资讯科学系专题研究

题目：
Chatbot 平台建置文章推荐系统

国立政治大学

资讯科学系专题研究

姓名：苏裕胜

1 摘要

此研究为研究推荐系统的建立，以及背后演算法的优化，
使用 Nodejs 的 express 架构并连接 Facebook 的 API，在 Facebook 上建立 Chatbot，此 Chatbot 的功能是推荐给 Facebook 上不同属性的使用者不同性质的文章阅读。主要研究是针对不同的资料集以及资料型态（文本资料,类别资料），找到最佳推荐方式，结论为以下三者：
1. LSA 分析文本资料找出与使用者所阅读过文章相似度高的推荐。
2. 使用两层的机器学习演算法 (First layer: Random Forest, Extra Trees, GradientBoosting, Second layer: Logistic Regression)做预测，采用资料集为类别资料，然后将排名高的文章推荐给使用者。(这里的资料集使用，[2]所得到的文章相似度结果，以及 Facebook 所提供的 API 抓取使用者的: ID, Gender, Location, Time, readHistory 等等….作为资料集，Feature 的塞选是很重要的一环，但这我们先不谈 Feature 的塞选以及前置处理，主要着重在演算法的选择以及模型的优化。)
3. 最新发布的文章，实验显示至少有 70%以上的使用者，会点选最新发布的文章

研究结果已实作完成（如图），目前也持续优化中，专案以及相关程式可以经由 github 下载：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

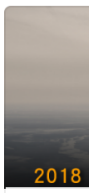
回首頁

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...

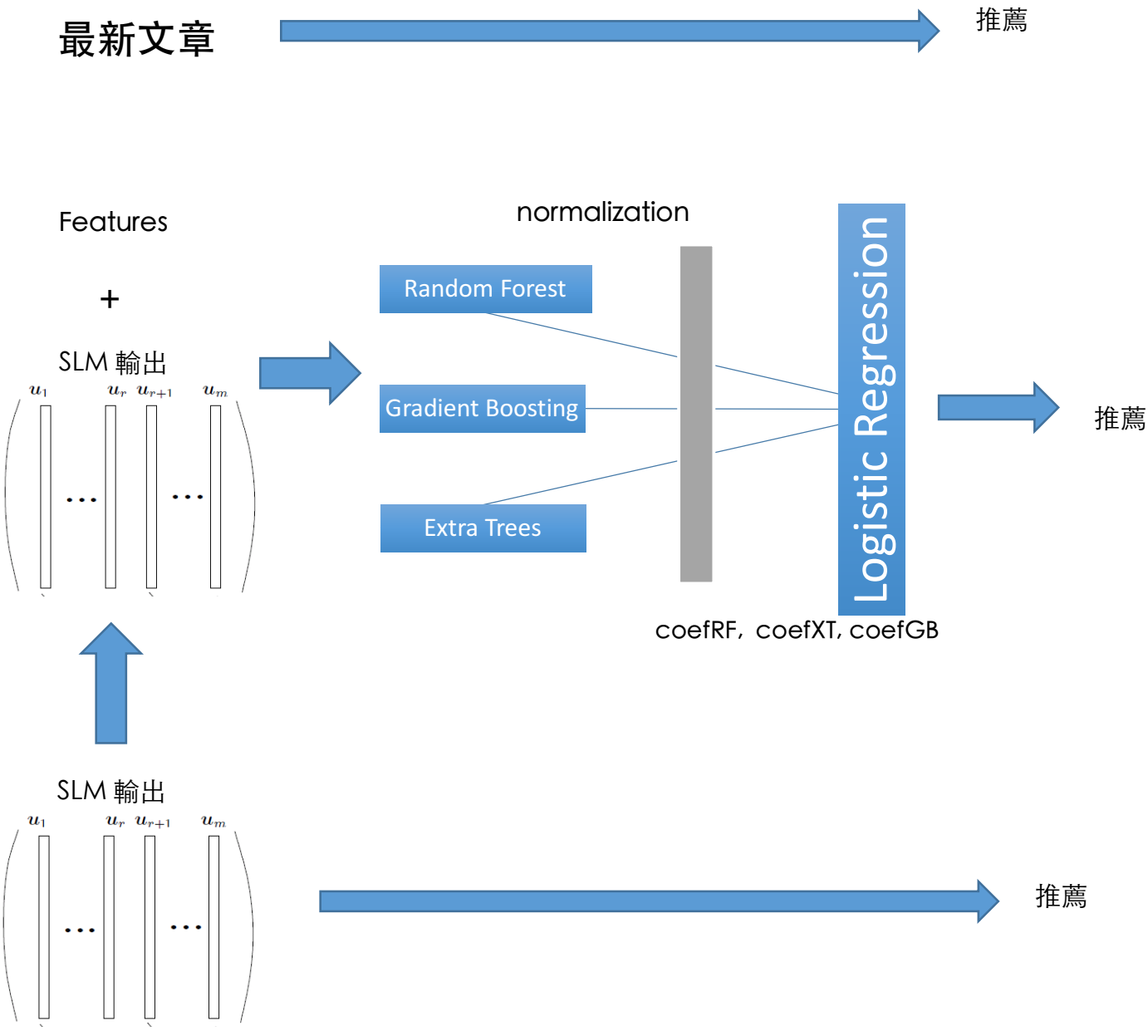
Type a message...

2 動機

因为现在市面上做推荐的系统非常多，不论是做广告推荐，商品推荐等， 此研究是希望能够建立一个好的文章推荐系统，并更了解如何训练以及优化背后的推荐演算法。

3 實驗流程與方法

實驗整個架構圖



3.1 文本資料分析：LSA

使用爬虫程式（参考 paser*.py） 从 stock feel 股感网站 https://www.stockfeel.com.tw，抓下来所有财经相关文章，当成文本训练，以及资料集使用。
使用 LSA 语意分析方式，找出与使用者所阅读过相似度高的文章（参考：LSM/lsm.py , LSM/lsm_ gensim.py）

3.1.1 Jieba 断字断词处理

- 载入繁体词典+自定义词库
  除了 Jieba 所预设的字词外，还添加了更多财经相关的字。
- 取出文章中的关键词
  import jieba.analyse 取出关键字
- 关键词去除停用字
  jieba.analyse.set_stop_words()

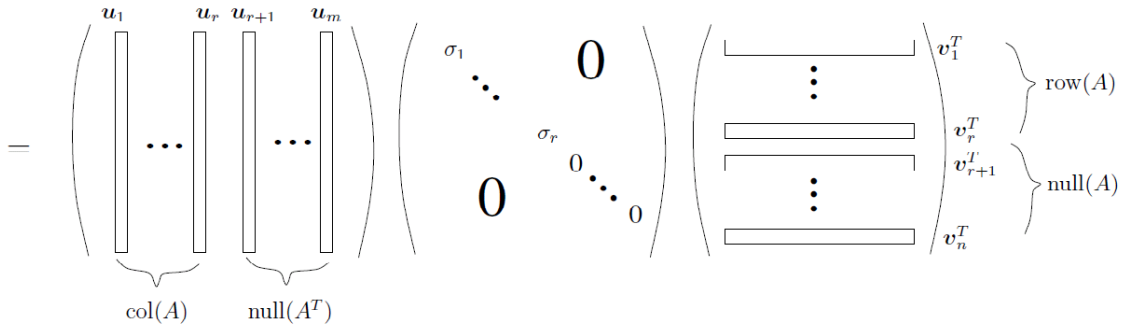
3.1.2 将每篇文章转成向量表示(doc2vec)

將处理好的字，去扫全部的文本，然后做成 doc2Vec

❖ 注：但是最后采用 gensim，因为训练的资料集够大效果较佳(参考 LSM/lsm_ gensim.py)

3.1.3 singular value decomposition 降低数据维度

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



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

使用 singular value decomposition 降低资料杂讯

3.1.4 計算 Cosin similarity

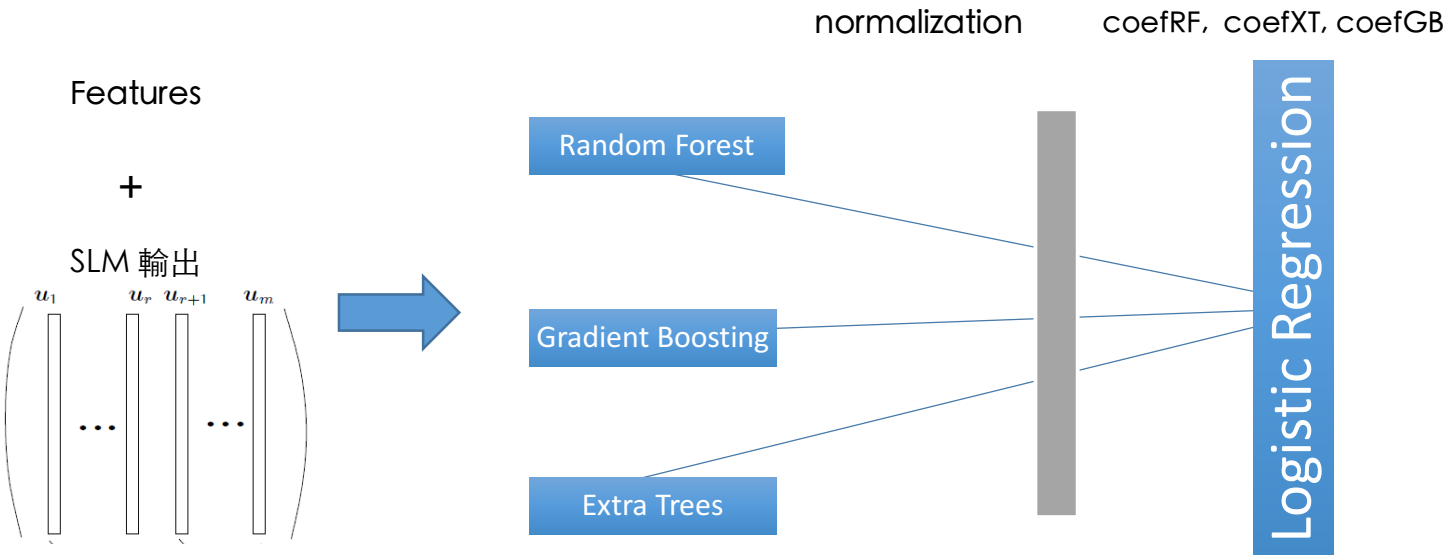
$$similarity = \cos(\theta) = \frac{A\cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

spatial.distance.cosine(low_rank_document_term_vec, vec)
最终得到的结果如这样：
[(53, 0.99999994), (19, 0.96728575), (43, 0.96589249), (33, 0.96417892), (28, 0.95892304).....]
(文章 id 编号　，　此文章与 No. id 文章的相似度)
这样很容易找出与此文章相关的文章排名
。

3.2 类别资料分析：两层机器演算法模型

3.2.1 机器演算法架构

第一层先使用各个基本的模型进行尝试，如 KNN、Decision Tree、Naïve Bayes、SVM、Random Forest、AdaBoost、Extra Trees、Gradient Boosting，选最佳的三个模型（下文会提到这三个，分别为：Random Forest, Extra Trees, Gradient Boosting），用这三个模型的产出 Logistic Regression ：F(x)= coefRF*rf(x)+ coefXT*xt(x)+ coefGB*gb(x) 的参数(coefRF, coefXT, coefGB)，然后做最终的预测。（在做预测时，特征值得处理以及选择很重要，但这边的研究主要着重在于模型的建立以及优化，所以暂且不讨论）



3.2.2 第一层模型的选择

先把目前比较常用常看到的分类模型拿进去测试，并得到以下结果。

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

从上面的数据可以得知，比较传统的模型效果相对比较差一些，如简单暴力的 KNN、naïve Bayes、或是未经改良的 Decision Tree。相对的，许多改良 Decision Tree 的方式都可以得到一些比较好的结果，如 AdaBoost 及 Gradient Boosting 透过对于分类结果权重的改变去增进 Tree 的准确度，又或者 Extra Trees 及 Random Forest，透过 Feature 的选择去剪裁出最好的 Tree。

而实际上，若将这些比较不准确的 Model 结果放入，将产生非常严重的负面影响，因此最终我仅将 Random Forest、Extra Trees 以及 Gradient Boosting，用这三个模型的产出 Logistic Regression ：F(x)= coefRF*rf(x)+ coefXT*xt(x)+ coefGB*gb(x) 的参数(coefRF, coefXT, coefGB)。

3.2.3 三個 Model 的參數調整

由于参数测试非常耗时，因此使用 sklearn 套件，透过自动化的方式进行参数测试。

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

最后把做 F(x)= coefRF*rf(x)+ coefXT*xt(x)+ coefGB*gb(x)的三数带入，得到预测的结果。推荐给使用者阅读，最好可以达到 83%被阅读的机率。

```
# === 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%

由资料的分布也可知，被读率其实是否是最新文章差异很大，因此将文章分成最新文章与非最新文章，而得到的结果：LSA 与 2 layer Logistic Regression 在最新文章中的表现较好，但有一个很重要的影响因素是因为『最新文章』被阅读的机率原本就很高了，所以 Date 这个特征值在 2 layer Logistic Regression Model 中影响蛮大，有可能产生 overfitting 的影响，最终用 soothing 的方式去处理，最后所以粽合以上结果，此推荐系统的机制为文章：最新文章、LSA、2 layer Logistic Regression、各推荐一篇，最新文章还可以再用 LSA、2 layer Logistic 塞选判别是否适合推荐给使用者。

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，因为可能因为介面的设计而影响使用者是否阅读，不一定是因为文章内容或是个人的喜好，因此在实验一开始是假设使用者对于介面是不反感的，若要有更精确的数据应该在使用者介面以及使用者体验上下一些研究
2. 资料集：资料集所收集的资料来自于订阅这个 chatbot 的使用者，人数的基数其实不够大，随着时间增加，应该可以在增加资料量，并做更精确的分析以及模型的调整
3. 演算法：最今开始在研究与类神经网路相关的论文在文字上的应用，而目前在类神经网路方面最佳的应用多半是图像，或是讯号的处理，在文字上的判断进展相对较不足，像此篇研究当中文字处理的方式是用 word embedding 的方式，而目前有人在研究，如何把自然语言的处理转换成图像或是讯号，不再是传统的用 word embedding 也得到较佳的效果，所以之后在文本的分析上可以朝这方面研究看看。