Trending Article Prediction Report

Author: 張承遠

如何跑我的code?

zip檔裡包含: train.py predict.py requirement.txt Report.pdf ran100model_3.h5

- For Training:

```
python3 train.py {database_host} {model_filepath}
ex: python3 train.py 35.187.144.113:5432 ran100model_3.h5
# 程式運行完後會在當前路徑產生model.h5檔案，內含有model參數
```

- For Testing:

```
python3 test.py {database_host} {model_filepath} {output_filepath}
ex: python3 test.py 35.187.144.113:5432 ran100model_3.h5 output.csv
# 使用train.py產生的model.h5檔案，並且把資料存在output.csv
```

方法及為什麼要這樣做?

Analysis:

當拿到一筆資料時，我會先觀察data的分佈大概長怎樣，我從features及labels來分析：

- Features:

  關於這筆資料的Features，我認為是文章發布後十小時內share, comment, liked, collect的趨勢。因此，我們可以設一個threshold來切割feature.

  ex:若把threshold的offset設為5，則可以把feature分為: 發佈0-5小時後的狀況, 發布0-10小時候的狀況, 狀況指的是share, liked, comment, collect的累加數量)

- Labels:

  這筆資料的Labels，很明顯就是like_count 是否≥1000，若是則為1，否則為0

Model decision:

在觀察 train_posts 及 test_posts 的label的分佈後，可以很明顯發現實際上為熱門文章的數量非常少，也就是這筆資料產生了數據不均衡的問題。

在上網搜尋後，我發現使用有一些解決方法，如下：

- Sampling:

  把資料的label比例用到1:1

- Ensemble Method:

  使用Ensemble Method，ex: Random Forest

再嘗試過兩種方法後，我發現用Ensemble Method的表現比較好，因此最終使用 Random Forest (100棵decision Tree)的方法

在實驗過每個threshold後，最後發現 threshold的offset設為3表現較好。

Evaluate 在我們提供的 testing data 的結果:

Confusion Matrix: ([0, 1])

```
Confusion Matrix:
[[220969    510]
 [  2352   2155]]
```

F1 Score for two label:

```
F1 score: label=0
0.9935656474820144
F1 score: label=1
0.6009481316229783
```

Accuracy:

```
Accuracy: 0.987335498659209
```

實驗觀察:

在試過多種方法 (ex: Random Forest, Decision Tree, Logistic Regression, Naive Bayes...) 後，以及嘗試了不同的threshold，其實結果都相近的，但Random Forest的方法還是表現較佳。

由於數據不均衡的關係，若是在 test set 中預測每篇文章都為非熱門文章，準確率還是可以很高，但是這樣做沒有意義，因此我試著用Ensemble的方法解決這個問題。

在做Random Forest的時候，我也試著drop 不同的 feature，但結果仍然差不多，因此最後還是每個feature都用上去。

總結來說，這次的作業雖然並不是太複雜，但是並沒有明確的給feature，而是要自己從其他資料創造features來做training，這是一個跟平常一般作業比較大的差別，也讓我收穫良多。