RNN.py

La classe RNN permet de stocker un objet RNN.

Structure de l'objet Morceau :

```
self.type = type                                    # type du RNN
self.input_list = input_list                        # liste des entree
self.lr = float(param_list[0])                      # taux d'apprentissage du RNN
self.nb_epochs = int(param_list[1])                 # nombre de cycles d'entraînement
self.len_hidden_dim = int(param_list[2])            # taille de la dimension cachée
self.nb_layers = int(param_list[3])                 # nombre de couches
self.seq_len = int(param_list[4])                   # longueur d'une séquence
self.is_batch = bool(param_list[5])             # entraînement sous forme de batch
self.batch_len = int(param_list[6])             # nombre de séquences dans un batch
self.nb_morceaux = int(param_list[7])           # nombre de morceaux à produire
self.duree_morceaux = int(param_list[8])    # longueur des morceaux

self.device = self.device_choice()              # choix de l'appareil
self.dict_int2val = None                # liste des dictionnaires de traduction de int vers val
self.dict_val2int = None                # liste des dictionnaires de traduction de val vers int
self.dict_size = None                   # liste des taille des dictionnaires

# définition du modèle, de la fonction d'évaluation de l'erreur et de l'optimiseur
self.model = None
self.criterion = None
self.optimizer = None
```

méthode one_hot_encode() :

Prend en paramètre une liste de séquences, la taille du dictionnaire, la longueur des séquences et la taille des batch
Retourne un array de dimensions 3 de taille (taille des batch, longueur des séquences, taille du dictionnaire) encodé avec la méthode du one-hot-encode.

méthode sample_seq() :

Prend en paramètres un nombre d'échantillons, le nombre total d'échantillons, une liste de torseurs input, une liste de torseurs target.
Retourne un échantillon de torseurs input et target associés parmi tous les échantillons torseurs.

méthode distribution_pick() :

Prend en paramètres un vecteur de probabilité
Retourne l'indice de l'élément choisi en fonction de sa probabilité.

méthode predict() :

Prend en paramètres une note
Retourne une note choisie par le RNN avec la fonction distribution_pick

méthode sample() :

Prend en paramètre une longueur de sortie et une liste de notes
Retourne une liste de notes de la longueur choisie complétée par le RNN

méthode device_choice() :

Renvoie le device utilisé pour l'entraînement du RNN en se basant sur l'existence ou non d'un GPU connecté à l'ordinateur

méthode training_file_number_choice() :

Prend en paramètre le nombre total des séquences d'entrée
Renvoie un nombre correspondant à 80% du total arrondi à l'entier supérieur (pour être sûr qu'au moins 80% des données d'entrée soient destinées à l'entraînement du RNN.

méthode training_file_choice() :

Prend en paramètre une liste de séquence d'entrée et le nombre de séquences à marquer comme séquence d'entraînement
Retourne une liste composée des listes d'entraînements et des listes de test

méthode decouper_morceau() :

Prend en entrée une liste de données d'entrée et une taille.
Retourne le même tableau où les morceaux ont été découpées en séquence de longueur taille+1.

méthode train() :

Ne prend pas d'argument
Ne retourne rien
Entraîne le RNN

méthode generate() :

Ne prend pas d'argument
Renvoie une liste contenant autant de morceaux générés que l'utilisateur a choisi

class Model():

Classe héritant du torch.nn.Module
Implémente la structure du RNN