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

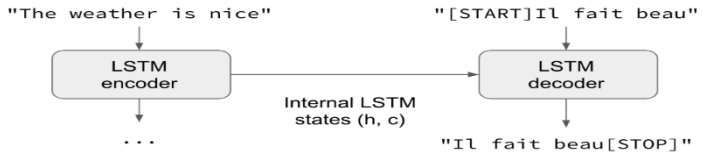
TP n5 en Apprentissage profond

Exercice : EocoderDecoder pour la traduction de texte

L'apprentissage séquence à séquence (Seq2Seq) consiste à former des modèles pour convertir des séquences d'un domaine (par exemple des phrases en anglais) en séquences dans un autre domaine (par exemple les mêmes phrases traduites en français).
Cela peut être utilisé pour la traduction automatique ou pour la réponse aux questions (génération d'une réponse en langage naturel à partir d'une question en langage naturel).
Dans le cas général, les séquences d'entrée et les séquences de sortie ont des longueurs différentes (par exemple, traduction automatique) et la séquence d'entrée entière est nécessaire pour commencer à prédire la cible.

Fonctionnement du modele :

- Une couche RNN agit comme "encodeur":
  - elle traite la séquence d'entrée et renvoie son propre état interne.
  - Cet état servira de "contexte", ou de "conditionnement", du décodeur à l'étape suivante.
- Une autre couche RNN fait office de « décodeur » :
  - elle est entraînée à prédire les caractères suivants de la séquence cible, compte tenu des caractères précédents de la séquence cible.
  - il est entraîné à transformer les séquences cibles en les mêmes séquences mais décalées d'un pas de temps dans le futur, un processus d'entraînement appelé "forçage de l'enseignant" dans ce contexte.
- L'encodeur utilise comme état initial les vecteurs d'état de l'encodeur, c'est ainsi que le décodeur obtient des informations sur ce qu'il est censé générer.
- En effet, le décodeur apprend à générer des cibles[t+1...] des cibles données[...t], conditionnées à la séquence d'entrée.



En mode inférence, c'est-à-dire lorsque l'on veut décoder des séquences d'entrée inconnues, on passe par un processus légèrement différent :

1) Encoder la séquence d'entrée en vecteurs d'état.
2) Commencer avec une séquence cible de taille 1 (juste le caractère de début de séquence).
3) Envoyez les vecteurs d'état et la séquence cible de 1 caractère au décodeur pour produire des prédictions pour le caractère suivant.
4) Échantillonner le caractère suivant en utilisant ces prédictions (utilisant simplement argmax).
5) Ajouter le caractère échantillonné à la séquence cible
6) Répéter jusqu'à ce que nous générions le caractère de fin de séquence ou que nous atteignions la limite de caractères.

1. Charger le jeu de donneees fra.txt a aprtir du site : www.manythings.org/anki/

Il s'agit d'un dictionnaire francais-anglais

```
path_to_data = 'D:/MyData/Cours/DM/Data/fra-eng/fra.txt'

translation_file = open(path_to_data,"r", encoding='utf-8')
raw_data = translation_file.read()
translation_file.close()
```

2. Segmenter le corpus en un ensemble de lignes avec la fonction suivante :

```
raw_data = raw_data.split('\n')
pairs = [sentence.split('\t') for sentence in  raw_data]
pairs = pairs[10000:20000]          # on garde que la collection de phrases d'index entre 10000 et 20000
```

3. Nettoyer le corpus par la suppresssion des ponctuations, etc…

```
def clean_sentence(sentence):
    lower_case_sent = sentence.lower()
    string_punctuation = string.punctuation + "¡" + '¿'
    clean_sentence = lower_case_sent.translate(str.maketrans('', '', string_punctuation))

    return clean_sentence
```

4. Tokenizer le corpus en un array de mots en utilisant la fonction suivante :

```
def tokenize(sentences):
    text_tokenizer = Tokenizer()
    text_tokenizer.fit_on_texts(sentences)
    return text_tokenizer.texts_to_sequences(sentences), text_tokenizer
```

5. Normaliser la longueuer de deux vocabulaires : francais et anglais

```
french_vocab = len(fra_text_tokenizer.word_index) + 1
english_vocab = len(eng_text_tokenizer.word_index) + 1

#padding
max_french_len = int(len(max(fra_text_tokenized,key=len)))
max_english_len = int(len(max(fra_text_tokenized,key=len)))

fra_pad_sentence = pad_sequences(fra_text_tokenized, max_french_len, padding = "post")
eng_pad_sentence = pad_sequences(eng_text_tokenized, max_french_len, padding = "post")

# Reshape data
fra_pad_sentence = fra_pad_sentence.reshape(*fra_pad_sentence.shape, 1)
eng_pad_sentence = eng_pad_sentence.reshape(*eng_pad_sentence.shape, 1)
```

6. Definir le modele Encoder

```
input_sequence = Input(shape=(max_french_len,))
embedding = Embedding(input_dim=french_vocab, output_dim=128,)(input_sequence)
encoder = LSTM(64, return_sequences=False)(embedding)
```

7. Definir le modele Decoder

```
r_vec = RepeatVector(max_english_len)(encoder)
decoder = LSTM(64, return_sequences=True, dropout=0.2)(r_vec)
logits = TimeDistributed(Dense(english_vocab))(decoder)
enc_dec_model = Model(input_sequence, Activation('softmax')(logits))
```

8. Compiler le modele, utiliser l'optimiseur 'adam', la fonction de perte sparse_categorical_crossentropy, et la metrique d'evaluation l'accuracy
9. Afficher le resume des parametres du modele
10. Executer le modele sur les deux vocabulaires, avec le nombre d'epochs =100, et batch_size=30

```
model_results = enc_dec_model.fit(fra_pad_sentence, eng_pad_sentence, batch_size=30, epochs=100)
```

11. En utilisant la fonction « logits_to_sentence », prédire la traduction des phrases d'index [8, 14, 100]

```
def logits_to_sentence(logits, tokenizer):
    index_to_words = {idx: word for word, idx in tokenizer.word_index.items()}
    index_to_words[0] = '<empty>'

    return ' '.join([index_to_words[prediction] for prediction in np.argmax(logits, 1)])
```