Advanced Machine Learning for Big Data and Text Processing : Projet TripAdvisor

Abdallah ESSA / Ismail Hour MAHAMAT

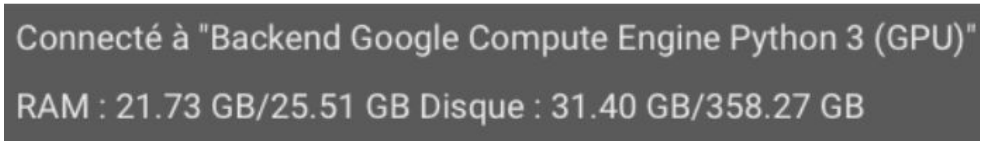
1 Introduction

Ce projet consiste à donner une note de 1 à 5 à chaque commentaire de la dataset “Hotel Reviews” du TripAvisor.Pour cela nous avons utilisé différentes types des modèles (classification ou régression) de deep learning implémentées en tensor flow et Keras. Cependant nous avons créé deux fichiers pour la réalisation de ce projet, la première consiste à faire de la prétraitement des données (IMDB, TripAdvisor) pour enfin exporter tous ces données dans un fichier csv et la seconde pour la

2 Notebook 1 : “ProjetTripAdvisor_Partie_1_Dataset.ipynb”

- 1. NLTK stop words
- 2. WordNet Lemmatizer
- 3. Applique le Prétraitement sur les datasets TripAdvisor et IMDB
- 4. Création d’un csv qui rassemble les deux datasets pour la création du tokenizer.
- 5. Dividision du dataset IMDB en train et test
- 6. Importer tout ces datasets au forma csv pour les utiliser sur le Colab dédier.

RAM du Colab actuelle:



On crée un nouveau notebook sur Colab pour avoir accès à une nouvelle RAM.

3 Notebook 2 : “ProjetTripAdvisor_Final.ipynb”

Après avoir importer les fichiers csv créé par le notebook 1 du prétraitement, on créé un tokenizer sur l’ensemble des données (TripAvisor et IMDB).

- **Entrainement du modèle de langage (Next Word Prediction)**

Le but ici est d'entraîner les embeddings afin d’avoir une couche embedding qui arrive le mieux possible à représenter les mots dans un espace vectoriel. On apprend le modèle à prédire à partir d’une séquence de mots le mot suivant.

Le modèle utilisé :

```
EmbeddingLayer = tf.keras.layers.Embedding(voc_size, 64)
LstmLayer = tf.keras.layers.LSTM(150)
DenseLayerEmbedding = tf.keras.layers.Dense(64, activation='relu')
DropLayerEmbedding = tf.keras.layers.Dropout(0.1)
DenseLayerOutputEmbedding = tf.keras.layers.Dense(voc_size, activation='softmax')
model = tf.keras.Sequential()
model.add(EmbeddingLayer)
model.add(LstmLayer)
model.add(DenseLayerEmbedding)
model.add(DropLayerEmbedding)
model.add(DenseLayerOutputEmbedding)

model.summary()
```

Le modèle est entraîné avant sur les données IMDB puis après 3 epoch sur les données TripAdvisor pour 4 epoch.

- **Entrainement du modèle prédiction de Note :**

Comme les commentaires IMDB ressemble aux commentaires TripAdvisor et qu’ils sont beaucoup plus nombreux plus de 50 000 commentaires on a utilisé ces commentaires pour pré-entraîner notre modèle de prédiction de note.

Le modèle utilisé :

```
BidirectionalLayer1 = tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences = True))
BidirectionalLayer2 = tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences = True))
GlobalMaxPool1DLayer = tf.keras.layers.GlobalMaxPool1D()
DenseLayer1 = tf.keras.layers.Dense(40, kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01) , activation='relu')
DropLayer = tf.keras.layers.Dropout(0.1)
DenseLayerOutputRE = tf.keras.layers.Dense(1, activation=tf.keras.activations.linear)
```

```
modelBin = tf.keras.Sequential()
modelBin.add(EmbeddingLayer5)
modelBin.add(BidirectionalLayer1)
modelBin.add(BidirectionalLayer2)
modelBin.add(GlobalMaxPool1DLayer)
modelBin.add(DenseLayer1)
modelBin.add(DropLayer)
modelBin.add(DenseLayerOutputRE)
```

On utilise la couche Embedding entraîner à la prédiction de mots suivant (Next Word Prediction).

Pour éviter l'over fitting on a rajouter a la couche Dense un regularizer et une couche de Dropout.

N'ayant qu'une seule couche entraîné on va obliger le modèle à entraîner les autres couches avant de pouvoir modifier les Embeddings.

```
1 EmbeddingLayer.trainable = False
```

On utilise Comme Loss l'écart type (mean squared error) et comme optimizer Adam.

```
modelBin.compile(loss='mean_squared_error',
                 optimizer=tf.keras.optimizers.Adam(),
                 metrics=['mse','mae'])
```

```
loss: 0.3848 - mse: 0.1747 - mae: 0.3504 - val_loss: 0.1935 - val_mse: 0.1439 - val_mae: 0.2982

loss: 0.1657 - mse: 0.1391 - mae: 0.2883 - val_loss: 0.1441 - val_mse: 0.1296 - val_mae: 0.2767

loss: 0.1355 - mse: 0.1250 - mae: 0.2628 - val_loss: 0.1333 - val_mse: 0.1258 - val_mae: 0.2465
```

Après 3 epoch on a permis au model d'entraîner tous les couches.

```
1 EmbeddingLayer.trainable = True
```

```
loss: 0.1031 - mse: 0.0972 - mae: 0.2116 - val_loss: 0.0975 - val_mse: 0.0933 - val_mae: 0.1953

loss: 0.0647 - mse: 0.0614 - mae: 0.1437 - val_loss: 0.0875 - val_mse: 0.0847 - val_mae: 0.1577
```

On obtient une val_mse de 0.0847 après 2 epoch. Ensuite on a entrainé le model sur les données TripAdvisor.

```
[ ] mse

0.541063704593672

[ ] result1 = np.where(test_label == 1)
    result2 = np.where(test_label == 2)
    result3 = np.where(test_label == 3)
    result4 = np.where(test_label == 4)
    result5 = np.where(test_label == 5)

[ ] mse1 = mean_squared_error(prediction[result1], test_label[result1])
    mse2 = mean_squared_error(prediction[result2], test_label[result2])
    mse3 = mean_squared_error(prediction[result3], test_label[result3])
    mse4 = mean_squared_error(prediction[result4], test_label[result4])
    mse5 = mean_squared_error(prediction[result5], test_label[result5])

[ ] mse1

1.6392977852399884

[ ] mse2

1.0763499638807565

[ ] mse3

0.9796371669355061

[ ] mse4

0.34002175619521385

[ ] mse5

0.3623820822162271
```

On a obtenu une MSE de 0.54 sur le test set.

Piste etudier :

Au lieu d'étudier le modèle en régression on va l'étudier en classification .

Après l'entraînement de la couche embedding on a entrainé un premier modèle à classifier les commentaires TripAdvisor.

Le modèle utilisé:

```
BidirectionalLayer1 = tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences = True))
BidirectionalLayer2 = tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences = True))
GlobalMaxPool1DLayer = tf.keras.layers.GlobalMaxPool1D()
DenseLayer1 = tf.keras.layers.Dense(40, kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01) , activation='relu')
DropLayer = tf.keras.layers.Dropout(0.1)
DenseLayerOutputCL = tf.keras.layers.Dense(2, activation=tf.keras.activations.softmax)

DenseLayerOutputRE = tf.keras.layers.Dense(1, activation=tf.keras.activations.linear)
modelBin = tf.keras.Sequential()
modelBin.add(EmbeddingLayer5)
modelBin.add(BidirectionalLayer1)
modelBin.add(BidirectionalLayer2)
modelBin.add(GlobalMaxPool1DLayer)
modelBin.add(DenseLayer1)
modelBin.add(DropLayer)
modelBin.add(DenseLayerOutputCL)
```

On a entrainé durant 5 epoch sans modifier les embeddings puis 5 epoch avec le droit de les modifier.

Puis on a créé un nouveau model avec les mêmes couche et poids que le modèle entraîné précédemment
On modifie juste la dernière couche pour classifier en 5 classes.

```
DenseLayerOutput_5 = tf.keras.layers.Dense(5, activation=tf.keras.activations.softmax)
```

Resultat sur test set:

On a entrainé une première fois avec comme loss categorical cross entropy :

```
confusion_matrix_CL_CE
```

```
array([[ 56,   7,  30,  15,   3],
       [ 34,   6,  43,  23,   2],
       [ 15,   2,  66, 136,  30],
       [  8,   0,  23, 315, 205],
       [  2,   0,  11, 197, 787]])
```

```
mse_CL_CE
```

```
0.6884920634920635
```

Puis on a utilisé mean squared error comme loss :

```
confusion_matrix_CL_MSE
```

```
array([[ 36,   0,  56,  16,   3],
       [ 15,   0,  69,  21,   3],
       [  7,   0,  97, 104,  41],
       [  6,   0,  49, 278, 218],
       [  3,   0,  14, 157, 823]])
```

```
mse_CL_MSE
```

```
0.7385912698412699
```

On a également étudier la possibilité de grouper les commentaires en deux groupe. Positif et négatif pour après prédire leur note.

Groupe 1 : classes 1, 2 et 3 ensemble et classes 4 et 5 ensemble

```
# CL CE BI SUP
prediction_CL_CE_BI_SUP = l11
confusion_matrix_CL_CE_BI_SUP = confusion_matrix(l12, prediction_CL_CE_BI_SUP)
mse_CL_CE_BI_SUP = mean_squared_error(prediction_CL_CE_BI_SUP, l12)
```

```
confusion_matrix_CL_CE_BI_SUP # accuracy cl1 = 63.6
```

```
array([[ 298,  170],
       [  82, 1466]])
```

```
mse_CL_CE_BI_SUP
```

```
0.125
```

Groupe 2 : classes 1 et 2 ensemble et classes 3, 4 et 5 ensemble

```
# CL CE BI INF < 3
prediction_CL_CE_BI_INF = l11
confusion_matrix_CL_CE_BI_INF = confusion_matrix(l12, prediction_CL_CE_BI_INF)
mse_CL_CE_BI_INF = mean_squared_error(prediction_CL_CE_BI_INF, l12)
```

```
confusion_matrix_CL_CE_BI_INF # accuracy cl1 = 57.5
```

```
array([[ 126,   93],
       [  41, 1756]])
```

```
mse_CL_CE_BI_INF
```

```
0.06646825396825397
```

On voulait entraîné un modèle de régression à prédire ensuite la note d'un commentaire suivant sans appartenance à un groupe mais par manque de temps, on a pas pu finir.