Université Mohammed V de Rabat
Ecole Nationale Supérieure d'Arts et Métiers
Cycle Ingenieur Filière INDIA (3ème année) A.U. : 2023/2024 Pr. A. El Qadi

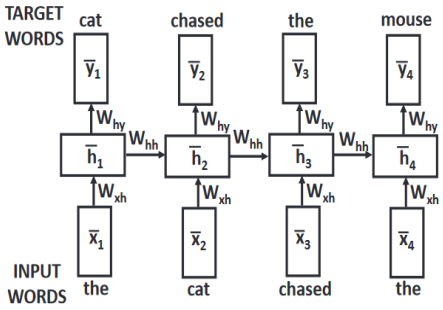
TP n4 en Apprentissage profond

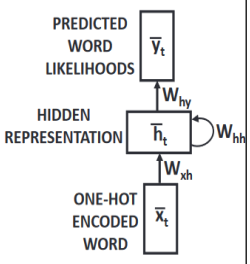
Reseau de neurones recurrent

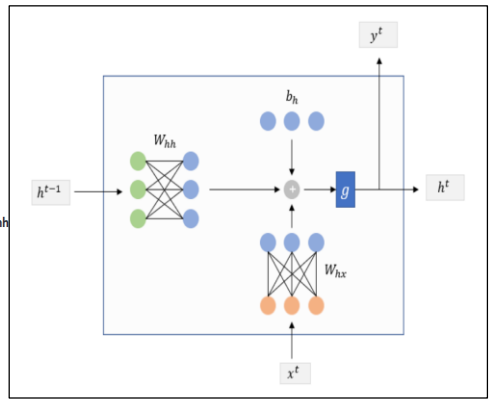
Exercice 1 : Entrainer un RNN simple sans couche cache, sur la sequence de mots : « the cat chased the », pour la prédiction des mots qui suit le mot 'the'. Initiliser les matrices Wxh, et Whh, et le bias b par des valeurs aletoires, avec

ht = activation (dot(Wxh, input_t) + dot(Whh, state_t) + b)
et $yt = ht$

Donner les valeurs de ht, et yt







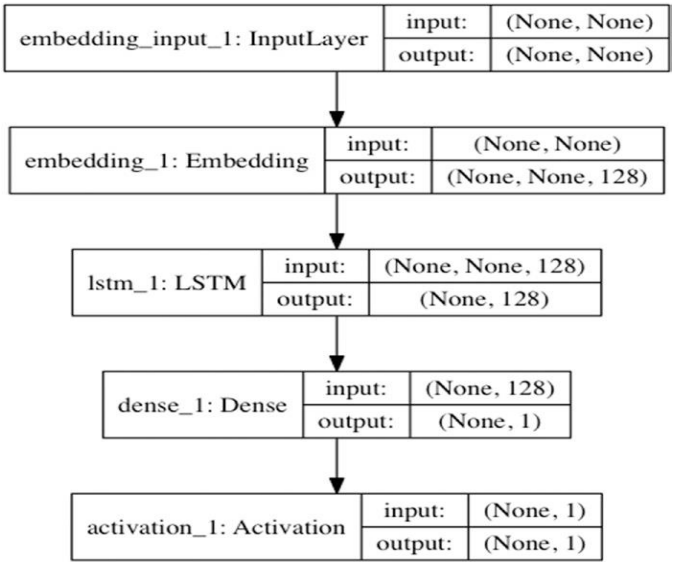
```
import numpy as np
timesteps = 4                #Number of timesteps (pas dans le temps) in the input sequence
input_features = 4           #Dimensionality of the input feature space
output_features = 4          #Dimensionality of the output feature space
# the cat chased the mouse
inputs =  np.array([
      [1, 0, 0, 0],[0, 1, 0, 0],[0, 0, 1, 0],[1, 0, 0, 0]
      ])
state_t = np.zeros((output_features,))

Wxh = np.random.random((output_features, input_features))
Whh = np.random.random((output_features, output_features))
#Why = np.random.random((output_features, output_features))
b = np.random.random((output_features,))
```

……

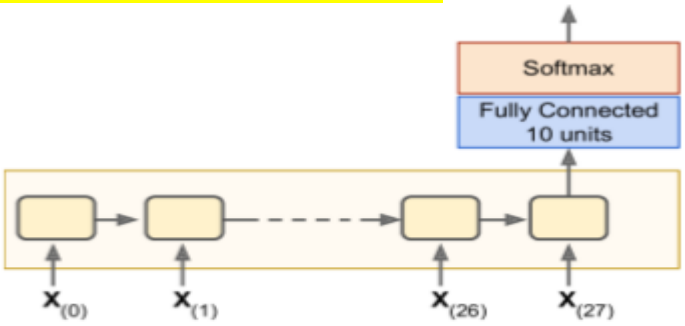
Exercice 2 : Entrainer, evaluer, et compiler le modele d'exercice en utilisant la bibliothque Keras

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import SimpleRNN
```

Exercice 3. Entrainer, evaluer, et compiler le modele LSTM illustré dans la figure ci-dessous.



L'ensemble de données a utilisé provient d'IMDB et représente 25 000 avis sur des films classés comme positifs ou négatifs, ce qui en fait un problème de classification de séquence binaire.
Pour plus d'informations sur l'ensemble de données, consulter le lien suivant, http://ai.stanford.edu/~amaas/data/sentiment/



Les données sont prétraitées pour ne contenir que des mots fréquents (les mots sont en fait représentés sous forme d'entiers).

1. Charger ce jeu de donnes

```
from keras.datasets import imdb
from keras.preprocessing import sequence
max_features = 10000     # Nombre de mots à considérer comme features
maxlen = 500
batch_size = 32                    # pour couper les textes après ce nombre de mots (parmi les
                                   # max_features mots les plus courants)

(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=max_features)
```

2. Redimesionner le modele n_instances=25000, et n_features=500

```
X_train = sequence.pad_sequences(X_train, maxlen=maxlen)
X_test = sequence.pad_sequences(X_test, maxlen=maxlen)
```

3. Entrainer le modele

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
```

```
model = Sequential()
model.add(Embedding(max_features, 128))
…
```

4. Compiler le modele, utiliser l'optimiseur 'adam', et la fonction de perte entropie croise
5. Executer le modele sur les donnees d'apprentissage, avec le nombre d'epochs =10, batch_size=128, et validation_split=0.2
6. Evaluer le modèle : afficher les valeurs de perte, et de l'accuracy
7. Prediction : Déterminer la classe du premier tweet, et afficher uniquement l'etiquette de prob max.

Exercice 4: Prediction de sequence (LSTM many2one)

Soit la sequence suivante :

```
data = [1,2,3,4,5,6,7,8,9,10]
```

1. Ségmenter (split) ce je de donnees en deux tableaux X, et y

X= [[1, 2, 3, 4], [2, 3, 4, 5], [3, 4, 5, 6], [4, 5, 6, 7], [5, 6, 7, 8], [6, 7, 8, 9].
[7, 8, 9,10], [8, 9,10, 11]]
y= [5, 6, 7, 8, 9, 10, 11, 12]

Les valeurs de y representent les sorties de chaque input de X

2. Entrainer un modele LSTM sur les sequences X, et y, pour la prediction de la sortie de la sequence [9,10,11, 12].

Le modele a une seule couche cachee de 50 neurones, et caracterise par la dimesion [batch, timesteps, feature].
Redimensionner les donnees de [samples, timesteps] en [samples, timesteps, features]
Samples=X.shape[0] (represente le nombre d'input data)
Timesteps=X.shape[1] (represente le nombre de variables pour chaque input data)
Features=n_features (represente le nombre de variables pour chaque output data (ici=1))

3. Compiler le modele, utiliser l'optimiseur 'adam', et la fonction de perte tf.keras.losses.MeanSquaredError()
4. Executer le modele sur les donnees d'apprentissage, avec le nombre d'epochs =100
5. Déterminer la valeur suivante pour la sequence : test_data = np.array([9,10, 11, 12]).

Exercice 5 : Prediction du mot suivant (LSTM one2one)

Soit le texte:
text= 'the cat chased the mouse '

1. Segmenter (split) ce texte en tableau de mots, et charger les mots uniques dans le tableau unique_words, et leurs index dans le tableau unique_word_index

```
words=text_to_word_sequence(text)
vocab_size = len(words)
unique_words = np.unique(words)
unique_word_index = dict((c, i) for i, c in enumerate(unique_words))
```

2. Créer les deux sequences : prev_word et next_word
   prev_words= [['the'], ['cat'], ['chased'], ['the']]
   next_words= ['cat', 'chased', 'the', 'mouse']

3. Définir les X, et y (data input, et data output)

```
LENGTH_WORD=1
X = np.zeros((len(prev_words),LENGTH_WORD, len(unique_words)), dtype=bool)
y = np.zeros((len(next_words), len(unique_words)), dtype=bool)

for i, each_words in enumerate(prev_words):
  for j, each_word in enumerate(each_words):
      X[i, j, unique_word_index[each_word]] = 1
  y[i, unique_word_index[next_words[i]]] = 1
```

4. Construire le modèle LSTM qui permet la prédiction du mot suivant de chaque mot dans input data.
5. Utiliser un modèle LSTM monocouche avec 128 neurones, une couche entièrement connectée et une fonction softmax pour l'activation.
6. Compiler le modele, utiliser l'optimiseur ' RMSprop', et la fonction de perte 'categorical_crossentropy'
7. Executer le modele sur les donnees d'apprentissage, avec le nombre d'epochs =20, et afficher les valeurs du parametre history
8. Prédir le mot qui suit le mot 'chased'

Exercice 6 : Generation de texte avec LSTM (many2many)

Je jeu de données utilisé est « New York Times Comments ».
Ce sont des commentaires faits sur les articles publiés dans le New York Times en janvier-mai 2017 et janvier-avril 2018.
Les données mensuelles sont présentées dans deux fichiers csv - un pour chacun des articles sur lesquels des commentaires ont été faits et un pour les commentaires eux-mêmes.
Les fichiers csv pour les commentaires contiennent plus de 2 millions de commentaires au total avec 34 fonctionnalités et ceux pour les articles contiennent 16 fonctionnalités sur plus de 9 000 articles.

1. Charger tous les articles en tant que titres et de les fusionner dans une liste

```
files_path = 'D:/MyData/Cours/DM/Data/NewyorktimesComment/'
all_headlines = []              # charger le contenu de la colonne headline
for filename in os.listdir(files_path):
   if 'Articles' in filename:
      article_df = pd.read_csv(files_path + filename)
      all_headlines.extend(list(article_df.headline .values))

print(all_headlines[:4])
```

2. Afficher le nomdes des titres
3. Nettoyer les donnees : supprimer les ponctuations, les mots en minuscules, etc

```
import string
def clean_text(text):
   text = "".join(t for t in text if t not in string.punctuation).lower()
   text = text.encode("utf8").decode("ascii",'ignore')
   return text

corpus = [clean_text(titre) for titre in all_headlines]
print(corpus[:5])
```

4. Générer une séquence de n-grammes pour l'apprentissage de la prédiction du mot suivant

```
tokenizer = Tokenizer()
```

```
def get_sequence_of_tokens(corpus):
   ## tokenisation
   tokenizer.fit_on_texts(corpus)
   total_words = len(tokenizer.word_index) + 1
   ## convertir les données en une séquence de jetons
   input_sequences = []
   for line in corpus : #pour la ligne dans le corpus :
      token_list = tokenizer .texts_to_sequences([line])[0]
      for i in range(1, len(token_list)):
         n_gram_sequence = token_list[:i+1]
         input_sequences.append(n_gram_sequence)
   return input_sequences, total_words

inp_sequences, total_words = get_sequence_of_tokens(corpus)
print(inp_sequences[:5])
```

5. Afficher le nombre de lignes
6. Utiliser le rembourrage, pour que chaque séquence ait la même longueur

```
import keras.utils as ku
def generate_padded_sequences(input_sequences):
   max_sequence_len = max([len(seq) for seq in input_sequences])
   input_sequences=np.array(pad_sequences(input_sequences,maxlen=max_sequence_len,
padding='pre'))
   X, y = input_sequences[:, :-1],input_sequences[:,-1]
   y = ku.to_categorical(y, num_classes=total_words)

   return X, y, max_sequence_len

X, y, max_sequence_len = generate_padded_sequences(inp_sequences)
```

```
# predictors : ce sont des jetons qui seront utilisés comme entrée pour prédire le mot suivant.
# label: est le prochain mot à prédire.
# max_sequence_len: est la longueur de la séquence.ici max_sequence_len=17
# pad_sequence: fourni par Keras est utilisé pour remplir un tableau de jetons à une longueur donnée.
```

9. Construire le modèle LSTM qui prendra predictors comme entrée X et labels comme entrée y
   - Couche d'entrée: est responsable de la prise de séquence d'entrée.

```
input_len = max_sequence_len - 1
model.add(Embedding(total_words, 10, input_length=input_len))
```

   - Couche LSTM : calcule la sortie à l'aide des unités LSTM et renvoie les états cachés et cellulaires. Dans le cas ici. Ajouter 100 unités.
   - Couche de suppression : responsable de la régularisation, ce qui signifie qu'elle empêche le sur-ajustement. cela se fait en désactivant les activations de certains neurones dans la couche LSTM (Dropout=0.1).
   - Couche de sortie : calcule la probabilité de prédiction.
10. Compiler le modele, utiliser l'optimiseur 'adam', et la fonction de perte categorical_crossentropy
11. Executer le modele sur les donnees d'apprentissage X et label y, avec le nombre d'epochs =100
12. Générer du texte. Prédire le mot suivant en fonction des mots d'entrée. Il faut tokeniser la séquence et la remplir avec le même sequence_length.
    Prédir les 2 mots suivants pour le texte "donald trump"
    Prédir les 5 mots suivants pour le texte : "science et technologie"