

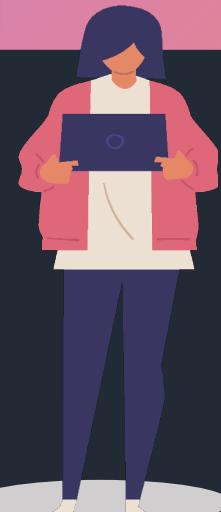
KKCompany Music Challenge: Next-5 Songcraft

Vivaldi-My_cat_can_turn_somersaults

陳繹帆、陳品絜、沈欣柏、徐韶汶

政治大學

蔡銘峰教授 - 網路搜索與探勘



團隊介紹



陳繹帆

N-gram & JMLM Solutions

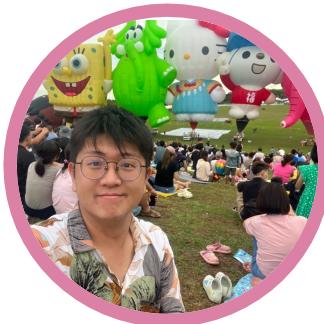
- 政大資料所碩一
- HITCON CTF 2023
世界第八
- 神盾杯初賽 第二
- 神盾杯決賽 第四
- 很會講笑話



徐韶汶

Exploratory Data Analysis

- 政大風管碩二



沈欣柏

Deep Learning Solutions

- 政大中文所碩士
- 政大資料所碩二



陳品絜

Sparse Retrieval Solutions

- 政大資料所碩一

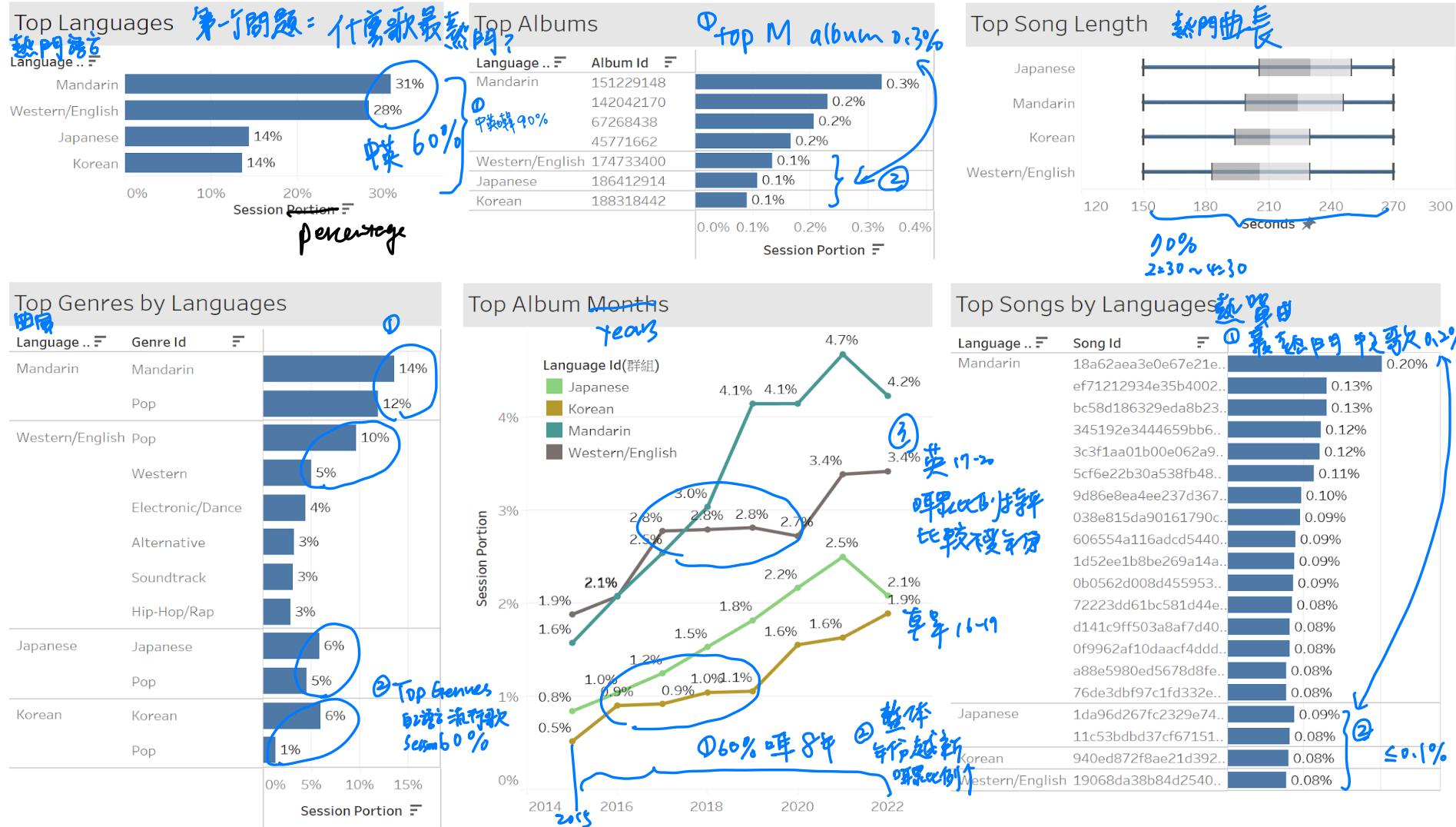
目錄

探索性數據分析

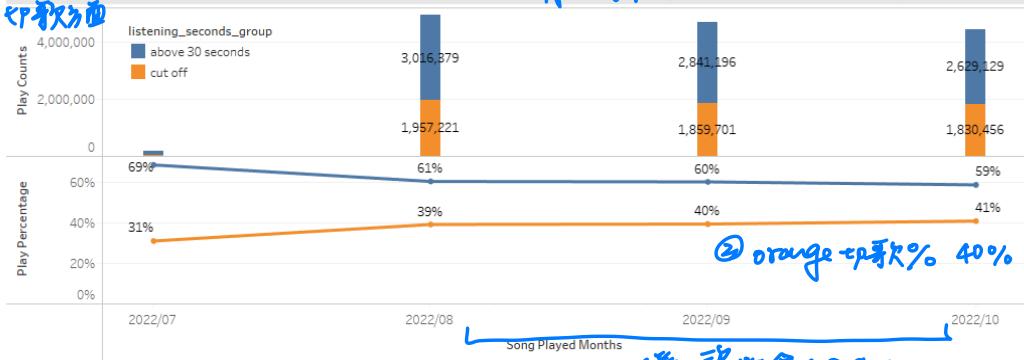
N-gram

N-gram & 歌曲相似度模型

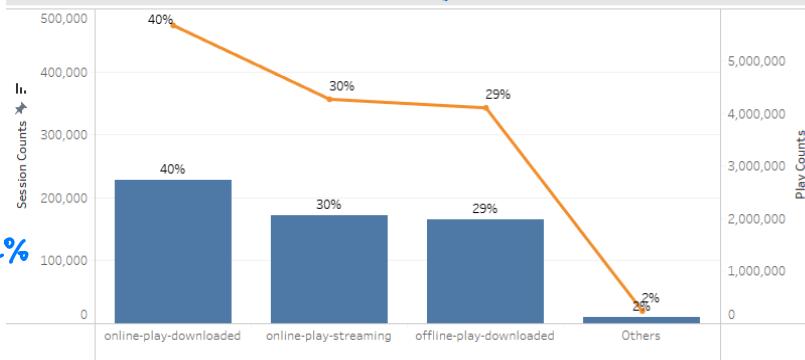
總結



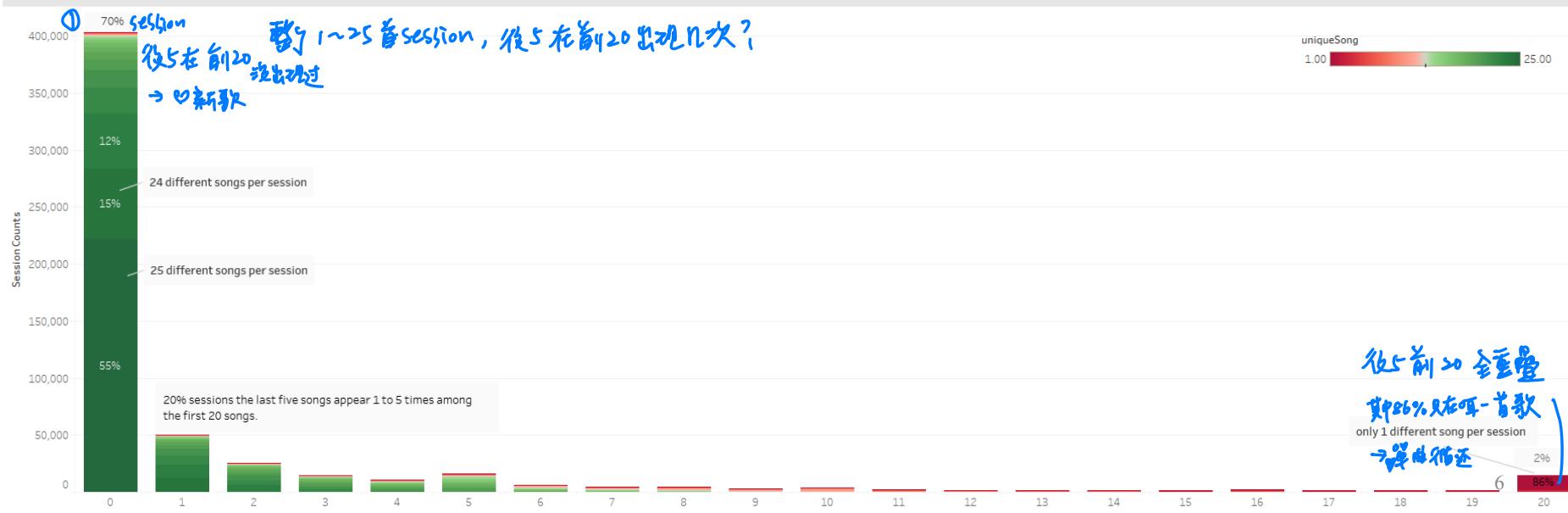
第一問題：歌曲時長



第二問題：播放習慣



Repetition Frequency of Final 5 Songs Within the First 20 Songs



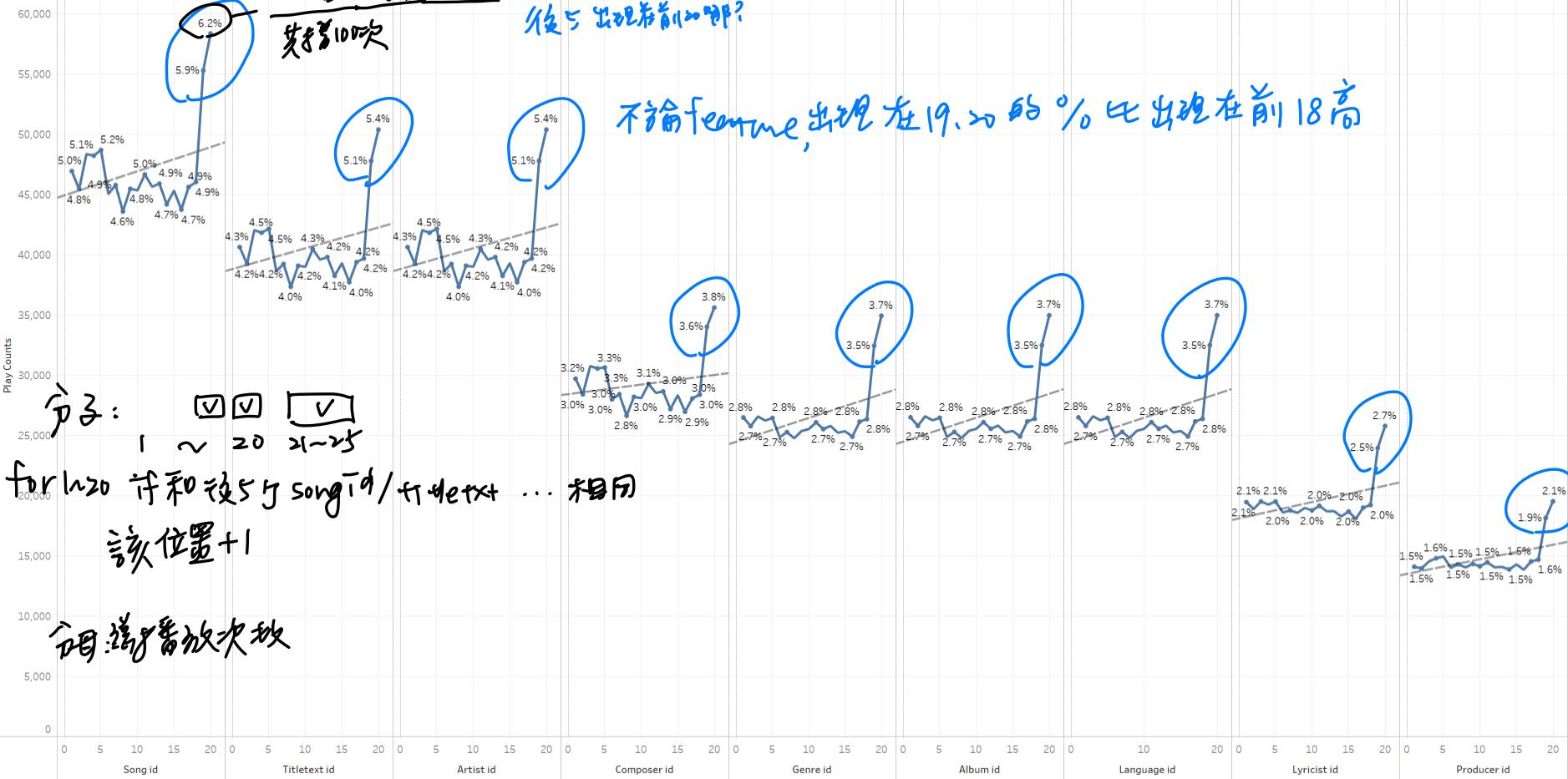
Recurrence Position of Final 5 Songs Within the First 20 Songs

前20首歌和後5首歌最後

共出現10次

後5首歌在前20嗎？

不論feature, 出現在19, 20的%比出現在前18高



分子：
1 ~ 20 21~25

for h20 和後5首歌 songid / titletext ... 相同

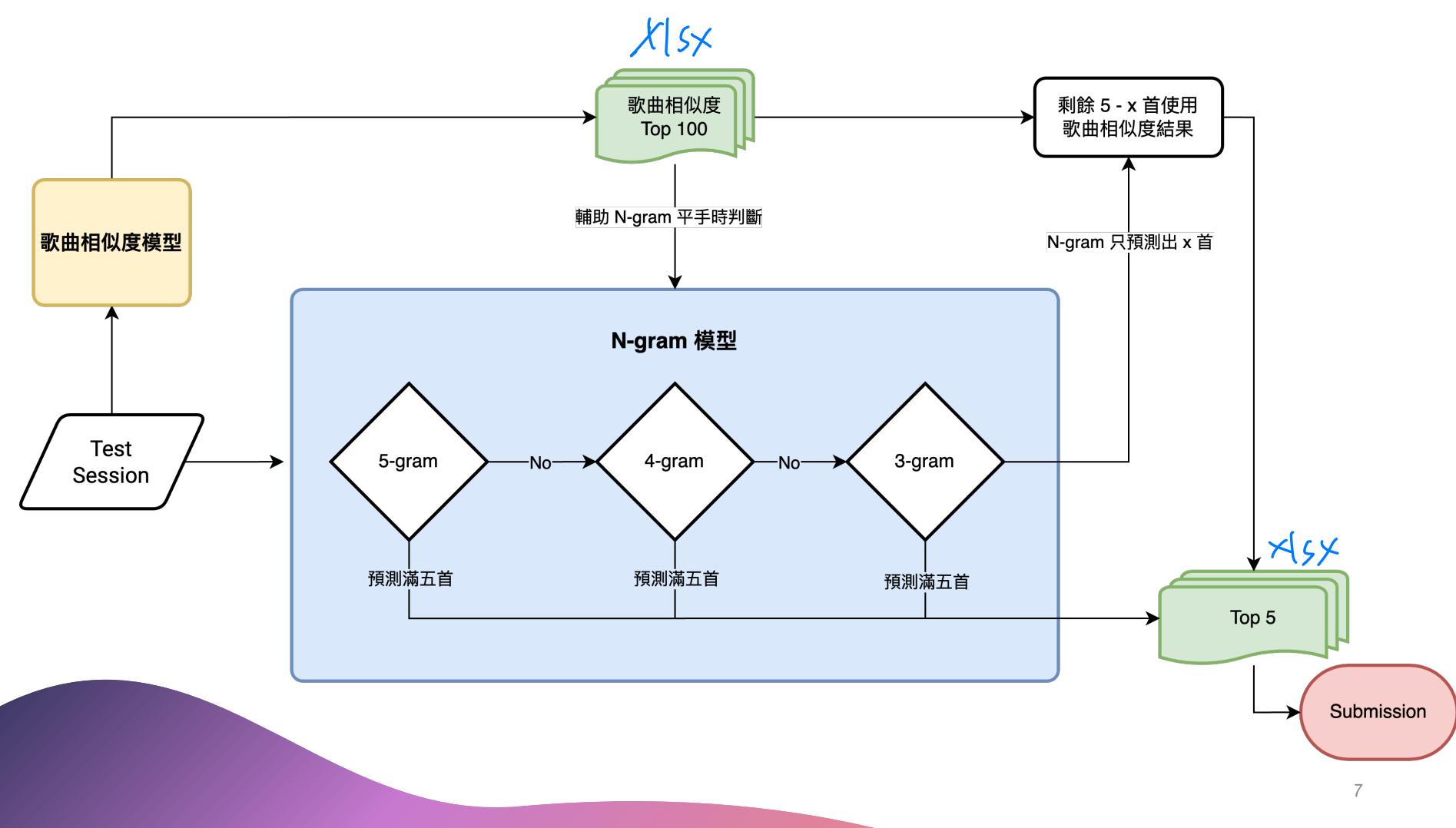
該位置 + 1

何謂播放次數

什麼樣的模型可以最好的預測後續的歌？

提出假設：

「如果我們的歌曲聆聽樣本夠多，是不是可以透過 user 聽的前幾首歌順序去模糊推論 user 後續歌曲，把難解的機器學習問題轉換成簡單的統計問題。」



N-gram 模型

如何一次性預測後面五首歌？

我們的做法是把原本預測單一首歌的 N-gram 改成預測後五首歌，也在訓練的時候把資料放進去。

```
class myConditionalFreqDist:
    def __init__(self):
        self._data = {}
        self.values = [1, 0.63, 0.5, 0.43, 0.38] # ndcg 的加權分數

    def __getitem__(self, key):
        return self._data.get(key, {})

    def __setitem__(self, key, value):
        self._data[key] = value

    def inc(self, condition, sample, index):
        if condition in self._data:
            if sample in self._data[condition]:
                self._data[condition][sample] += self.values[index]
            else:
                self._data[condition][sample] = self.values[index]
        else:
            self._data[condition] = {sample: self.values[index]}

    def most_common(self, condition, n=None):
        if condition in self._data:
            counter = Counter(self._data[condition])
            return counter.most_common(n)
        else:
            return []
```

我們的 N-gram 模型 : 4-gram 範例

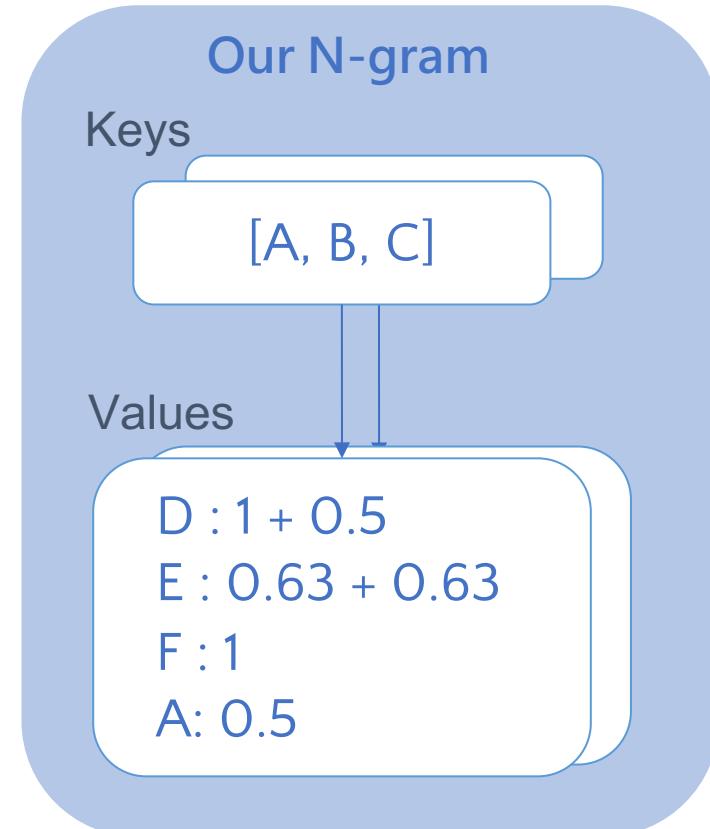
Data:

| | | | | | | |
|---|---|---|---|------|-----|-----|
| A | B | C | D | E | A | ... |
| | | | 1 | 0.63 | 0.5 | |

| | | | | | | |
|---|---|---|---|------|-----|-----|
| A | B | C | F | E | D | ... |
| | | | 1 | 0.63 | 0.5 | |

預測 A, B, C 後面接的歌:

| | | | | | |
|---|---|---|---|---|---|
| A | B | C | ? | ? | ? |
|---|---|---|---|---|---|



我們的 N-gram 模型 : 4-gram 範例

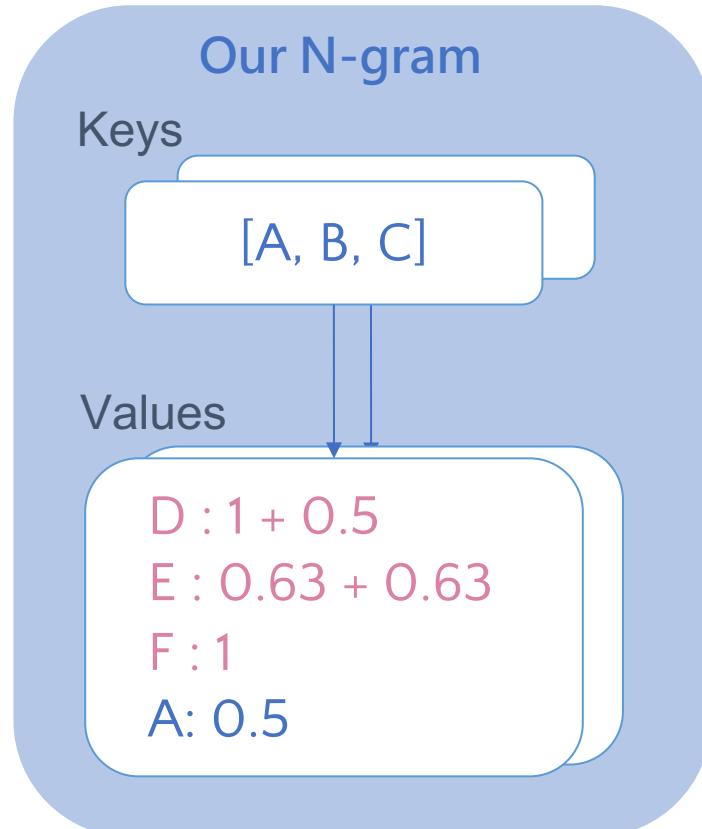
Data:

| | | | | | | |
|---|---|---|---|------|-----|-----|
| A | B | C | D | E | A | ... |
| | | | 1 | 0.63 | 0.5 | |

| | | | | | | |
|---|---|---|---|------|-----|-----|
| A | B | C | F | E | D | ... |
| | | | 1 | 0.63 | 0.5 | |

預測 A, B, C 後面接的歌:

| | | | | | |
|---|---|---|---|---|---|
| A | B | C | D | E | F |
|---|---|---|---|---|---|



這樣設計有什麼好處

- 可以捕捉模糊的歌曲相對關係
- 例如出現在相同順序後面第二首歌兩次的歌曲 E，它在模型中的分數會比只出現在第一首歌一次的歌曲 F 的分數高
 - Score E = $0.63 * 2 = 1.26$
 - Score F = $1 * 1 = 1$

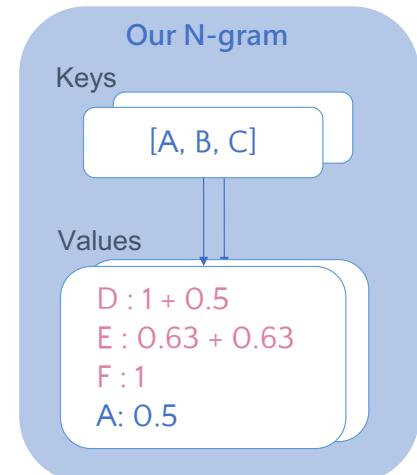
Data:

| | | | | | | |
|---|---|---|---|------|-----|-----|
| A | B | C | D | E | A | ... |
| | | | 1 | 0.63 | 0.5 | |

| | | | | | | |
|---|---|---|---|------|-----|-----|
| A | B | C | F | E | D | ... |
| | | | 1 | 0.63 | 0.5 | |

預測 A, B, C 後面接的歌:

| | | | | | |
|---|---|---|---|---|---|
| A | B | C | D | E | F |
|---|---|---|---|---|---|



N-gram 成果

| Method | Kaggle Score |
|---------------------------------------|--------------|
| Only 3 gram | 0.46931 |
| 3 gram (找不到的歌曲，就放 session 內倒數幾首) | 0.52459 |
| 3 + 4 gram | 0.55137 |
| 3 ~ 5 gram | 0.55307 |

歌曲相似度模型：把歌當成文件搜尋

我們把歌當成文件：

- 文件內容使用歌的 meta-data 以及這首歌出現在哪些 session
- 最佳成果使用的 meta-data: artist, album, language, genre

```
artist_id_297472  
album_id_4091453  
language_id_62  
genre_id_abdc0a50aff67c591737bc6f57a36e09  
session_1 session_137043 session_324391 ...
```

(示意) 某一首歌對應的文件內容

歌曲相似度模型：Language Model 算相似度

- 使用 **Language model with Jelinek-Mercer smoothing** 計算歌曲相似度並搜尋。
- 較大的 λ 可以提升對冷門歌曲的估計準確性。
- 我們提出假設：熱門的歌曲都會被 N-gram 挑走，那我們是不是專門做冷門歌曲會比較好，於是用 Jelinek-Mercer 語言模型相似度來進行剩下歌曲的推薦。

$$P(w|D) = (1 - \lambda) \frac{tf_{w,D}}{|D|} + \lambda \frac{cf_w}{|C|}$$

▲ Language model with Jelinek-Mercer smoothing

歌曲相似度模型：輔助 N-gram

- 我們能拿這個模型的結果來做什麼？
- 發現 **N-gram 結果可能出現同分**的情況，因此若同分就去看他們在歌曲相似度模型當中的分數，就可以達到更好的排序效果。
- 賽中我們的方案可以達到 0.56580 。
- 賽後分數：0.56580->0.56691

模型的優勢

1. 沒有不可解釋的東西，沒有隨機性，不怕過擬合。
2. 如果用戶不喜歡推薦的歌曲，可以在模型中扣分做抑制。
3. 運算快速，N-gram 可以實時做運算，相似度模型每天更新即可。
4. 容易在模型中加入 trick，例如可以簡易把很久以前的歌曲聆聽紀錄從模型中清除掉。

| | 訓練時間 | 預測時間 | 分數 |
|----------------|-------|-------|------|
| N-gram | 5 分鐘 | 2 分鐘 | 0.55 |
| N-gram + 歌曲相似度 | 10 分鐘 | 50 分鐘 | 0.56 |

過程中遇到的問題和解決方案

(藍字為口頭分享重點，黑字以文字分享優先)

1. N-gram 本來是一次預測一個 token 的，但是會偏掉，為了不讓它偏掉只能把後面五個放進去。
2. 但是 nltk 套件不支援不同權重的計算方式，所以乾脆刻一個。
3. 每次都要重算一部分東西太麻煩，後來乾脆把運算拆成幾個環節，算好就存起來就好，也是我們可以不斷進行提交的關鍵。
4. 雖然 BM25 和 JMLM 的分數表現差不多，但實際上 BM25 在熱門歌曲上表現比 JMLM 好，冷門歌曲上 JMLM 較佳。

過程中的失敗嘗試

(藍字為口頭分享重點，黑字以文字分享優先)

1. N-gram 裡面放 set，只要前面是特定歌曲，不管順序就可以預測，不過發現順序性很重要。
2. N-gram 在放資料時做遮罩，只要歌有相對順序關係就好，但分數糟糕。
3. 透過 2. 結果猜測機器學習學不到，是因為需要絕對順序關係。
4. GRU 從 Cross-Entropy 訓練一百輪約 0.27。
5. 更大的自然語言模型更學不到東西。
6. Collaborative filtering 可能是因為矩陣太大，所以效果不是很好，並且難以加入歌曲資訊，頂多做到 0.27。

後續可以做的研究

1. 改善語言模型
2. N-gram 改為將歌曲按照時間進行推理，例如每天遍歷一次模型將所有數值 $\times 0.98$ ，讓越久以前聽過的歌曲越不重要。
3. 嘗試加入 learning to rank
4. 增加抑制項，更好地控制模型，例如用戶跳歌就扣分，避免模型大量推特定歌曲。

最終方案

- N-gram + 歌曲相似度模型最終可以達到 0.56691
- 具體程式碼放在 <https://github.com/afan0918/KKCompany-Music-Challenge-Next-5-Songcraft>
- 歡迎 star 和 follow 窩 > <

