資訊科學系專題研究

題目：
Chatbot 平台建置文章推薦系統

國立政治大學
資訊科學系專題研究
姓名：蘇裕勝

1 摘要

此研究為研究推薦系統的建立，以及背後演算法的優化，
使用 Nodejs 的 express 架構並連接 Facebook 的 API，在 Facebook 上建立 Chatbot，此 Chatbot 的功能是推薦給 Facebook 上不同屬性的使用者不同性質的文章閱讀。主要研究是針對不同的資料集以及資料型態（文本資料,類別資料），找到最佳推薦方式，結論為以下三者：

1. LSA 分析文本資料找出與使用者所閱讀過文章相似度高的推薦。
2. 使用兩層的機器學習演算法 (First layer: Random Forest, Extra Trees, GradientBoosting, Second layer: Logistic Regression)做預測，採用資料集為類別資料，然後將排名高的文章推薦給使用者。(這裡的資料集使用，[2]所得到的文章相似度結果，以及 Facebook 所提供的 API 抓取使用者的: ID, Gender, Location, Time, readHistory 等等….作為資料集，Feature 的塞選是很重要的一環，但這我們先不談 Feature 的塞選以及前置處理，主要著重在演算法的選擇以及模型的優化。)
3. 最新發佈的文章，實驗顯示至少有 è0%以上的使用者，會點選最新發佈的文章

研究結果已實作完成（如圖），目前也持續優化中，專案以及相關程式可以經由 github 下載：https://github.com/CoolSheng/FacebookChatBot

個股介紹

美股清單

美股清單

更多公司資訊

3M | Abercrombie - Fitch | Adobe

Type a message...

瀏覽最新文章

Google 自曝無人駕駛核心技術 一探 700 億估值 Waymo 的煉成之謎
相關報導: 2017/10/30
www.stockfeel.com.tw

閱讀此文章

Share

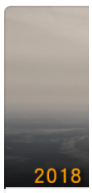
回首頁

輝達在 AI 領域的護城河是什麼？
相關報導: 2017/10/28
www.stockfeel.com.tw

閱讀此文章

Share

回首頁



2018 征服綠
相關報
www.s

Type a message...

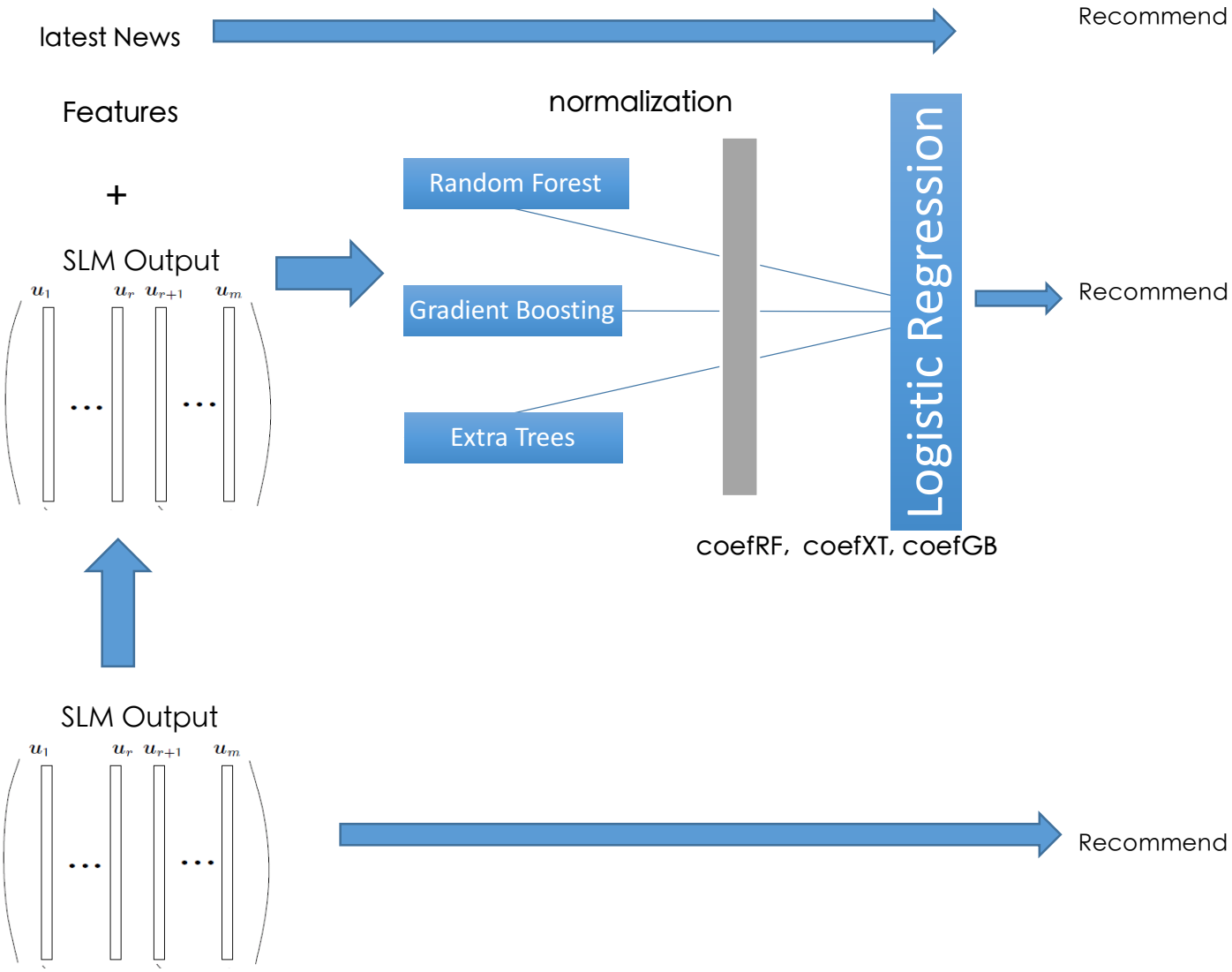
Type a message...

2 動機

因為現在市面上做推薦的系統非常多，不論是做廣告推薦，商品推薦等， 此研究是希望能夠建立一個好的文章推薦系統，並更了解如何訓練以及優化背後的推薦演算法。

3 實驗流程與方法

Recommend Framework



3.1 文本資料分析：LSA

使用爬蟲程式（參考 paser*.py） 從 stock feel 股感網站 https://www.stockfeel.com.tw，抓下來所有財經相關文章，當成文本訓練，以及資料集使用。
使用 LSA 語意分析方式，找出與使用者所閱讀過相似度高的文章（參考：LSM/lsm.py , LSM/lsm_ gensim.py）

3.1.1 Jieba 斷字斷詞處理

- 載入繁體詞典+自定義詞庫
  除了 Jieba 所預設的字詞外，還添加了更多財經相關的字。
- 取出文章中的關鍵詞
  import jieba.analyse 取出關鍵字
- 關鍵詞去除停用字
  jieba.analyse.set_stop_words()

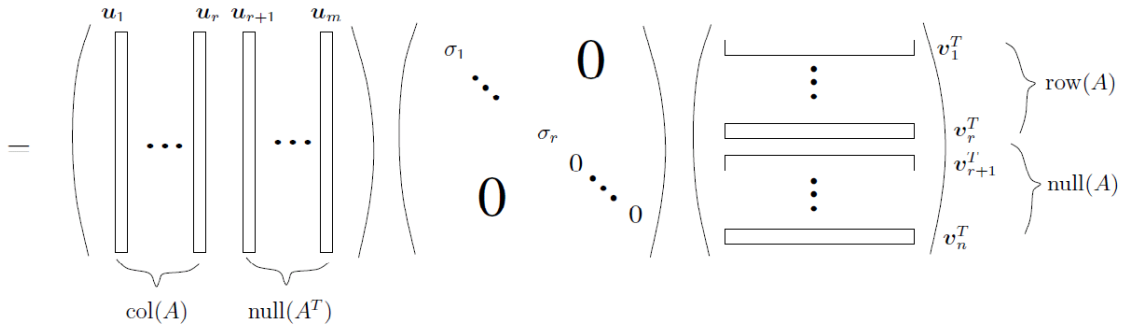
3.1.2 將每篇文章轉成向量表示(doc2vec)

將處理好的字，去掃全部的文本，然後做成 doc2Vec

❖ 註：但是最後採用 gensim，因為訓練的資料集夠大效果較佳(參考 LSM/lsm_ gensim.py)

3.1.3 singular value decomposition 降低數據維度

$$\underset{m\times n}{A} = \underset{m\times m}{U}\ \underset{m\times n}{\Sigma}\ \underset{n\times n}{V^T} \tag{1}$$



u, s, vt = linalg.svd(lyrics_dataset_vec)

使用 singular value decomposition 降低資料雜訊

3.1.4 計算 Cosin similarity

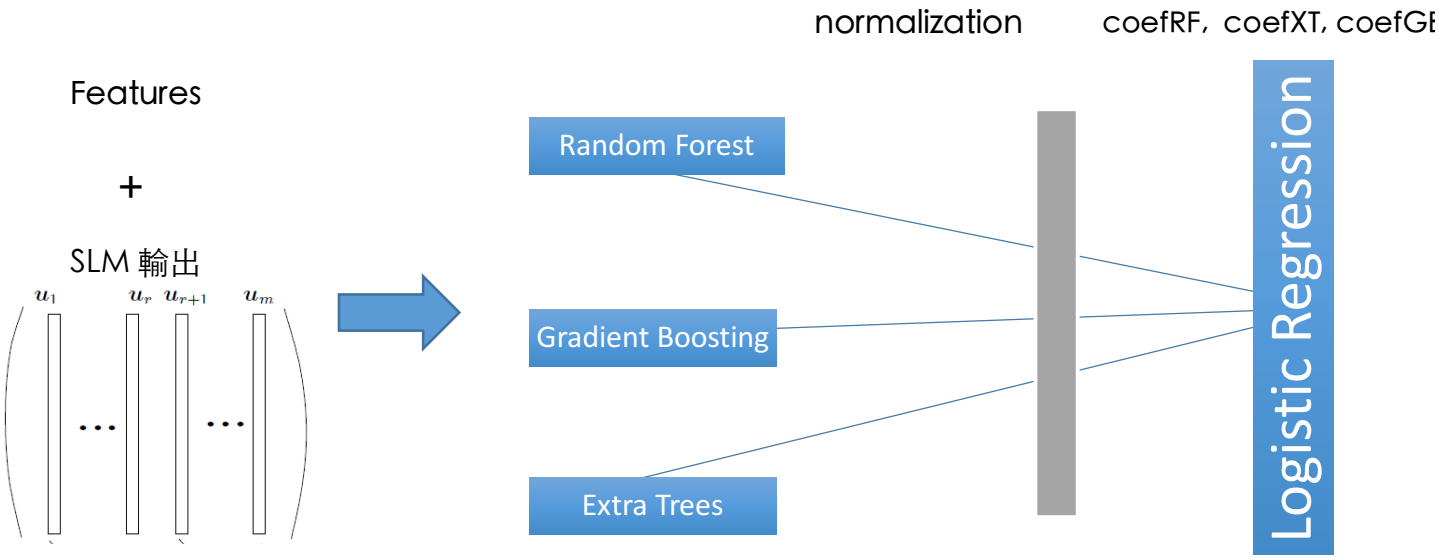
$$similarity = \cos(\theta) = \frac{A\cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

spatial.distance.cosine(low_rank_document_term_vec, vec)
最終得到的結果如這樣：
[(53, 0.99999994), (19, 0.96728575), (43, 0.96589249), (33, 0.96417892), (28, 0.95892304).....]
(文章 id 編號　，　此文章與 No. id 文章的相似度)
這樣很容易找出與此文章相關的文章排名。

3.2 類別資料分析：兩層機器演算法模型

3.2.1 機器演算法架構

第一層先使用各個基本的模型進行嘗試，如 KNN、Decision Tree、Naïve Bayes、SVM、Random Forest、AdaBoost、Extra Trees、Gradient Boosting，選最佳的三個模型（下文會提到這三個，分別為：Random Forest, Extra Trees, Gradient Boosting），用這三個模型的產出 Logistic Regression ：F(x)= coefRF*rf(x)+ coefXT*xt(x)+ coefGB*gb(x) 的參數(coefRF, coefXT, coefGB)，然後做最終的預測。（在做預測時，特徵值得處理以及選擇很重要，但這邊的研究主要著重在於模型的建立以及優化，所以暫且不討論）



3.2.2 第一層模型的選擇

先把目前比較常用常看到的分類模型拿進去測試，並得到以下結果。

Model	Accuracy
Random Forest	0.760830527497
Extra Trees	0.72379349046
Gradient Boosting	0.714814814815
SVM	0.709203142536
Decision Tree	0.660942760943
Naïve Bayes	0.673288439955
K Nearest Neighbor	0.641863075196

從上面的數據可以得知，比較傳統的模型效果相對比較差一些，如簡單暴力的 KNN、naïve Bayes、或是未經改良的 Decision Tree。相對的，許多改良 Decision Tree 的方式都可以得到一些比較好的結果，如 AdaBoost 及 Gradient Boosting 透過對於分類結果權重的改變去增進 Tree 的準確度，又或者 Extra Trees 及 Random Forest，透過 Feature 的選擇去剪裁出最好的 Tree。

而實際上，若將這些比較不準確的 Model 結果放入，將產生非常嚴重的負面影響，因此最終我僅將 Random Forest、Extra Trees 以及 Gradient Boosting，用這三個模型的產出 Logistic Regression ：F(x)= coefRF*rf(x)+ coefXT*xt(x)+ coefGB*gb(x) 的參數(coefRF, coefXT, coefGB)。

3.2.3 三個 Model 的參數調整

由於參數測試非常耗時，因此使用 sklearn 套件，透過自動化的方式進行參數測試。

3.2.4 Cross-Validation

```
# === Combine Models === #
# Do a linear combination using a cross_validated data split
X_train, X_cv, y_train, y_cv = cross_validation.train_test_split(X, y, test_
size=0.5, random_state=SEED)

modelRF.fit(X_cv, y_cv)
modelXT.fit(X_cv, y_cv)
modelGB.fit(X_cv, y_cv)

predsRF = modelRF.predict_proba(X_train)[:, 1]
predsXT = modelXT.predict_proba(X_train)[:, 1]
predsGB = modelGB.predict_proba(X_train)[:, 1]
preds = np.hstack((predsRF, predsXT, predsGB)).reshape(3,len(predsGB)).trans
pose()
preds[preds>0.9999999]=0.9999999
preds[preds<0.0000001]=0.0000001
preds = -np.log((1-preds)/preds)
modelEN1 = linear_model.LogisticRegression() #!!!!!!!!!!regression
modelEN1.fit(preds, y_train)
print(modelEN1.coef_)

modelRF.fit(X_train, y_train)
modelXT.fit(X_train, y_train)
modelGB.fit(X_train, y_train)
predsRF = modelRF.predict_proba(X_cv)[:, 1]
predsXT = modelXT.predict_proba(X_cv)[:, 1]
predsGB = modelGB.predict_proba(X_cv)[:, 1]
preds = np.hstack((predsRF, predsXT, predsGB)).reshape(3,len(predsGB)).trans
pose()
preds[preds>0.9999999]=0.9999999
preds[preds<0.0000001]=0.0000001
preds = -np.log((1-preds)/preds)
modelEN2 = linear_model.LogisticRegression()
modelEN2.fit(preds, y_cv)
print(modelEN2.coef_)
```

```
coefRF = modelEN1.coef_[0][0] + modelEN2.coef_[0][0]
coefXT = modelEN1.coef_[0][1] + modelEN2.coef_[0][1]
coefGB = modelEN1.coef_[0][2] + modelEN2.coef_[0][2]
```

CV 完後，找到最佳的 coefRF, coefXT, coefGB

3.2.5 結果

最後把做 F(x)= coefRF*rf(x)+ coefXT*xt(x)+ coefGB*gb(x)的三數帶入，得到預測的結果。推薦給使用者閱讀，最好可以達到 83%被閱讀的機率。

```
# === Predictions === #
# When making predictions, retrain the model on the whole training set
modelRF.fit(X, y)
modelXT.fit(X, y)
modelGB.fit(X, y)

### Combine here
predsRF = modelRF.predict_proba(X_test)[:, 1]
predsXT = modelXT.predict_proba(X_test)[:, 1]
predsGB = modelGB.predict_proba(X_test)[:, 1]

predsRF[predsRF>0.9999999]=0.9999999
predsXT[predsXT>0.9999999]=0.9999999
predsGB[predsGB>0.9999999]=0.9999999

predsRF[predsRF<0.0000001]=0.0000001
rredsXT[predsXT<0.0000001]=0.0000001
predsGB[predsGB<0.0000001]=0.0000001

predsRF = -np.log((1-predsRF)/predsRF)
predsXT = -np.log((1-predsXT)/predsXT)
predsGB = -np.log((1-predsGB)/predsGB)

preds = coefRF * predsRF + coefXT * predsXT + coefGB * predsGB
```

4 結論

推薦方法 / 被閱讀率	隨機給推薦文章	隨機最新文章（本週）	LSA 推薦	2 layer Logistic Regression
全部文章	52.6%	78.84%	77.86%	78.80%

推薦方法 / 被閱讀率	LSA 推薦	2 layer Logistic Regression
最新文章（本週）	80.84%	85.84%
非本週	76.69%	78.80%

由資料的分布也可知，被讀率其實是否是最新文章差異很大，因此將文章分成最新文章與非最新文章，而得到的結果：LSA 與 2 layer Logistic Regression 在最新文章中的表現較好，但有一個很重要的影響因素是因為『最新文章』被閱讀的機率原本就很高了，所以 Date 這個特徵值在 2 layer Logistic Regression Model 中影響蠻大，有可能產生 overfitting 的影響，最終用 soothing 的方式去處理，最後所以粽合以上結果，此推薦系統的機制為文章：最新文章、LSA、2 layer Logistic Regression、各推薦一篇，最新文章還可以再用 LSA、2 layer Logistic 塞選判別是否適合推薦給使用者。

5 未來展望與建議

這這次實驗中主要的目的是做一個推薦系統模型，並製作出整個流程，以下大概是提出可以再繼續研究或是改進的部分。

1. 介面：使用 Facebook 的介面，因為可能因為介面的設計而影響使用者是否閱讀，不一定是因為文章內容或是個人的喜好，因此在實驗一開始是假設使用者對於介面是不反感的，若要有更精確的數據應該在使用者介面以及使用者體驗上下一些研究
2. 資料集：資料集所收集的資料來自於訂閱這個 chatbot 的使用者，人數的基數其實不夠大，隨著時間增加，應該可以在增加資料量，並做更精確的分析以及模型的調整
3. 演算法：最今開始在研究與類神經網路相關的論文在文字上的應用，而目前在類神經網路方面最佳的應用多半是圖像，或是訊號的處理，在文字上的判斷進展相對較不足，像此篇研究當中文字處理的方式是用 word embedding 的方式，而目前有人在研究，如何把自然語言的處理轉換成圖像或是訊號，不再是傳統的用 word embedding 也得到較佳的效果，所以之後在文本的分析上可以朝這方面研究看看。