Présentation du Jeu de Données

Vous allez travailler avec le jeu de données de critiques de films de l'IMDb, disponible sur Kaggle. Ce jeu de données contient des critiques de films étiquetées comme positives ou négatives. Il est largement utilisé pour des tâches de classification de texte et vous permettra de vous familiariser avec les techniques de traitement du langage naturel.



## Télécharger les Données

Téléchargez les données directement depuis Python en utilisant l'API Kaggle. Assurez-vous d'avoir une clé d'API Kaggle configurée.

| !pip install kaggle  !kaggle datasets download -d lakshmi25npathi/imdb-dataset-of-50k-movie-reviews  !unzip imdb-dataset-of-50k-movie-reviews.zip |
| --- |

## Charger les Données à l'aide de Python

Utilisez les bibliothèques Python telles que pandas pour charger les critiques et leurs labels.

## Nettoyage et Prétraitement des Textes

Effectuez des opérations de nettoyage des textes (suppression de la ponctuation, des stop words, mise en minuscules, etc.) et tokenisez les textes en utilisant nltk ou spaCy.

| import nltk from nltk.corpus import stopwords from nltk.tokenize import word\_tokenize import string  nltk.download('stopwords') nltk.download('punkt')  stop\_words = set(stopwords.words('english')) df['cleaned\_reviews'] = df['review'].apply(lambda x: ' '.join([word for word in word\_tokenize(x.lower()) if word not in stop\_words and word not in string.punctuation])) print(df['cleaned\_reviews'].head()) |
| --- |

## Encodage des Labels

Utilisez LabelEncoder de sklearn pour transformer les labels catégoriels en valeurs numériques.

| from sklearn.preprocessing import LabelEncoder  le = LabelEncoder() df['sentiment\_encoded'] = le.fit\_transform(df['sentiment']) |
| --- |

## Séparation des Données

Divisez les données en ensembles d'entraînement et de test.

## Padding et Troncature des Séquences

Utilisez pad\_sequences de Keras pour que toutes les séquences aient la même longueur.

| from tensorflow.keras.preprocessing.text import Tokenizer from tensorflow.keras.preprocessing.sequence import pad\_sequences  tokenizer = Tokenizer(num\_words=5000) tokenizer.fit\_on\_texts(X\_train)  X\_train\_seq = tokenizer.texts\_to\_sequences(X\_train) X\_test\_seq = tokenizer.texts\_to\_sequences(X\_test)  X\_train\_pad = pad\_sequences(X\_train\_seq, maxlen=100) X\_test\_pad = pad\_sequences(X\_test\_seq, maxlen=100) |
| --- |

# II) Entraînement de 3 Modèles

## Implémentation de 2 Callbacks

1. Early Stopping : Utilisez EarlyStopping de Keras pour surveiller la perte de validation (val\_loss) et arrêter l'entraînement lorsque celle-ci cesse de s'améliorer.
2. Learning Rate Decay : Proposez une fonction de décroissance pour le taux d'apprentissage, par exemple en utilisant ReduceLROnPlateau de Keras.

## Modèle LSTM from scratch :

Créez un modèle LSTM avec Keras en utilisant la classe Sequential. Ajoutez des couches LSTM, Dense et des mécanismes de régularisation comme le dropout.

| from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout  model\_lstm = Sequential([  Embedding(input\_dim=5000, output\_dim=128, input\_length=100),  LSTM(128, return\_sequences=True),  Dropout(0.2),  LSTM(64),  Dropout(0.2),  Dense(x, activation='xxxxxx')# à vous de voir ]) |
| --- |

## Modèle LSTM avec Attention :

Implémentez un modèle LSTM avec un mécanisme d'attention. Utilisez une couche Attention pour pondérer les sorties des LSTM.

| from tensorflow.keras.layers import Attention  input\_layer = Input(shape=(100,)) embedding\_layer = Embedding(input\_dim=5000, output\_dim=128)(input\_layer) lstm\_layer = LSTM(128, return\_sequences=True)(embedding\_layer) attention\_layer = Attention()([lstm\_layer, lstm\_layer]) output\_layer = Dense(xxx, activation=xxxx)# à vous de voir  model\_lstm\_attention = Model(inputs=input\_layer, outputs=output\_layer) |
| --- |

## 

## Modèle Transformer :

Implémentez un modèle Transformer en utilisant la classe Transformer de Hugging Face ou une autre bibliothèque similaire.

| from transformers import TFBertForSequenceClassification, BertTokenizer  tokenizer = BertTokenizer.from\_pretrained('bert-base-uncased') model\_transformer = TFBertForSequenceClassification.from\_pretrained('bert-base-uncased') |
| --- |

## Entraînement des 3 modèles

Utilisez model.fit pour entraîner chaque modèle en utilisant les callbacks définis précédemment.

## Sauvegarde des Modèles

Utilisez model.save\_weights pour sauvegarder les poids des modèles entraînés.

# III) Test des Modèles

## Calcul de la Matrice de Confusion

Utilisez confusion\_matrix de sklearn pour calculer la matrice de confusion de chaque modèle sur le jeu de test.

## Calcul des Métriques de Performance

Pour chaque modèle, calculez les métriques suivantes : accuracy, precision, f1-score, recall, sensibilité et spécificité. Utilisez les fonctions de sklearn.metrics.

## Affichage des Courbes ROC et Calcul de l’AUC

Affichez les courbes ROC pour chacun des modèles et calculez l'AUC (Area Under the Curve). Interprétez les résultats obtenus.