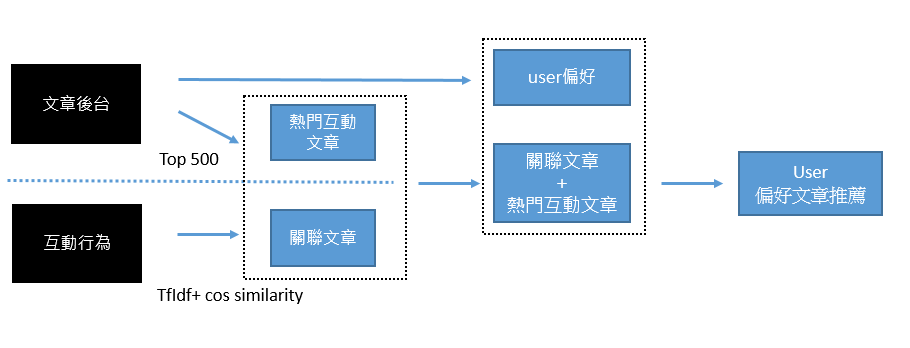
**\*根據使用者互動找出使用者最喜愛之文章，再依最喜愛文章推薦關聯文章50篇。**

Remark: 文章順序關聯(高->低)->熱門互動(高->低)

**推薦架構示意**



**API(可參考request\_demo.py)**

啟動server(\_\_init\_\_.py)

每日9:00-9:02可發一request排程進行更新(約1min 50sec)與cache

request

* URL: <http://127.0.0.1:8010/recommend_api/pid_search>
* method: post

parameter

* + pid : string
  + article\_num : int
  + example: {“pid”:”abc1234”,” article\_num: 30}

request example

import requests

post\_pid = {"pid":"002a058d-40d4-4883-9556-4015a516df88","article\_num":15}

output = requests.post(f"http://127.0.0.1:8010/recommend\_api/pid\_search",json = post\_pid)

**函數步驟(question\_recommend.py)**

**def cleaning\_content**

*#question貼標*

step1: 使用每篇question之content進行斷詞

step2: 保留前10000之詞彙

step3: 保留前10000之詞彙

step4: 將文章與標籤進行tf-idf算出每一文章

step5: 進行cos similarity,比對文本與文本間相似標籤程度

step6: 找出每一篇文章排名前50相關文章

**def Top\_Ranking**

*#算出behaviors熱門互動文章*

step1: 算出每一篇question之互動數

step2: 標準化每一互動數

step3: 加權每一互動數score = view\*0.05 + clap\*0.1 + attention\*0.3 + share\*0.3 + thank\*0.2

step4: 比對互動文章是否存在後台存在,只保留存在之文章

step4: 篩選前500互動數score之文章

**recommend\_article**

*#將每一文章推薦補滿50篇文章*

step1: 將關聯文章未推滿50篇貼上熱門互動文章

step2: 去除重複之文章(保留first)

**user\_prefer**

*#算出每一pid對各文章互動行為分數,保留pid score最高之文章代表pid最喜歡之類型,若分數相同則保留最近一次互動文章*

step1: 算出pid與文章互動matrix

step2: 加權pid對每一文章互動行為加權score = view\*0.05 + clap\*0.1 + attention\*0.3 + share\*0.3 + thank\*0.2

step3: 篩選每一pid分數最高之文章

step4: 排序pid, score, event\_date 篩選pid之下若score相同則保留最近一次互動文章

**user\_recomment**

*#建立推薦名單*

step1: 進行pid喜好文章與關聯文章進行合併

step2: 補上後台未存在之文章熱門互動前50