Assignment

model 程式說明

一、model 包括程式和資料兩大部分。

1、程式部分

model 的程式部分由 rnn.py、tree.py 和 utils.py 三個指令檔組成。rnn.py 是整個 model 的主程序。tree.py 和 utils.py 配合 rnn.py 使用，完成相關的資料預處理工作，包括 binaryTree 的建立和詞典的建立。

（1）、rnn.py

rnn.py 完成 model 的資料載入、詞典建立、RNN 模型建立、監督訓練模型、預測和結果輸出共六種主要功能。其具體構成如下所示。

class Config():完成模型的參數配置，配置詞向量大小、情感標籤數目、退火速率、學習率、損失率以及反覆運算次數等主要參數的設定。

class RNN_Model():完成 model 的資料載入、詞典建立、RNN 模型建立、監督訓練模型、預測等主要功能。

RNN_Model.load_data():完成資料的載入和詞典建立的工作。資料的載入工作需要依賴 tree.py 腳本來完成，其處理過程是將資料集讀入 model 並處理為 model 需要的完全 binaryTree 資料類型。之後，再用 utils.py 根據完全 binaryTree 的資料類型建立詞典。

RNN_Model.inference():設定 model 是否處於只做根節點情感分析的工作模式，根節點的情感分析即為整句話的情感分析。

RNN_Model.add_model_vars():根據本檔中的class Config()裡的參數設定設置RNN模型的相關主要參數。

RNN_Model.add_model():根據 binaryTree 形式的資料結構為每個節點建立模型，並根據上述的 RNN_Model.add_model_vars()中對整個 RNN 模型的相關主要參數的設定來設置每個節點模型的參數。

RNN_Model.loss():根據本檔中的 class Config()裡的參數設定損失函數。

RNN_Model.predictions():完成情感分析的結果預測功能。

RNN_Model.run_epoch():完成 RNN 模型的每一步訓練過程的建立。用訓練資料集訓練模型，並根據驗證資料集的預測結果回饋調節更新模型的參數，並將模型參數保存，保存的檔案名為./weights/rnn_embed=XXXXXX_l2=XXXXX_lr=XXXXX.weights 裡。

RNN_Model.train():將 RNN_Model.run_epoch()的內容反覆運算執行，反覆運算的次數為 class Config()裡設定的最大反覆運算次數。反覆運算完成後，將達到要求的模型保存，以供後續的測試工作使用。

test_RNN():用於測試訓練好的 RNN 模型。將測試資料通過訓練好的模型進行預測，並將測試資料中每句話的情感預測結果用“1”和“0”逐行保存到./output/anwser.txt 裡。

以上就是整個 rnn.py 的工作構成。

（2）、tree.py

tree.py 主要完成資料的預處理工作。主要包括資料集的讀入、整理和生成完全 binaryTree。

class Node: 聲明 binaryTree 每個節點，節點的屬性包括情感標籤、詞語、雙親和左右孩子。

class Tree: 完成資料集的讀入、整理和生成完全 binaryTree。

Tree.newNode():聲明 binaryTree 的新節點，節點的屬性包括情感標籤、詞語、parents 和左右 child。

Tree.parse():解析 loadTrees()和 loadTestTrees()載入後的文本資料。解析工作完成後，生成不完全 binaryTree 的資料形式。

leftTraverse():用遞迴呼叫的方式左遍歷 binaryTree。

getLeaves():用遞迴呼叫的方式獲取每個 binaryTree 的葉子節點。

get_labels():用遞迴呼叫的方式獲取每個 binaryTree 的每個節點的標籤。

stripTestTreesLabel():將讀入的測試資料的 Bracket Labels 替換為數位形式，以便模型的處理。

pre_test_data():將讀入的測試資料處理為每行一句話的格式。

loadTrees()和 loadTestTrees():載入訓練和測試文本資料。

binarize_labels():將 binaryTree 每個節點的標籤二值化。

make_BinTree():將 Tree.parse()得到的不完全 binaryTree 整理為完全 binaryTree 的形式。

simplified_data():使用以上所有方法，完成整個資料集的預處理工作。

（3）、utils.py

utils.py 主要完成詞典建立和詞頻統計工作。

class Vocab(object):完成詞典建立和詞頻統計工作。

Vocab.add_word():完成將 binaryTree 中每個節點的單詞提取出來添加到詞典並統計詞頻的工作。

Vocab.construct()：通過使用 Vocab.add_word()完成構架詞典的功能。

Vocab.encode():完成將詞典裡的每個詞編碼建立索引的工作。

Vocab.decode()：完成將詞典裡的每個編碼索引到單詞的工作。

Bonus1

model 程式說明

Bonus1 的 model 程式與 Assignment 的基本相同。之間的不同在於文本編碼方式裡，Bonus1 要設定為 UTF-8、tree.py 裡的標籤替換需要使用 Penn Chinese Treebank，其餘基本相同