TP: Analyse de Sentiment avec des Réseaux Bi-LSTM

Objectif :

Implémenter un réseau Bi-LSTM pour l'analyse de sentiment à l'aide d'un ensemble de données au format CSV.

Partie II : Implémenter un réseau Bi-LSTM pour l'analyse de sentiment à l'aide d'un ensemble de données au format CSV.

Étapes :

1. Préparation de l'environnement du travail

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense, Embedding, LSTM, Bidirectional
```

2. Chargement et Exploration du Jeu de Données

- Chargement des données
- Exploration du dataset.

3. Pre-processing

- Nettoyer et prétraiter les textes (enlever la ponctuation, convertir en minuscules, etc.).
- Encoder les données si nécessaire
- Tokeniser les textes en utilisant la fonction Tokenizer de tensorflow.keras.preprocessing.text
- Une fois le corpus de texte entier tokenisé, nous devons convertir chaque critique textuelle en une séquence numérique à l'aide du tokenizer adapté.
- Préciser une longueur de séquence nominale de 200 pour chaque token. max_length = 200
- Les séquences numériques dont la longueur est supérieure à 200 seront tronquées à la fin, tandis que celles dont la longueur est inférieure à 200 seront complétées par des zéros à la fin.

4. Construction du Modèle Bi-LSTM basique

- Créer un modèle avec une couche Embedding (La première couche est une couche d'embedding. Elle sert à représenter les mots sous forme de vecteurs dans un espace vectoriel) avec les spécificités suivantes

  Embedding_dim=32 : Chaque mot sera représenté par un vecteur de 32 dimensions.
  input_length=sequence_length
- Ajouter une couche Bidirectional.
- Ajouter une couche LSTM avec 32 unités. Cela signifie qu'il y a 32 cellules LSTM dans cette couche.
- Ajouter une couche dense avec une seule unité (car il s'agit d'une tâche de classification binaire) et une fonction d'activation sigmoid.

Layer	Dimensions
Embedding	10000 input dimensions, 32 output dimensions
Bidirectional(LSTM)	32 units
Dense	1 unit, sigmoid activation function

- Afficher le résumé du modèle avec la fonction .summary.

HINT

```
model = Sequential()

#Firstly, we will add an embedding layer which will convert each word into a dense vector of embedding dimensions specified in the hyperparameters of the layer

embedding_dim = 16

model.add(Embedding(vocab_size, embedding_dim, input_length=sequence_length))

#Here we have specified the vocabulary size as well as the sequence length of each review. Next, we need to specify the Bidirectional() layer and the LSTM layer with a specified unit size in the LSTM layer:

lstm_out = 32

model.add(Bidirectional(LSTM(lstm_out)))

#Next, we will specify a fully connected layer having 10 units and ‘relu’ activation

model.add(Dense(10, activation='relu'))

#Finally, we will add an output layer having only 1 unit and ‘sigmoid’ activation. This layer will output the probability that an input belongs to 1 (or positive) using the sigmoid filter.

model.add(Dense(1, activation='sigmoid'))

#Now we need to compile the model such that the model optimizes the `binary_crossentropy` during training. The loss value is the parameter that the ‘adam’ optimizer will minimize by tweaking the weights during the training phase.

#Essentially, the ‘adam’ optimizer tries to find the global minima for the loss value among all the local minima by optimizing the trainable parameters. Moreover, the ‘accuracy’ of the model will be reported for each training batch/epoch to gauge the convergence of the neural network.

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

print(model.summary())
```

5. Compilation et Entraînement du Modèle

- Compiler le modèle en spécifiant l'optimiseur (Adam), la fonction de perte (binary_crossentropy car c'est une classification binaire) et les métriques à suivre (dans ce cas, l’accuracy).

- Entraîner le modèle avec les données d'entraînement (x_train et y_train) pendant 10 epochs, avec un batch size de 128.

6. Évaluation du Modèle

- Évaluer la performance du modèle sur l'ensemble de test.
- Calculer la précision.
- Discuter des métriques de performance.

7. Analyse des Paramètres du Modèle

Expliquer les principaux paramètres du modèle Bi-LSTM :

units : Nombre de neurones dans la couche LSTM.

embedding_dim : Dimension de l'espace d'incorporation (embedding).

input_length : Longueur des séquences en entrée.

activation : Fonction d'activation utilisée dans la dernière couche.

- Modifiez le nombre d'unités dans le Bi-LSTM
- Essayez de modifier le nombre d'unités dans le Bi-LSTM. Comment cela affecte-t-il la performance du modèle ?
- Utilisez un autre type de fonction d'activation
- Essayez d'utiliser un autre type de fonction d'activation pour la couche de sortie du modèle. Comment cela affecte-t-il la performance du modèle ?