NLP Homework 3

@author A10515001 李大祥 2018-04-26

作業3-1和處理資料

第一個作業基本就正常按照投影片做，用 Word2Vec 這個函數生成 model 。後來根據kaggle上面同學分享的經驗改進了一下：用pkl存儲所有文本。
計算的“日本”和“臺灣”的兩個詞的相似度 : 0.675。與“日本”最像的詞裏面有：賞櫻 0.649、旅遊 0.649、去過 0.635、京都 0.634、東京 0.633 等等。

作業3-2

神經網絡模型

正確的概念上的理解應該是把每一篇文章作爲一筆輸入資料 ，把一個 10 唯度的類別向量作爲一個對應的 label 。這樣訓練集有 9000 筆資料，測試結果同理有 1000 筆資料。而每筆資料中的文章，應該是這篇文章裏每個切割後的字詞的 word embedding 向量。

第一層是 embedding_layer ，輸入的是 250 X 232236 的字典，輸出是 3971, 250 格式的一篇文章的 word_embedding 格式。RNN 層設定爲壓縮五倍。Dense 層只是中規中矩，輸出 10 維度的類別 vector。因爲是多類別分類，所以使用 softmax 的 loss function。

訓練實作部分

第一訓練階段

訓練過程中，第一次訓練只訓練一次，結果發現分類都是一個類別，還以爲自己寫錯了，後來嘗試過10次以後發現，其實是訓練太少的原因。

正式訓練時，增加 epochs 到 50 次，修改 batch_size 到 100 次，做一次正式訓練，kaggle 結果顯示 0.12 左右。大概花費時間 3 個小時左右。

```
validation_split=0.1, batch_size=100, epochs = 50
```

再後來增加訓練次數，epochs 增加繼續訓練到 150 次，並增大 batch_size 的值到 150， kaggle 結果有明顯提升到 0.13 。這證明增加訓練次數是有效果的。batch_size 大一些，會增加梯度下降的準確性，提升模型預測的性能。

```
validation_split=0.1, batch_size=150, epochs = 150
```

epochs 增加繼續訓練到 250 次，kaggle 結果有明顯提升到 0.14 。這證明增加訓練次數是有效果的,但是效果很有限。

```
validation_split=0.1, batch_size=150, epochs = 350
```

第二訓練階段

考慮到在上面一階段訓練中增加訓練次數的效果是很有限的，僅僅才0.14，所以想到一些方法來提升效果。我本來想清洗一下無用的英文字串資料數據, 並多訓練 word_embedding 的 model 結果，看看這樣效果會不會變好。但是後來請教了同學，才發現其實重點是我的神經網絡模型和資料處理有問題，於是我先處理這些重要問題。

主要問題在 **3** 個方面：

1. 處理輸入時忽略了不在字典中的字詞，加之我把所有文章補成了 3971 最長長度（實際上不應該這樣，因爲一多半的文章長度連300字詞都沒有），所以這樣子結果都不會很好。所以我修改所有文章長度統一爲 250 ，然後把不在字典

中的字詞設爲 0 vector 。
2. 訓練資料應該打亂重排序，減少規律排序對訓練的不良影響。現實中的 test 資料一般不會幫你實現排好序、劃分類群。
3. 字典構建缺少了測試資料集。

```
## 解决第一個問題，統一每個文章長度爲 250
## 解决第二個問題，讀取資料時，打亂原有排序
train_df_sample = pd.read_pickle('train.pkl').sample(frac=1, random_state=123)
## 解决第三个问题： 字典构建缺少了测试资料集，而且要用同一个 tokenizer 去构建字典
```

並重新調整 batch_size、max_doc_word_length、epochs。

```
# kaggle: 0.58199
max_doc_word_length = 250
validation_split=0.1, batch_size=3000, epochs = 100

# kaggle: 0.61
max_doc_word_length = 250
validation_split=0.1, batch_size=1620, epochs = 200
```

思來想去，覺得之前訓練的 WordEmbedding 可能太差了，所以又去刪掉多余 14 長度的無用英文單詞，並反復試了幾個參數，把單詞之間相似讀提高了。

```
model = Word2Vec(corpus, size=250, iter= 8, workers=3, min_count=5, negative=3, max_vocab_size=None, window=5)
```

再之後我去檢查了字詞出現的次數，發現大概15萬的字詞都是少於3次的，我嘗試了刪去這些字詞，把總參數從5000多萬降低到：10,474,876。但發現結果並沒有很好，而且很難提升，所以也就沒有去kaggle測試效果。

```
validation_split=0.1, batch_size=405, epochs = 400
```

所以後來還是回歸原先的模型，只是刪除了多余 14 長度的無用英文單詞和加入測試資料集後重構了字典，其余保留。

```
# kaggle: 0.79700
validation_split=0.1, batch_size= 1620, epochs = 400
```

兩個訓練階段的總結

1. embedding_layer 必須作爲第一層輸入，它的內容是字典裏所有的 token 對應的 vector 。這個字典裏所有的 token 一般來說是不重復的，是我們能給出的訓練集裏出現過的字詞。本次作業中，我設定了每個字詞的 word_embedding 是 250 維的 vector ，所以這個 embedding_layer 的shape 應該是 250 X 232236 。
2. 喂給神經網絡模型的 fit 函數的應該是文章序列以及對應的label，維度都是 9000 X 1 和 9000 X 10 。文章序列裏每篇文章的格式應該是數字，準確的說是每個字詞在上述字典裏的 ID 值。在訓練的時候，**keras** 會自動在第一層 **embedding_layer** 中把這個值轉換成 **word_embedding** 的**vector** 格式 ，所以不用我們自己轉換成 word_embedding 後再輸入給 fit 函數。
3. 由於 keras 和電腦本身更加適應統一長度的輸入，所以實作中需要把長度不一的文章填補一些 data ，以形成統一長度的格式，本次作業中統一填補成 3971 長度。因爲最長的文章就是只有 3971 個字詞。
4. 開始我是對每篇文章的每個切割後字詞的 word_embedding 加總 取平均後 去訓練，但是這樣預測出來結果都是一個分類，明顯錯了。後來請教助教的時候發現應該是壓縮取平均的過程中 把每篇文章的差異性消除了，所以才會導致分成一個類別。
5. 因爲訓練資料集爲 9000 ，而且又是使用 RNN 訓練分類 ，所以 batch_size 不應該過小（比如 150 ），否則會出現訓練出來後幾乎無法分類、分類準確率很低。
6. max_doc_word_length 設置爲 250 ，是因爲算過所有訓練文章的平均長度，差不多在 210 左右，但是也有幾百篇文章是超過 800 多個字詞甚至三千多字詞的，所以設置爲 200 ～ 300 左右還算合理。
7. epochs 次數大概在 200 ～ 400 左右會有一個比較好的 model 結果，只訓練一兩次用來測試幾乎無法分類。
8. 出現問題最多的在於如何處理資料，比如去掉雜訊、亂序重排、構建字典要足夠完整並且不能構建兩個否則字詞id會亂掉。keras 自帶的 library 很好用，但是也很容易用錯，要多看文檔。