Final Project

Group 7

0556629 唐千琳
0656007 黃千芸
0656027 陳懌安
0656109 黃慎航

Warning

- 因為不是真的用來報告的PPT，而比較像是Final Report，以下有些頁面的字會很多，雖然看起來很不舒服(至少我自己覺得看到PPT字很多很不舒服啦)，也請慢慢看完，感謝您的閱讀！

Outline

- Key Point of Our Model
- Dataset
- Task1 - Title Generation
  - Proposal Version
  - Final Version
  - Difference Between Proposal and Final Version
  - Other Description
- Task2 - Title Evaluation
  - Proposal Slide
  - Final Version
  - Feature 1
  - Feature 2
  - Feature 3
- Demo Results
  - Task 1
  - Task 2
- Other Suggestion

Key Point of Our Model

- Task 1
  - 我們的Model最大的特點是我們有做分類，每一類都有自己產生Title的特殊規則，再結合Task2從一堆Candidates中取分數高的Title輸出。
- Task 2
  - 我們去分析蒐集而來的1000多部熱門電影名稱，分析這些電影常用的命名方式，如：詞性組合、頻繁用詞，當作一部電影名稱是否吸引人的依據。並且分別將這些features量化，最後加總起來，以此分數作為電影標題的排名依據。

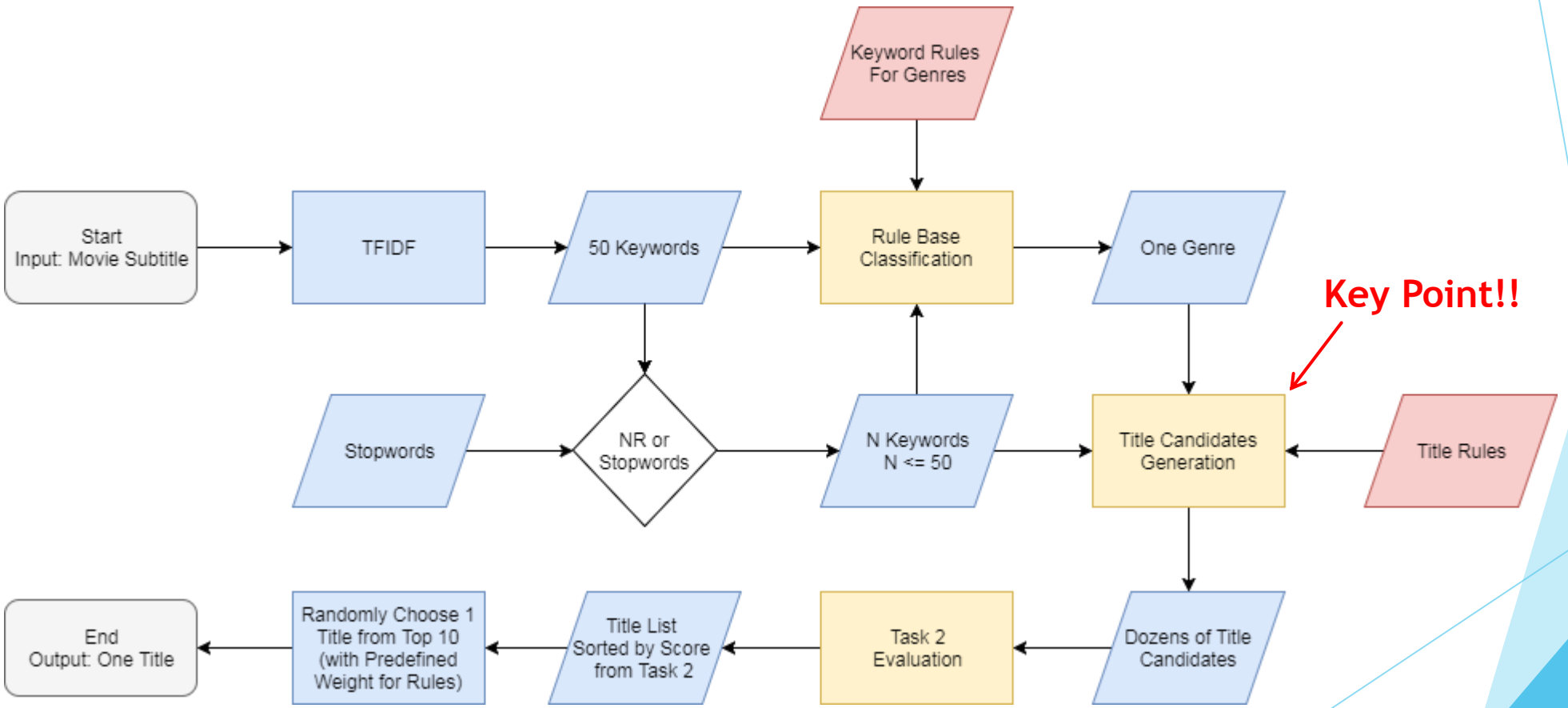
Dataset

- 從IMDB依照類型(action, romance, fantasy…)
- 每個類型抓Top Rated的100個電影英文名稱
- 到Opensubtitles下載字幕
- 英文片名用豆瓣電影API搜尋中文片名

Title Generation
Proposal Version

- Training
  - Script
    - TFIDF -> 100 words
    - Movie genre classification (use word embedding feature)
  - Title
    - POS analysis for each movie genre (e.g. NN, VN…)
- Testing
  - Classification -> find movie genre
  - Title Generation
    - 100 words from TFIDF
    - POS form (e.g. NN, VN…)
    - Special rules

Title Generation
Final Version



Title Generation

Difference Between Proposal and Final Version

- Genre Classification (Proposal Version)
  - 一開始想要用分類器做分類，Feature有以下兩種取法
    - 每部電影都取100個TFIDF最高的詞，用Embedding Model轉成250維的Vector再做相加，這樣每部電影都會有一個250維的Feature Vector
    - 和Hands-on1一樣，先從IDF大的字中挑250個詞，每部電影都去算這250個詞的TF(詞頻)，這樣每部電影也都會有一個250維的Feature Vector
  - 然而不管哪一種效果都不佳，原因可能有以下三個
    - 我們取得的Label是Multi-label而我們沒有經過篩選就做Single-label的分類(因為收集資料和做分類是不同人的工作，一開始沒有注意到有這樣的狀況)
    - 扣除重複部分，我們實際下載的電影只有四百多部，Training Data太少
    - 我們所取的Feature或分類方法真的無法達到我們想要的效果

Title Generation

Difference Between Proposal and Final Version (cont.)

- Genre Classification (Final Version)
  - 我們認為上頁Proposal Version分類做得不好的主要原因是第一點，也就是Multi-label的問題，本來想人工篩選成Single-label卻發現很不好篩，很多電影都很難只界定成一類，但若要做Multi-label的分類又怕資料不夠，而且觀察Single-label分類出來的Confusion Matrix之後，感覺問題好像也不是完全來自這裡，總之它就是毫無章法的分類，跟用猜的差不多。
  - 綜合上述原因，我們決定捨棄酷炫的Machine Learning方法，回歸Rule-base的方法，先人工定義每一個Genre的Keyword，比如說科幻片(sci-fi)就有「星艦」、「太空」等等字眼，若這些字有出現在某部電影前N個TFIDF大的詞中，那就判定為科幻片。
  - 這種做法的好處是Precision會很高，一旦被判定為科幻片，我們可以很有把握的在Title中使用「星際」這樣的字眼，不太會有問題。不過這是在我們的Training Data上的表現情況，實際demo時大部分好像都錯了XD
  - 相反的Recall就不高，找不出分類的電影就統稱為劇情片(drama)，會套一些很普通的Title Rule，例如直接用一個TFIDF大的詞當Title，而不加任何像「星際」、「玩命」這樣的詞 。實際demo時有一半以上的電影都被我們判定為劇情片(drama) 。

Title Generation
Other Description

- 我們最終嘗試的結果決定取每部電影前50個TFIDF大的詞做為該電影的關鍵詞(最一開始是100個)，並會再從其中篩去所有的NR(名字)，成為最終該電影的關鍵詞。
- 我們的Model好壞最主要取決於我們所訂的Title Rule
- 產生幾十個Title Candidates後經過Task 2算分數，取分數前十高的Candidates來做weighted random取一個，名次越前面weight就越高。
- 實際demo時，因為有一個多小時的時間可以跑測資，詢問助教後並沒有禁止random，也沒有規定我們只能跑一次，只是只能上傳一次，因此我們就是花了一個小時一直random，直到大部分的電影都random出我們喜歡的Title為止，不可否認有人為的主觀成分在裡面，但也沒有違反助教的規定就是了。

Title Evaluation
Proposal Slide

- Base: 1000 movies from IMDB Top Rated Movies
- 4 features
  - POS tag: common form of top 1000 movies
  - Movie Title frequency (top 1000 movies)
  - Word2vec: similarity to top 1000 movies
  - Title length: 3, 4, 5 words are most popular
- Score = $w_1f_1 + w_2f_2 + w_3f_3 + w_4f_4$

Title Evaluation
Final Version

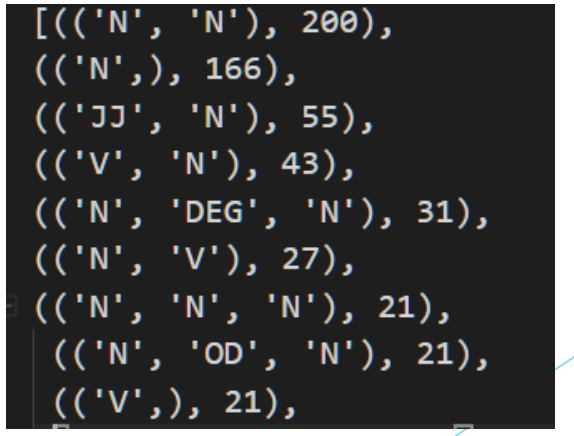
- 首先，在前面部分有提到我們蒐集了1000多部電影資料，而這些電影都屬於IMDB評分較高的電影，因此我們假設這些電影命名方式都屬於人們較喜歡且吸引人的，於是我們以這些電影為基礎去做分析，進而發想了Title Evaluation的方式。

Title Evaluation
Final Version (cont.)

- Base: 分析dataset的電影標題(蒐集而來的1000多部熱門電影)
- 拆成四個feature去分析並且量化成數值，分別為：
  - f1) POS tag
  - f2) movie title frequency
  - f3) Word2vec: similarity to top 1000 movies
  - f4) Title length: 3, 4, 5 words are most popular
- 而對於每一個新進來的電影標題，我們都會去計算這四個features的數值，進行微調權重參數(wi)加權後做加總
  - score = $w_1*f_1 + w_2*f_2 + w_3*f_3 + w_4*f_4$
- 並依據此分數去做電影名稱排名，分數越高，排名越前，屬於較吸引人的電影命名
- P.S. 後來我們發現熱門電影跟影片標題長度不一定有一定關係，因此在最後我們將第四個feature捨棄

Title Evaluation
Feature 1

- 分析常見的詞性組合，若電影名稱屬於熱門的詞性組合，則分數越高。
- 計算方法：
  - 右下圖為我們分析1000多部電影前幾名常見的詞性組合，及出現次數(x_i)
  - 而feature1的分數即為feature1 = x_i / (x_1+x_2+.....x_n)
  - (i = 此電影的詞性組合)



Title Evaluation
Feature 2

- 分析1000部熱門電影Title常用的用詞，若電影標題有用到這些用詞則分數越高。
- 計算方法：
  - 首先我們先對現有熱門電影名稱斷詞並計算詞頻，產生一張詞(w_1~w_n)及詞頻(f1~fn)的對應表
  - 當一部電影標題組成為($word_1$, $word_2$, ... , $word_k$)， 再去對應表中各詞的詞頻(f_i)做計算加總
    - feature2 = (f_1+f_2+...f_k) / total
    - total = (f_1+f_2+.....+f_n)

Title Evaluation
Feature 3

- 為了避免有些詞並沒有在feature2產生的詞頻表中出現而導致電影標題計算的分數為零，因此我們發想了feature3，只要電影標題有與熱門電影的名稱命名**相似**即會有分數。
- 計算方法：
  - 採納word2vec的Model計算字詞的Vector
  - 將電影標題與1000部熱門電影標題計算Cosine Similarity
  - 取Cosine Similarity最大的值作為feature3的分數

Demo Results
Task 1

1. 黑道
   - 原始標題: 艋舺
   - 分類: 劇情 (符合)
   - Candidates: 黑道廟口、黑道灰狼、蚊子角頭、外省黑道...
   - 分析: Keyword除了黑道、角頭之外，也篩選出人物綽號，其原因推測是在電影中多次使用提高TF。
2. 愛在電影
   - 原始標題: 等一個人咖啡
   - 分類: 浪漫 (符合)
   - Candidates: 真愛電影、女朋友、大白菜、愛在暴哥、愛在伯伯...
   - 分析: Keyword類別非常多樣。

Demo Results
Task 1 (cont.)

3. 畫畫的故事
   - 原始標題: 魯冰花
   - 分類: 劇情 (符合)
   - Candidates: 乙班、小朋友、貓咪乙班、茶園乙班、老師乙班...
   - 分析: Keyword圍繞在校園，但組合並沒有畫畫的故事來的好。
4. 紅翼
   - 原始標題: 紅衣小女孩
   - 分類: 劇情 (不太符合，應為驚悚恐怖)
   - Candidates: 紅翼天鵝、鬧鐘全名、便當鬧鐘...
   - 分析: 這部電影標題其實不算成功，不僅分類錯誤，關鍵字也無法對應劇情大綱，但觀察我們經由task2 model計算的分數也符合預期都不高。

Demo Results
Task 1 (cont.)

5. 葉師功夫
   - 原始標題: 葉問
   - 分類: 劇情 (有點符合，動作或劇情)
   - Candidates: 問叔功夫、白米功夫、師傅功夫、師傅葉問...
   - 分析: 功夫幾乎皆為Candidate，葉師看起來並沒有斷詞正確，可以發現師傅葉問在人為判斷上應更適合標題。
6. 咖喱
   - 原始標題: 食神
   - 分類: 劇情 (不太符合，應為喜劇)
   - Candidates: 雞姐咖喱、咖喱牛丸、雞姐豬皮、咖喱、豬皮...
   - 分析: Keyword裡面，tf-idf高的大部分都是食材名稱，例如：咖喱、豬皮、牛丸

Demo Results
Task 1 (cont.)

7. 功夫風暴
   - 原始標題: 少林足球
   - 分類: 恐怖 (不符合，應為喜劇、動作)
   - Candidates: 功夫風暴、瘸子風暴、或然率風暴、嫂子風暴、球鞋風暴、輕功風暴、絕命功夫...
   - 分析: Keyword有功夫出現，但是因為分類的關係，所以組合有風暴、絕命，沒有抓出足球關鍵字。
8. 琴房的故事
   - 原始標題: 不能說的秘密
   - 分類: 劇情 (符合)
   - Candidates: 琴房的故事、男朋友、小雨琴房、琴彈、小雨...
   - 分析: Keyword有琴房跟小雨，蠻符合電影主題的，加入'的故事'做組合。

Demo Results
Task 1 (cont.)

9. 目擊者
   - 原始標題: 目擊者
   - 分類: 劇情 (符合)
   - Candidates: 目擊者、舉發、同學、仲文車禍、總編車禍...
   - 分析: Keyword抓出了目擊者，電影裡面出現了五次，剛好符合原本的標題
10. 星空
   - 原始標題: 星空
   - 分類: 喜劇 (符合)
   - Candidates: 星空、聖誕、 臭屁、同學、爸媽、宇傑、素描
   - 分析: Keyword抓出了星空，電影裡面出現了十九次，剛好符合原本的標題

Demo Results
Task 2

- 以下為我們的執行結果，依據排名的結果排序 (id, 電影標題, score)
- 從結果我們有個有趣的發現，以(N, N)詞性組合的命名排序較為前面，而(ADJ, N)或(N)詞性組合的命名排序結果普遍較為後面。我們推論這是由於我們feature1的權重設定較大，才會有這樣的結果，但這樣的結果似乎也挺符合人偏好的命名方式。

```
('9', '明日帝國')   1.29691365078
('3', '女王密使')   1.14794602044
('6', '黎明生機')   1.07363410264
('5', '最高機密')   0.933507126734
('2', '霹靂彈')   0.800939400048
('7', '殺人執照')   0.774160713516
('4', '太空城')   0.726776045156
('8', '黃金眼')   0.688565396287
('10', '皇家夜總會')   0.605294744263
('1', '金手指')   0.57282139959
```

Other Suggestion

- Demo方式
  - 因為demo時間長，有些規定也難以實現的關係，我相信很多組都有用和我們一樣random或是其他方式改變Model使其產生的Title可以更符合預期且更適合那10組Testing Data(至少我有問到有一組也是這樣random)，而那組把Testing Data加回Training產生和原本片名一模一樣的Title的就更誇張了，如果可以改其他demo方式，來禁止鑽漏洞或許會更好(例如一組一組去找助教現場跑一次，但助教的時間成本可能會很高)。
- Demo測資
  - 助教給的Task 1測資完全不包含西洋片，如果有的話可能會更有趣。
- 雖然demo有一些小意外，但是我們真的覺得這個活動很有趣，真的很感謝助教和老師花那麼多心力在辦這個Final Project的活動。

THANK YOU