維基分

Using Semantics and Correlation to Perform Clustering and Web Traffic Forecasting

江讀晉	徐梓豪	王錦盛	周毓修	黃宇禓
R11921038	R11921044	R11921108	R11942098	R11942100

January 5, 2023

Abstract

本專題提出了幾種數據預處理方法和神經網絡（ANN，Artificial Neural Network）模型，以有效解決網絡流量預測問題。預處理部分藉由採用頁面類型的one-hot編碼表示和時間序列的計算統計來生成轉換後的特徵向量。模型的部分其本質上是一個全連接網路，旨在盡可能準確預測Kaggle測試域的每日瀏覽次數。在整體工作中，除了一開始的資料分析以及上述提案，因受到分群概念的啓發，我們另外開發了兩個分支來將資料分組，並執行與原始方法相同的步驟訓練模型。實驗結果比較了原始提案以及兩個分支的對稱性平均絕對百分比誤差（SMAPE，Symmetric Mean Absolute Percentage Error）。

1 維基頁面名稱分析

由於訓練資料龐大，且其媒介專案、存取方式和代理模式多元，為了使後續的資料前處理可以較容易著手，我們決定先分析每筆維基頁面資料的名稱。1.1首先整理並分析各個參數的統計數量和分佈狀況，1.2則會檢視缺失資料在各個媒介專案的比例。

1.1 媒介專案、存取方式和代理模式統計

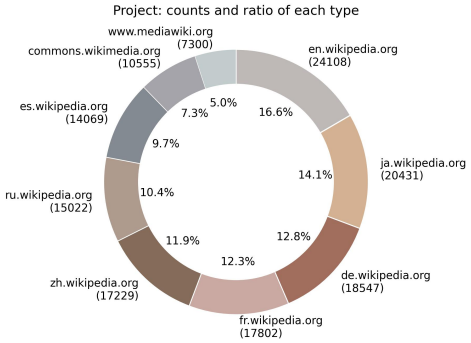
此段落首先找出媒介專案、存取方式和代理模式分別有幾種類別，並統計每種類別對應的資料個數，以及對應相關資料的平均瀏覽次數。Figure 1 (a)、(c)和(e)可看出，媒介專案的類別數較多，分布也相對存取方式和代理模式的分佈平均，這為我們後續分群提供了著手的地方。

1.2 缺失資料統計

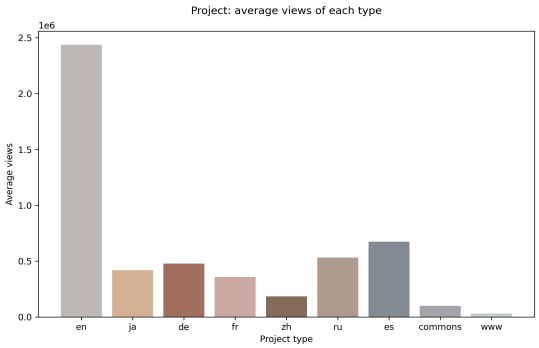
由於並非每一筆資料的瀏覽紀錄皆是完整的，因此我們亦將有效的完整資料統計出來，如Table 1所示，以方便資料前處理時參考。此外，根據資料特性的不同，有些資料可能為單一突發事件，其缺失的每日瀏覽次數甚至達總時長的一半以上。針對此議題，在後續訓練模型的資料前處理的部分，我們會闡述如何彌補缺失的每日瀏覽次數，以達到較好的訓練結果。

媒介專案	完整瀏覽次數的資料比例（%）
en.wikipedia.org	75.90
ja.wikipedia.org	90.06
de.wikipedia.org	87.23
fr.wikipedia.org	88.70
zh.wikipedia.org	76.56
ru.wikipedia.org	89.53
es.wikipedia.org	90.97
commons.wikimedia.org	51.22
www.mediawiki.org	51.59

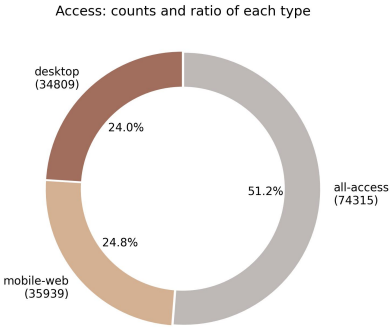
Table 1: 各個媒介專案及其對應之完整資料比例



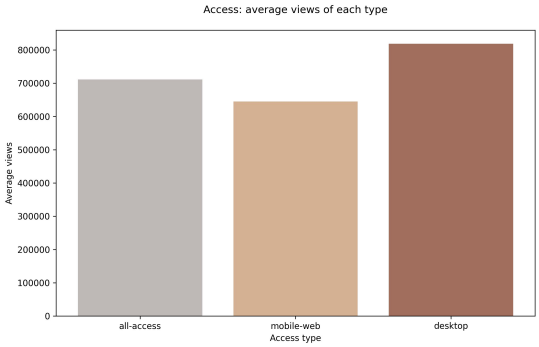
(a) 媒介專案之9組對應資料數和佔比



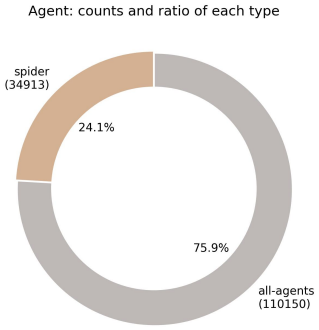
(b) 媒介專案之9組對應平均瀏覽次數



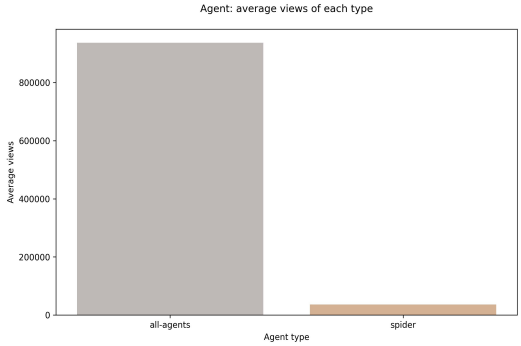
(c) 存取方式之3組對應資料數和佔比



(d) 存取方式之3組對應平均瀏覽次數



(e) 代理模式之2組對應資料數和佔比



(f) 代理模式之2組對應平均瀏覽次數

Figure 1: 媒介專案、存取方式和代理模式資料統計

2 Analysis on Popular Pages

這邊我們主要是想要看看維基百科最熱門的那些標題頁面是什麼，那他們彼此之間又有什麼樣的關聯等。

2.1 The most popular pages over all language

首先，我們先把維基百科的Dataset中的頁面所有access-type等作相加，然後去排序他們的流量。

Pages	MA7	MA10	MA30
Main Page	1.275862E+08	127633295.1	1.118742E+08
Заглавная страница	3.433818E+07	33842579.2	2.568044E+07
Special:Search	1.896414E+07	15734040.3	9.114485E+06
Wikipedia:Hauptseite	7.157759E+06	6946137.6	6.446635E+06
Organisme de placement collectif en valeurs mobilières	6.756440E+06	6726299.4	2.437741E+06
Donald Trump	5.570386E+06	4241435.1	1.728031E+06
404.php	5.129978E+06	4097900.4	2.788113E+06
Web scraping	5.077539E+06	4163644.4	2.397790E+06
Prince (musician)	4.702836E+06	3445521.8	1.246019E+06
David Bowie	4.679207E+06	3409293.8	1.245296E+06

觀察上表可以看到，"Main Page "Заглавная страница 及"Wikipedia:Hauptseite"，分別在全球前十大平均流量中佔了第一、二、四名，而這三個page即為維基百科的不同語言首頁。

其實這是個很有趣的現象，因為我們個人使用維基的方式很少會點到首頁去。例如我自己，我會從google搜尋查找關鍵字，如果維基百科有相關的解釋頁面，那我會點進去觀看。後面則是如果在那個頁面看到其他相關的頁面連結可能就一路點進去看並瞭解，但是並不會點到首頁去。所以針對這個流量的解釋，我們認為應該是其他維基百科使用者還是會習慣點回維基百科首頁去做使用。

再來就是第三名的流量網站，"Special:Search"。有了前面的結果後，這個搜尋頁面的在第三名的也不足為奇。不過這也讓我們認識到維基百科確實不同使用者有很多不同的使用方式，亦即為除了google搜尋結果連結過去的網路流量，也有不少使用者是直接使用維基百科內建的搜尋引擎下去查找資料。

2.2 Take a look back to zh-domain

再來，我們限縮觀察的範圍，回到中文page的維基百科來觀察最熱門的幾個網頁。下面是zh-domain前二十大流量的網站的流量分佈圖。

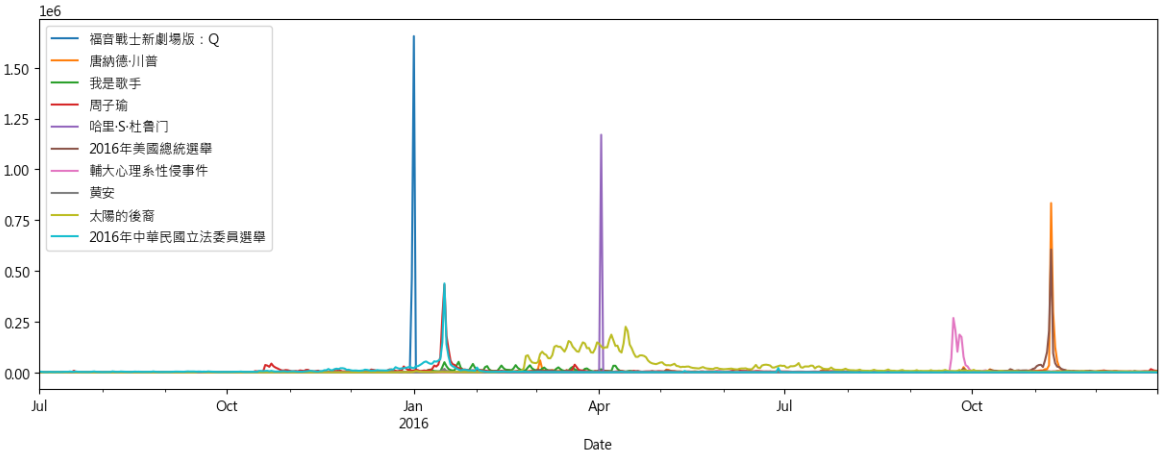


Figure 2: zh-doman 流量前十大網站

2.3 Related page

2.3.1 Pearson correlation

以下分析會使用到相關係數做為分析的metric，那我們這邊採用的是Pearson correlation。

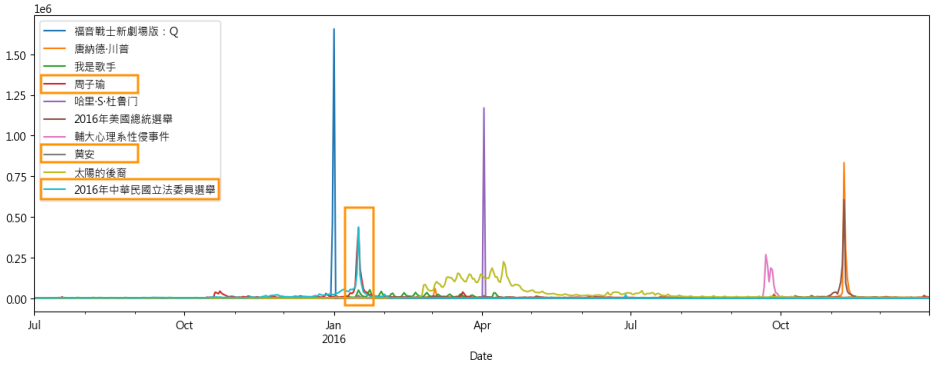
而Pearson correlation的定義如下：

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

其實與平常機率或是統計常用的相關係數計算方式相同。

2.3.2 周子瑜, 黃安, 2016中華民國總統大選

首先，我們看到有周子瑜相關的爆發性搜尋流量，他居然跟黃安還有2016中華民國總統立委大選的網路流量有一致的趨勢。(即為下圖中橘色框的部分)



深入研究，這三個事件都發生在2016/1/16那天。而周子瑜會跟黃安同時有暴增的搜尋量，就是因為黃安檢舉周子瑜台獨，導致周子瑜被迫到道歉的相關事件。

所以後面我們去找跟周子瑜頁面相關係數前十大的頁面出來研究：

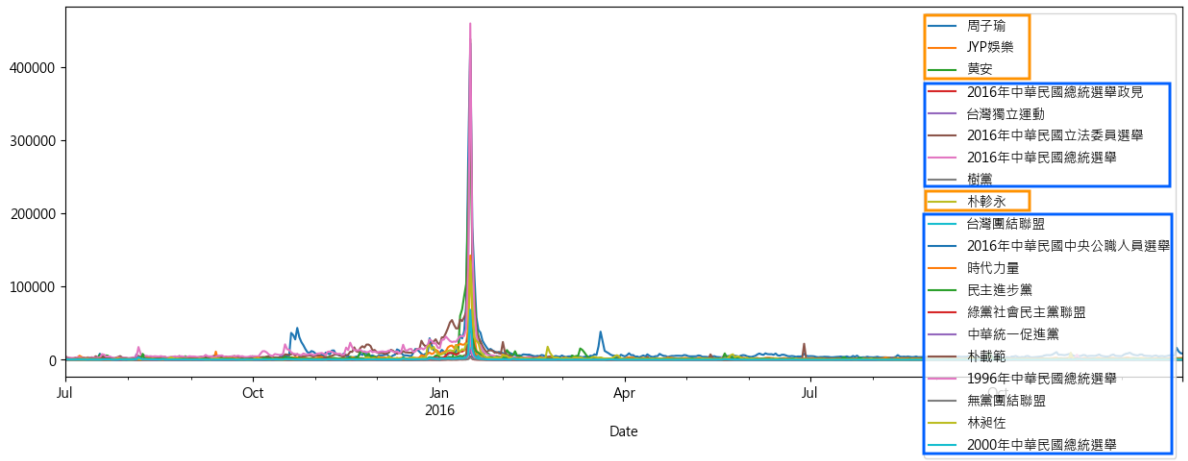


Figure 3: 周子瑜相關頁面篩選結果

可以看到在1/16那天，有周子瑜、黃安的相關事件(橘框)等，在前20相關的page有被抓出來。只不過當天剛好是16年的總統大選，所以有關於選舉的相關頁面都會被找出來，如選舉、政黨、參選人或是民主運動等。

但這裡也可以歸納出一個小小結論，如果使用time series的相關係數去做分析，雖然兩個事件獨立不太相關，但只要發生的時間點湊巧在同一天，那就會因為兩者的peak位置相同而被列為高相關性事件。即使如此，但如果用於初步分類，其實對於相關分析應該是挺有幫助的。

2.3.3 2016 美國選舉

那利用相關係數分析去針對其他有突發性流量的page，如2016美國選舉：

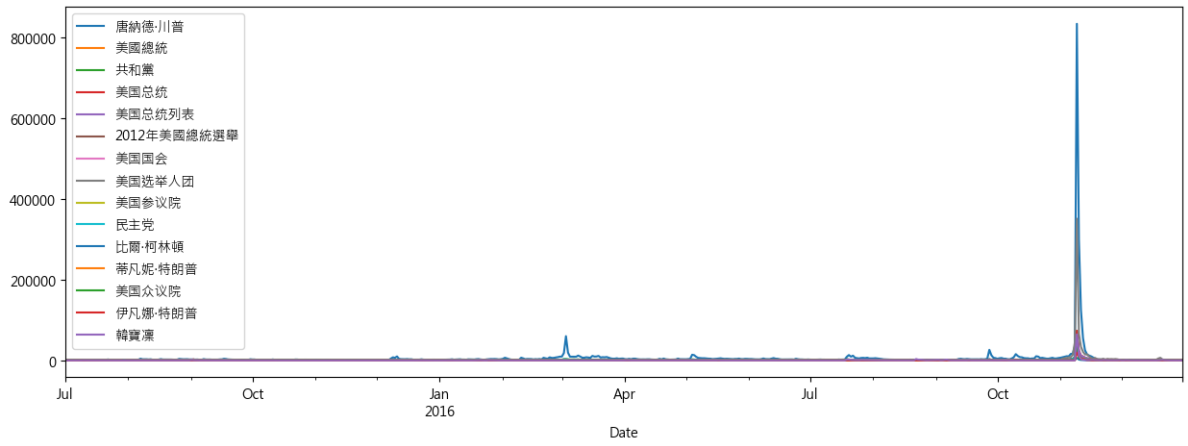


Figure 4: 2016 美國選舉相關頁面篩選結果

確實可以發現，利用相關係數分析，找出流量具相似pattern的page的確其主題都會彼此相關。美國選舉這個也是如此，附近找到的都是候選人、選舉、國會、政黨等。

2.3.4 我是歌手

但如了上述有突發性爆發流量，我們也可以看看持續性的流量網站，例如“我是歌手“。因為他是一個檔期節目，就會持續受到大家關注，進而維持一定的網頁關注度。

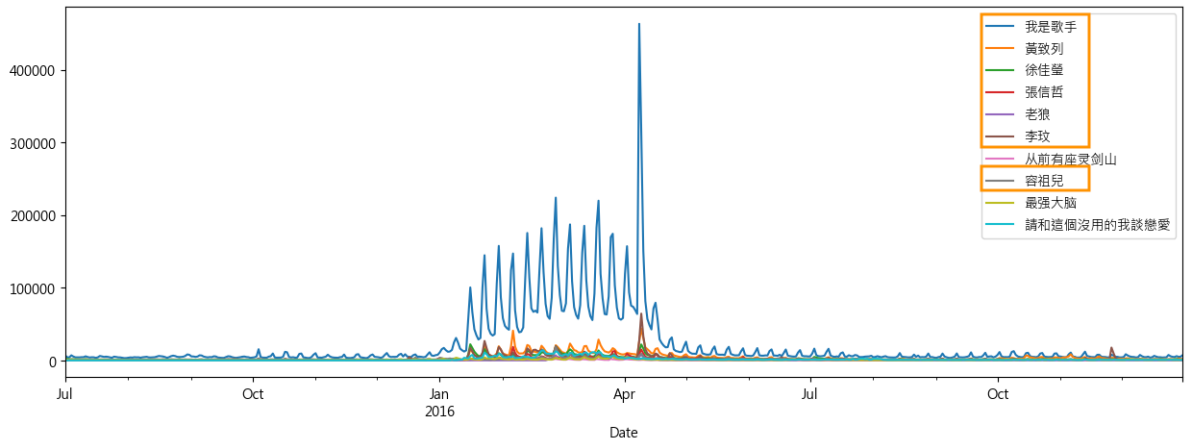


Figure 5: 我是歌手相關頁面篩選結果

橘框中的人名全都是本期節目的參賽者，所以可以證明這種持續性流量的網站也可以用相關係數去查找有關性的頁面。

3 Training Pipeline

參考自Kaggle Leader Board上CPMP[1]，我們的Training Pipeline如下:

[1] https://github.com/jfpuget/Kaggle

3.1 Data Preprocessing

3.1.1 Page

第一步預處理是對Page做，將PageTitle，Site和Access Agent提出來，以便作為之後的Input features。如下圖Fig. 6所示:

	Page	PageTitle	Site	AccessAgent
0	2NE1_zh.wikipedia.org_all-access_spider	2NE1	zh.wikipedia.org	all-access_spider
1	2PM_zh.wikipedia.org_all-access_spider	2PM	zh.wikipedia.org	all-access_spider
2	3C_zh.wikipedia.org_all-access_spider	3C	zh.wikipedia.org	all-access_spider
3	4minute_zh.wikipedia.org_all-access_spider	4minute	zh.wikipedia.org	all-access_spider
4	52_Hz_I_Love_You_zh.wikipedia.org_all-access_spider	52_Hz_I_Love_You	zh.wikipedia.org	all-access_spider

Figure 6: Page Preprocessing Result

其中Site所指的便是各個地區的不同網域，其中包含了以下九種:

- zh.wikipedia.org
- fr.wikipedia.org
- en.wikipedia.org
- commons.wikipedia.org
- ru.wikipedia.org
- www.mediawiki.org
- de.wikipedia.org
- ja.wikipedia.org
- es.wikipedia.org

而Access Agent則是用戶訪問網站的方式，有以下四種:

- all-access spider
- desktop all-agents
- mobile-web all-agents
- all-access all-agents

3.1.2 Date

第二步是對日期進行處理，可以想像Web Traffic會有以一周為週期的週期性變化，因此將日期皆轉換成第幾周第幾天的格式，如下圖Fig. 7所示:
其中w0_d0代表的便是第一周的星期一，d1便是星期二，以此類推。

3.1.3 Training Data

接下來是對training data所做的處理，首先設定一個offset=0.5，原先的visits次數會先乘上offset，再通過$\log 1p$，讓數據更符合高斯分布，例圖如下Fig. 8:

	w0_d1	w0_d2	w0_d3	w0_d4	w0_d5	w0_d6	w0_d0	w1_d1	w1_d2	w1_d3	w1_d4	w1_d5	w1_d6	w1_d0	w2_d1	w2_d2	w2_d3	w2_d4	w2_d5	w2_d6	w
0	1.5	1.0	2.5	0.5	1.5	2.5	1.0	2.5	0.0	1.5	1.5	0.5	1.0	3.5	2.5	1.0	5.5	1.5	1.5	5.5	
1	1.5	1.0	1.0	0.0	1.0	1.5	0.5	1.5	0.0	0.0	1.0	0.0	0.5	2.5	0.0	0.0	2.0	0.5	1.5	1.0	
2	1.5	1.0	1.5	0.5	1.5	2.0	1.0	2.0	0.0	1.5	1.5	0.5	1.0	3.5	2.5	1.0	5.5	1.5	1.0	5.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

Figure 7: Date Preprocessing Result

	2016-09-10	2016-09-09	2016-09-08	2016-09-07	2016-09-06	2016-09-05	2016-09-04	2016-09-03	2016-09-02	2016-09-01	2016-08-31	2016-08-30	2016-08-29	2016-08-28	2016-08-27	2016-08-26
0	2.197225	3.349904	2.302585	2.351375	2.302585	2.525729	2.351375	2.602690	2.397895	2.442347	2.602690	1.609438	2.639057	2.302585	2.140066	3.044523
1	2.014903	2.014903	2.917771	3.113515	3.135494	2.351375	3.688879	3.526361	2.251292	2.803360	2.803360	4.290460	2.484907	2.602690	1.945910	2.140066
2	2.351375	1.252763	1.791759	1.871802	1.386294	1.504077	0.693147	1.252763	1.504077	1.252763	0.693147	1.098612	1.252763	0.916291	0.693147	0.916291
3	1.704748	1.871802	2.014903	2.079442	1.504077	1.871802	2.197225	1.704748	2.079442	2.079442	1.704748	1.504077	2.351375	2.140066	2.674149	2.014903
4	1.252763	1.252763	1.252763	1.252763	0.405465	0.916291	1.098612	3.020425	0.916291	1.098612	0.693147	0.916291	0.693147	1.252763	0.916291	0.405465

Figure 8: Data Value Preprocessing Result

3.1.4 Input Data

最後是Input Data的處理，可以大致分成三種features，Statistic、Site、Access Agent。其中Site和Access Agent用one-hot encoded成dimention分別為9和4的vector，而statistics則包含了六種，並且套用在20個不同長度的區間，因此總共會有$6 \times 20 = 120$種features。這20種區間分別是: (0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (0, 2), (2, 4), (4, 6), (6, 8), (0, 4), (4, 8), (8, 12), (12, 16), (0, 8), (8, 16), (0, 12), (0, 16)，此區間以周為單位，可以看出有1周到16周的間隔，以此方式去得到長時間以及短時間的資訊。而六種統計資料如下:

- median_w1_w2: 計算此區間內的中位數。
- mean_w1_w2: 計算此區間內的平均值。
- max_w1_w2: 計算此區間內的最大值。
- median_diff_w1_w2: 計算此區間內的中位數和前一天的差距。
- median_diff7m_w1_w2: 計算此區間內的中位數和前一週的差距。
- mean_diff7m_w1_w2: 計算此區間內的平均數和前一週的差距。

3.2 Model Structure

我們所用來評估的baseline model是一層由133 mapping到63的Linear layer，因為data處理完之後，分別會有120種statistics，9種site，4種access agent，將這133種features concat成同一ector作為model input，而要output的則是63天的Visits times。

3.3 Training Steps

首先先建立20個models，每一個epoch將datasets用GroupKFold分組，每次分成20個folds，每個model從一個fold拿取所需資料進行訓練，最後會生成20個predictions，取這20個predictions的median作為最終預測結果。(GroupKfold會確保同一個group的數據不會同時出現在training set以及testing set，讓模型不會只學會同一個group的特徵。Grouping的方式是以pageTitle來分，因此會有大約14萬種group。)

3.4 Criterion

我們訓練模型所使用的loss function，採用和Kaagle評分相同的SMAPE [1]，此指標適合用於評估regression結果的好壞。SMAPE的定義如下：

$$\text{SMAPE} = \frac{1}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|F_t| + |A_t|)/2}$$

其中n為預測資料總數，F_t為預測資料值，A_t為實際資料值。

3.5 Baseline Model Result

此Baseline Model的結果如下Fig. 9，大約是排在Kaggle Leaderboard第20名的位置:

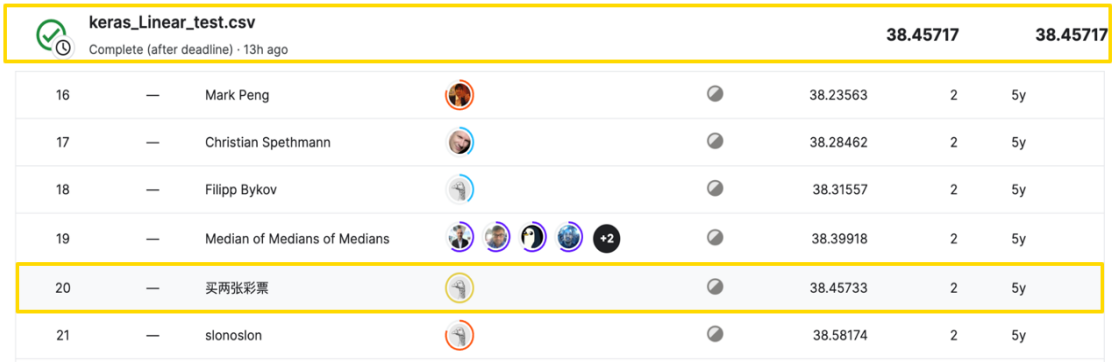


Figure 9: Baseline Model Result

接下來的部分皆是以此Training Pipeline去進行。

4 分群

前面一開始我們算是train出一個universal的模型來預測未來流量。那後面這個段落，我們想來探討分群各自train出一個model，我們猜測可能因同個群內有某些特性相同，所以train出來的model會比較有針對性，可能會有比較準的預測。

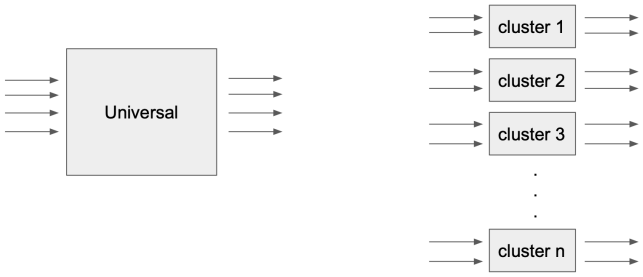


Figure 10: 使用分群示意圖

4.1 相關係數分群

4.1.1 分群結果

這裡我們使用threshold的方式下去分群，我們有設立6個threshold: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6。不同的臨界點的分群數量結果如下：

Correlation Threshold	Number of clusters
0.1	528
0.2	1126
0.3	3113
0.4	6030
0.5	9681
0.6	14262

4.1.2 訓練結果及Kaggle分數

下面是我們將四個threshold的丟上kaggle評分的結果，使用SMAPE loss（越小越佳）的方式下去計算。

Correlation Threshold	Kaggle SMAPE Loss
No use clustering	38.4572
0.1	39.3414
0.2	40.2171
0.3	41.0415
0.4	41.9702

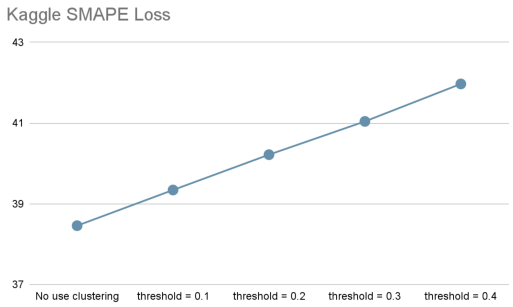


Figure 11: Kaggle SMAPE Loss

可以看到上面的分群結果，隨著threshold提高，分出來的cluster數量也隨之提高，但同時Kaggle的SMAPE Loss也是接近線性提升。

Correlation Threshold	Ratio of single member cluster
0.1	61%
0.2	57%
0.3	70%
0.4	76%
0.5	78%
0.6	81%

我們猜測這是由於threshold提高，導致存在很多單項的分群結果。而從上表可以看到，最少也有一半以上的cluster都是只有單個time series，所以各自的model在訓練時，會出現很嚴重的overfitting現象。那這樣就會導致Kaggle上的預測表現不理想。

4.2 語義分群

4.2.1 Sentence BERT

Sentence BERT（SBERT）[5]為BERT（Bidirectional Encoder Representations from Transformers）[2]的後續研究作品，其設計主要用於解決語句等級的分類任務。SBERT使用連體網路架構，也就是將輸入資料同時賦予兩個BERT模型，並將其各自模型產出最終的隱含狀態供予分類層。如此一來，SBERT模型能夠捕捉兩句子之間的相似性，這對於釋義檢測和語義文本相似性等任務有極大的幫助。SBERT已被證明在各種語句等級之分類任務中是有效的，並已用於許多自然語言處理等應用領域。

在此專題中，我們採用一開源的Python框架——SentenceTransformers——旗下預訓練的多語言模型[4]，將每筆維基百科頁面結語中的前數個句子嵌入至數值向量。

4.2.2 K-means分群

K-means 是一種用於聚類分群的無監督式機器學習演算法。它將資料集劃分為指定數量的集群，其中每個集群由其質心表示。該演算法的目標是最小化集群中的點與質心之間的距離。Algorithm 1是一個關於K-means分群演算法的簡明敘述。

基於K-means分群的概念，我們利用Lloyd演算法[3]將嵌入的向量分成幾組，其中演算法的實作使用開源的Python模組——Scikit-learn所提供的函數。

Algorithm 1 K-Means

procedure K-MEANS(X, k)
    隨機初始化k 個質心
    repeat
        將每個點$x_i \in X$ 分配給距離最近質心所在的族群
        計算新的質心作為集群中所有點的平均值
    until 質心座標點不再變化
end procedure

4.2.3 Agglomerative Clustering

Agglomerative Clustering是一種自底向上的聚類算法，在這種算法中，每個點最初都被認為是自己的群集。在每一步中，合併兩個最相似的群集，直到所有點都屬於一個單一群集。兩個群集之間的相似性可以通過各種方式衡量，例如它們的質心之間的距離、兩個群集中最近點之間的距離，或兩個群集中最遠點之間的距離。相似性度量的選擇可能會影響最終群集的形狀。Algorithm 2是一個關於Agglomerative Clustering分群演算法的簡明敘述。

類似於K-means，Agglomerative Clustering是此專題第二種語意分群的嘗試。

Algorithm 2 Agglomerative Clustering

procedure AGGLOMERATIVE-CLUSTERING(X, k)
    初始化n 個群，第i個群為X_i
    repeat
        計算所有pairs群的相似度
        合併兩個最相近的群
        更新被合併的群和其他群的相似度
    until 剩下k 群
end procedure

4.2.4 分群結果

Method	Number of clusters	Kaggle SMAPE Loss
Baseline	1	37.45717
K-means Based	211	38.89901
Agglomerative Based	17	**38.42902**

5 Conclusion

對於頁面分群，讓每一群的資料更加一致性有可能可以增加預測的表現，例如語意分群就有一個演算法可以贏過原始的模型。但是我們發現如果所分的群數量太多，反而會因為各個模型的訓練資料量不足，導致有過適(overfitting)的現象發生。

因此未來如果有機會，應該要注意不要分太多群，也就是確保每一個模型訓練資料的充足性。

References

[1] Jon Scott Armstrong. *Principles of forecasting: a handbook for researchers and practitioners*. Vol. 30. Springer, 2001.

[2] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[3] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[4] Nils Reimers and Iryna Gurevych. "Making monolingual sentence embeddings multilingual using knowledge distillation". In: *arXiv preprint arXiv:2004.09813* (2020).

[5] Nils Reimers and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084* (2019).