Pytorch 與機器學習 期末報告

107306052 資管三 詹筑婷
107306065 資管三 夏秋如

主題：UberEats 消費者評價情感分析

壹、前言

一、研究動機

2020 年疫情爆發，懶人經濟逆勢成長，美食外送服務正在臺灣如火如荼地發展中，成長速度快到一週的訂單量高達過去 6 年的總和，而相較於空腹熊貓等平台根植於「訂餐服務」，UberEats 是以「運輸業」的角度切入，當訂單暴漲時，前者的排班制如同傳統外送業只能縮小範圍因應，難以消化全部訂單，但後者會以獎勵制度鼓勵外送員上線，吸引鄰近外送員的支援，也使其成為本次研究對象。

獎勵制度影響了外送員的接單率，而接單率的高低仰賴著「評價系統」的每筆數據，根據自身操作經驗，發現在評價整體訂單的時候，UberEats 分成兩個階段，首先是針對餐廳的餐點，再來才是外送員的服務，前者有五顆星星的級距可選擇，後者只有好跟壞（讚與倒讚）的兩個選項，最後才是額外的評語輸入（也可選擇略過），這種等第選擇的方式雖然快速又便利，但就外送員來說太過兩極，所以我們希望透過直接分析文字評論的方式，增加區分好壞的彈性，以分數判斷負評的嚴重程度，藉此降低個別評價對外送員接單率的影響。

二、研究方法

我們採用人工智慧中極具潛力的一種技術 — 自然語言處理（Natural Language Processing），它透過複雜的數學模型及演算法來讓機器去認知、理解並運用我們的語言。而情感分析便是 NLP 其中一項應用，是一種挖掘文字或話語意見的方式，設立規則將詞彙量化，由此得知字句背後的情緒、意見或意圖，隨著這項技術更加成熟，業者能夠應用它去更好的理解用戶或是消費者的真實感受。

貳、正文

本次目標是建立一個模型，用於判定輸入語句的正反面情緒，以下以建置步驟說明：

檔案說明：

名稱檔案	說明
W2V_SV.py	轉換句子到句向量類
make_w2v_set.py	製作訓練所需的檔案 （資料集文字轉換向量）
JWP.py	神經網路定義類
jwp_train_bce.py	模型訓練
demo.py	模型示範

一、取得文字向量

首先，需要先將文字進行各種轉換或是標記（數值化），這邊使用了 jieba 將我們的 input 句子斷詞，然後使用 word2vec 取得了各單詞的向量，最後加總平均當作句向量。

使用以 wikidata 訓練好的 word2vec 模型，將「字詞」轉換成「向量」形式，計算向量的相似遠近程度，以距離範圍 0~1 的餘弦值表示，值愈大代表兩個詞關連度愈高，用來表示文本語意上的相似度。基於非監督學習的 word2vec，為了銜接後續的資料處理，這邊採用的是基於 python 的主題模型函式庫 gensim。

```python
from gensim import models
import jieba
import numpy as np
class W2VS():
    def __init__(self):
        """
        初始化加載
        """
        jieba.set_dictionary('dict/dict.txt.big')
        jieba.load_userdict('dict/my_dict')
        jieba.initialize()
        self.model = models.Word2Vec.load('w2vmodel/word2vec.model')

    def getSenVec(self,sentence):
        """
        取得單詞向量
        單詞向量相加平均
        返回向量(句子向量)
        """
        senCut = list(jieba.cut(sentence))
        lenOfCut = len(senCut)
        vecSum = np.zeros(200)

        for i in senCut:
            try:
                vec = self.model.wv.__getitem__(i)
                vecSum = np.add(vecSum, vec)
            except Exception as e:
                # print(e)
                lenOfCut -= 1
                continue
        if(lenOfCut == 0 ):
            return np.array([0]*200)
        divisor = np.array([lenOfCut]*200)
        return np.divide(vecSum, divisor)

if __name__ == "__main__":
    w2vs = W2VS()
    print(w2vs.getSenVec("店員很好吃嘀"))
```

```
[ 0.45786192 -0.07470008  0.1502905  -0.25330304  0.24774257 -0.65835389
  0.31488902  2.0542151  -0.94664764  1.00015236 -1.61205989 -0.63732188
  0.10256875 -0.53561275 -1.70897487  0.88763857  0.49992362 -0.78507867
  1.22246256  2.08869775  1.08996487  0.31891896  0.96248941  0.03739308
 -0.17019836 -0.21653467  0.83457948  0.18157801  0.70209159 -0.81508517
  0.26320085 -0.44651277  0.61766971  0.53260716  0.62443142 -1.50824907
 -0.46937907  0.78625704  1.47923735  1.16358182  0.57237408  0.43453781
  0.67475009  0.48513844  0.58344632 -1.24982427 -0.78708654 -0.73030764
  0.57393602  0.83043577  0.49221223 -0.63637896 -0.05860114 -0.03480168
  0.00983298  0.77102925  1.96028933  0.25047025  0.23368012  0.26252083
  0.32608465 -0.6810274   0.57170062 -0.86005161  1.79488692  1.76245546
  0.21460059  1.15373492 -0.07155022 -0.07280043 -0.04224822 -0.61802446
 -0.56948876 -0.11242693 -0.8013178   0.09972757  0.12603272  0.18164457
 -1.17456273  0.38941246  1.13060888 -0.75168352  0.20820692  0.71620353
  0.66979489  0.10764683  0.24059446  1.99034771  0.41859604  0.14338912
 -1.56440806 -0.10153789 -0.38643036 -0.99798806 -0.47862824  0.04135695
 -0.26673511  0.0035699  -0.74779186  0.18402238 -0.94694877 -0.21867154
 -0.57045451  0.89766153 -0.5034408  -0.09282572  0.12236389 -0.48515353
 -1.12486209  0.23258572  0.38904333  0.27684641  0.53491479  0.71864267
 -1.50302359 -1.13970366  0.34762137  1.56032263 -0.56279003  0.46985877
 -1.21144862  0.88709038 -1.8615815   0.74474016  0.04672603  0.25407184
  2.08648856  0.18446395 -0.40502116  0.50735891 -0.85486636  0.01849119
 -0.37999577 -0.14663075 -1.55770668  0.03840621  1.12041423 -0.04845363
 -0.4332563  -0.52354267  0.47797246  0.18370662  0.46730693  0.50679389
  0.06243071 -1.14440785 -1.05858557 -1.33818853 -1.39115293  0.33173263
  0.42654034  1.11014736 -0.33945492  0.41251229 -1.03620458  0.53521668
  0.14997269  0.06007987  0.94109575 -0.17892811 -0.09449878 -1.06710756
  0.33157719 -1.45984018  1.08488614  0.7603651   1.58175452  0.9286578
 -0.16071606  0.58175988  0.12931482 -1.73134457  0.59387099 -0.33477458
 -0.40765236 -0.6663638   0.32049655 -1.3654165  -1.16992425 -0.32060837
  0.44674559 -0.63455815 -0.76914805  0.08118507 -0.43589522 -1.83649555
 -0.35456163 -0.64466892 -0.2481386   0.74133875  1.17212324 -0.82107031
  0.87027926 -0.00680378 -0.45805325 -2.07135086  1.45857141  0.01916073
 -0.76159421 -0.19891775]
```

此為例句「店員很好吃嘀」的 200 維度向量。

← W2V_SV.py

二、準備訓練資料

資料集採用中國某外賣平台收集的用戶評價（正向 4000 條，負向 8000 條）：
https://github.com/SophonPlus/ChineseNlpCorpus/blob/master/datasets/waimai_10k/intro.ipynb

使用前面的 W2VS.py 將資料集讀入，並且轉換成句向量，存成 .pkl 檔供稍後使用。

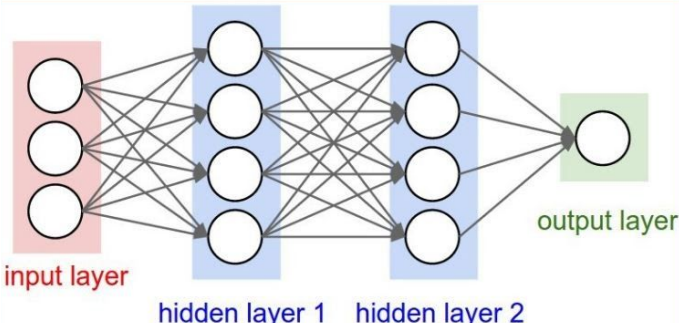
```python
from W2V_SV import W2VS
import pickle
import csv

if __name__ == "__main__":
    w2vs = W2VS()
    sentencesDict = {}
    with open('dataset/waimai_10k_tw.csv',newline='',encoding="utf-8") as f:
        rows = csv.reader(f)
        for i,row in enumerate(rows):
            if(row[0] == 'label'):
                continue
            line = row[1].strip('\n')
            sVec = w2vs.getSenVec(line)
            # sentencesDict.append(sVec)
            sentencesDict[str(i-1)] = (sVec,row[0])
    with open('dataset/waimai_10k_tw.pkl','wb') as f:
        pickle.dump(sentencesDict,f)
    print("finish")
```

↑ make_w2v_set.py

三、定義神經網路模型

接著使用 PyTorch 開始建立我們自己的神經網路模型，結構為一個有兩個隱藏層的模型，中間兩層使用 Relu 做輸出函數。

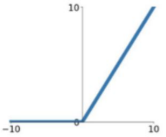
模型層次：



激活函數：

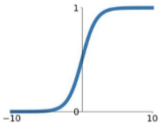
ReLU

$\max(0, x)$



Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



- ReLU（rectified linear unit）函數提供了一個很簡單的非線性變換，它是把負數先轉為零，正數就什麼都不做直接離開node。用意是把負值關係排除掉。

- apply_sigmoid 參數是給最後訓練完成應用時使用的，訓練時不應該在 output 使用 sigmoid，它會處理負數的 output，把負數的輸出規範在 0～0.5 之間，而正數的輸出規範在 0.5～1。即所有 x+y-b 都會規範在 0～1之間。

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class JWP(nn.Module):
    def __init__(self, n_feature, n_hidden,n_hidden2, n_output):
        super(JWP, self).__init__()
        self.hidden = nn.Linear(n_feature, n_hidden)
        self.hidden2 = nn.Linear(n_hidden, n_hidden2)
        self.out = nn.Linear(n_hidden2, n_output)

    def forward(self, x, apply_sigmoid=False):
        x = F.relu(self.hidden(x).squeeze())
        x = F.relu(self.hidden2(x).squeeze())
        #
        if(apply_sigmoid):
            x = torch.sigmoid(self.out(x))
        else:
            x = self.out(x)

        return x
```

↑ JWP.py

四、訓練模型（以下程式碼皆由 jwp_train_bce.py 中擷取）

<u>損失函數</u>：

接下來就是定義損失函數，因為分類問題的目標變量是離散的，不像迴歸問提是連續的數值，所以就用分類問題常用的損失函數為交叉熵（Cross Entropy Loss），交叉熵描述了兩個概率分布之間的距離，當交叉熵愈小說明二者之間愈接近。

```
loss_func = torch.nn.BCEWithLogitsLoss()
```

$$loss(z,y) = mean\{l_0, \dots . l_{N-1}\}$$

$$l_n = -(y_n * log(\delta(z_n)) + (1-y_n) * log(1-\delta(z_n)))$$

N：樣本數

Z ：表示預測第 n 個樣本為正例的得分

y：表示第 n 個樣本的標籤

sigma：表示sigmoid函數

這類整合性的 BCEWithLogitsLoss( )，比純粹使用 BCELoss( ) + Sigmoid( ) 數值更穩定。

Optimizer：

再者，就是透過優化器來更新模型中的參數，使得損失函數中的 Loss 降低：Adam（Adaptive Moment Estimation）是一種自適應學習率的方法，適用於大數據集和高維空間的資料，優點主要在於經過偏置校正後，每一次迭代學習率都有個確定範圍，使得參數較平穩。

```
optimizer = torch.optim.Adam(net.parameters(), lr=lr)
```

- Training set ： Validation set ＝ 8：2
- Batch size ＝ 200
- Epoch ＝ 50

學習率設定與調整：

```
args = Namespace(
    dataset_file = 'dataset/waimai_10k_tw.pkl',
    model_save_path='torchmodel/pytorch_bce.model',
    # Training hyper parameters
    batch_size = 200,
    learning_rate = 0.009,
    min_learning_rate = 0.001,
    num_epochs=50,
)
```

```
lr = args.learning_rate
min_lr = args.min_learning_rate
def adjust_learning_rate(optimizer, epoch):
    """
    調整學習率
    """
    global lr
    if epoch % 10 == 0 and epoch != 0:
        lr = lr * 0.65
        if(lr < min_lr):
            lr = min_lr
        for param_group in optimizer.param_groups:
            param_group['lr'] = lr
```

最好的學習率介於 0.001～0.01之間，所以設定初始值＝0.009，每一輪以 * 0.65 調整，直到已經達到設定的最小學習率＝0.001 為止。

```
for t in range(EPOCH):
    """
    動態調整學習率
    """
    adjust_learning_rate(optimizer,t)

    """
    Train phase
    """
    net.train() # 訓練模式
    TrainAcc = 0.0
    TrainLoss = 0.0
    # Train batch
    for step,(batchData, batchTarget) in enumerate(trainDataLoader):
        optimizer.zero_grad() # 梯度歸零
        out = net(batchData)
        trainAcc = compute_accuracy(out,batchTarget.long()) # 取得正確率
        TrainAcc = TrainAcc + trainAcc
        loss = loss_func(out,batchTarget) # Loss 計算
        TrainLoss = TrainLoss + loss
        loss.backward() # 反向傳播
        optimizer.step() # 更新權重
    TrainLoss = TrainLoss / (step+1) # epoch loss
    TrainAcc = TrainAcc / (step+1) # epoch acc
```

訓練模型時會動態的調整學習率，頻率為每個 epoch 調整一次。

正確率：

```python
def compute_accuracy(y_pred, y_target):
    """
    計算正確率
    """
    y_target = y_target.cpu()
    y_pred_indices = (torch.sigmoid(y_pred)>0.5).cpu().long()#.max(dim=1)[1]
    n_correct = torch.eq(y_pred_indices, y_target).sum().item()
    return n_correct / len(y_pred_indices) * 100
```

此處會根據 sigmoid 判斷 output 值的範圍介於 0～0.5 或是 0.5～1 之間，再與 batchsize 算出比率，計算模型準確度。

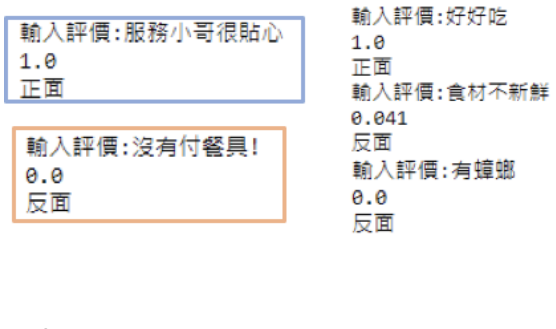
五、學習結果與示範

經過 50 次訓練跑完，訓練結果為：

```
epoch: 50 train_loss: 0.027 train_acc: 99.217 test_loss: 0.022
test_acc: 99.467 LR: 0.0016065562500000002
```

示範：

```python
import torch
from JWP import JWP
from gensim.models.doc2vec import Doc2Vec
import torch.nn.functional as F
import torch
import numpy as np
from W2V_SV import W2VS

print("init...")
w2vs = W2VS()
net = torch.load('torchmodel/pytorch_bce.model')
net.eval()
# test_data
while True:
    ts = input("輸入評價:")
    v1 = w2vs.getSenVec(ts)
    res = net(torch.FloatTensor(v1), apply_sigmoid = True)
    out = res
    res = res.clone().detach().numpy()[0]
    print(round(res,3))

    if(res>0.5):
        print("正面")
    else:
        print("反面")
```



← demo.py

當輸入評價後就會予以介於 0～1 的分數，並標示出為正/負面評價。

參、結論

改善空間：

由於目前所選資料集（Waimai_10k）包含針對店家、餐點或是外送人員的所有評價，未來若是能區別文字內容針對的對象，過濾掉不屬於外送員的評語（ex：食材不新鮮、餐點很美味），使整個評價系統更有說服力。

除了增加資料量之外，也可加入 dropout rate 進行正規化，避免模型過擬合。

肆、參考資料

- 如何成為外送員
  https://www.readr.tw/project/food-delivery/order3

- 中文自然語言處理數據集
  https://kknews.cc/code/lmxlqp9.html

- 簡單談談 Cross Entropy Loss
  https://blog.csdn.net/xg123321123/article/details/80781611?utm_medium=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromBaidu-1.control&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromBaidu-1.control

- 以 gensim 訓練中文詞向
  http://zake7749.github.io/2016/08/28/word2vec-with-gensim/

- 優化方法總結：SGD，Momentum，AdaGrad，RMSProp，Adam
  https://blog.csdn.net/u010089444/article/details/76725843?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-8.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-8.control

- NLP 自然語言處理
  https://oosga.com/nlp/