卍煞氣 der 新世紀高速學習輔助系統卐

夏誌陽 b00901166　游書泓 b00902107　陳亮傑 b01902112

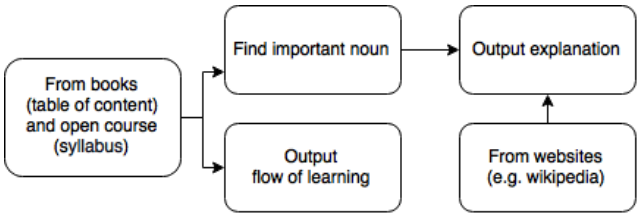
1 簡介

學習一個新領域時，搜尋結果往往出現各式各樣的專有名詞。面對如此繁雜的名詞，使用者只能慢慢查詢每個名詞的意義，或從許多書中挑一本來讀。然而這是相對低效率的學習方式。本系統專書目錄、開放式課程課綱等資料中整理出重要關鍵字以及推薦學習順序，以輔助使用者學習時快速進入狀況。

2 系統設計

本系統的架構如圖片。系統總共由以下元素組成：

- 文件 Corpus D。其中一份文件 $d_i \in D$ 代表一本書或是一個開放式課程。
- 每份文件 $d_i \in D$ 是一個序列 $d_i = (s_{i,1}, \ldots, s_{i,m_i})$，代表書的目錄或是開放式課程的課表。$s_{i,j}$ 是個字串，為書本目錄的一個 section 或課程進度表的一個項目。
- Search 函數。給定一串搜尋關鍵字 k，Search(k) 是所有包含這些關鍵字的文件的集合。
- GenerateKeyword 函數。給定文件子集 $D' \subset D$，GenerateKeyword(D') 會回傳是文件子集 D' 中的重要關鍵字，並會依據重要性做 ranking。
- GenerateFlow 函數。給定文件子集 $D' \subset D$，GenerateFlow(D') 是將 D' 中文件的目錄或課表適當地 cluster 與 ranking 形成的整合目錄。



2.1 資訊擷取

我們的資料分別從 Oreilly.com、Apress.com 以及 MIT open course 中擷取下來，包含書籍的目錄以及開放式課程的課程大綱。最後蒐集資料有 Apress 5419 筆、OReilly 1773 筆、MIT Open Course 1522 筆，共計 8714 筆資料。

2.2 名詞搜索

令 W 代表整個相關文件 D' 的單字集，包含所有 unigram 至 4−gram。則 GenerateKeyword 函式的演算法如下：

function GenerateKeyword(D', *limit*)
  $counts \leftarrow \forall w \in W, (w \mapsto |\{d \in D' : w \in d\}|)$
  for $i = 4$ down to 1 **do**
    $W_i \leftarrow W$ 中的 $i-$gram
    Sort W_i by $counts(w) \times \mathsf{IDF}_w{}^2$
    $W_i \leftarrow W_i[1..limit]$
    for any $w \in W_i$ **do**
      for any non-trivial substring v of w **do**
        $counts(v) \leftarrow counts(v) - counts(w)$
      end for
    end for
  end for
  return top *limit* ranking of W, sorted by
    $counts(w) \times \mathsf{IDF}_w{}^2 \times i$ for an i-gram w
end function

在排序過程中，我們透過 IDF 將出現頻率過高又不重要的字排除。在最終的排序中，我們將一個單字是幾個字組成也考慮進去，將較長的字排到前面。這點體現在最終排序的權重：給定一個 i-gram w,

$$weight(w) := counts(w) \times \mathsf{IDF}_w{}^2 \times i$$

此外，對於一個排序較高的長字串，我們也會將其子字串的頻率扣除長字串的頻率。原因如當輸入有 `content based image retrieval` 等較長的專有名詞時，連帶其子字串 `content based image`、`based image` 等等出現頻率也被拉高，但本身卻是無意義的字。若其子字串原先就有意義，則在此之外必須有其它不屬於 3-gram、4-gram 的出現次數。為了避免出現次數被扣到 0，每次我們扣掉出現頻率時，僅取 W_i 前 *limit* 個字串來扣。

2.3 流程排序

我們想要幫所有的 $s_{i,j}$ 做 clustering 和 ranking，建出合理的學習流程。

令 W 代表整個相關文件 D' 的單字集，包含所有 unigram 及 bigram。則 GenerateFlow 演算法如下：

function GenerateFlow(D')
  for $w \in W$ **do**
    $bucket(w) \leftarrow \{d \,|\, d \in D', w \in d'\}$
  end for
  $G \leftarrow$ Graph with $V := D'$ and $E = \emptyset$

for $w \in (W \setminus \text{BUZZWORDS})$ **do**
  for all $d_i, d_j \in bucket(w)$, **do**
    $d(d_i, d_j) < \text{THRESHOLD}$
    Connect d_i and d_j in G
  end for
end for
$groups \leftarrow$ connected components of G

Sort $groups$ by $\frac{\sum Rank(s_{i,j})}{|group| \times |\{d_i \mid \exists s_{i,j} \in group\}|}$

Remove groups of size $|group| \leq \text{GROUP SIZE}$
end function

2.3.1 Clustering with word vector

我們定義兩個 $s_{i,j}$ 的距離如下。如同 VSM，每個 $s_{i,j}$ 拆成很多 word (實驗中包含 unigram 及 bigram)，並建立 word-indexed vector $v' = (weight(w))_w$。對於每一個 word w，$weight(w) := \mathsf{TF}_w \times \mathsf{IDF}_w$ (其中 TF_w 僅計算 w 出現在 $s_{i,j}$ 中的次數)。最後我們用兩個 normalized-vector $v_{ij} := v'_{ij}/||v'_{ij}||$, $v_{pq} := v'_{pq}/||v'_{pq}||$ 之間的 cosine distance 來當 $s_{i,j}$、$s_{p,q}$ 之間的相似度。

接著以雙層迴圈 $O(N^2)$ 檢查所有的 $s_{i,j}$。對於所有 i, j, p, q，若 $\cos(v_{ij}, v_{pq}) > \text{THRESHOLD}$ 就將兩個 $s_{i,j}$, $s_{p,q}$ 連邊。

最後我們取大小超過 GROUP SIZE 的 connected components 作為我們的 cluster 完的目錄條目。

2.3.2 Bucket Optimization

因為對於不少關鍵字如 `machine learning`、`algorithm` 等，相關文件 D' 的目錄 $\{s_{i,j}\}$ 數量個數 N 約為 10^4 量級，因此 $O(N^2)$ 的算法效率有所不足 (尤其本系統以 python 實作)。作為優化，我們用字集 W 建立 $bucket$，並在 $bucket(w)$ 中存入所有包含 w 的項目 $s_{i,j}$。接著我們便可將 2.3.1 中建邊的動作單獨限制於每個 bucket 中。

假設每個 bucket 中 $s_{i,j}$ 個數為平均為 m 個，且 $n \approx |s_{i,j}|$，由於每個 $s_{i,j}$ 約出現在 $2|s_{i,j}|$ 個 bucket 中，所以計算的複雜度會變成 $O(m^2 \times \frac{N \times n}{m}) = O(mnN)$。實驗結果平均 $|s_{i,j}| = 5, m = 10$，因此絕大多數時間演算法效率接近 $O(N)$。

正確性 如果 $\cos(v_{ij}, v_{pq}) > \text{THRESHOLD}$，表示 v_{ij} 與 v_{pq} 至少有一個相同單字 (不然 $\cos(v_{ij}, v_{pq}) = 0$)。令 w 為 $s_{i,j}$ 與 $s_{p,q}$ 的共同單字，則我們有 $s_{i,j}, s_{p,q} \in bucket(w)$。因此建邊的演算法並不會少建 (或多建)、連通塊仍然一樣。

2.3.3 Group Ranking

對於每一個 $s_{i,j}$，我們可以利用其原本在書中目錄的排序 $j/|d_{i,j}|$ 來當作其在 $d_{i,j}$ 中的 rank。每個 group 便可以其擁有的所有 $s_{i,j}$ 的 rank 做平均來排序。

但實驗中亦發現存在 $w \in s_{i,j}$ 滿足 w 只於某本書大量出現，而且章節也很前面，導致 $s_{i,j}$ 的 group 雖然不是重要內容，卻被排序到前面。所以我們將 group 中有幾個「不同」的 $d_{i,j}$ 也考慮進去，並以此 penalize 僅出現於少數 $d_{i,j}$ 的 group。

最後 group 的 rank 我們定義如下。

$$\frac{\sum Rank(s_{i,j})}{|group| \times |\{d_i \mid \exists s_{i,j} \in group\}|}$$

2.3.4 Buzzword 過濾

Clustering 過程中也發現很多 $s_{i,j}$ 被一些教科書、課程常出現的 BUZZWORDS 像是 Introduction、Chapter、Midterm 等等 cluster 在一起。我們可以透過直接忽略 BUZZWORDS 的 bucket 解決這個問題，避免 $s_{i,j}$ 利用 buzzword 被 cluster 在一起。而未被 cluster 在一起的 $s_{i,j}$ 會被 GROUP SIZE 的門檻移除。

2.4 使用者介面

使用者可以在網頁上輸入欲搜尋的關鍵字 (或者點選好手氣，由系統隨機挑選關鍵字)，送出後在網頁上顯示名詞搜索和流程排序的結果。顯示的結果分成三個部分：

- Terms(名詞搜索結果) 畫面上會顯示至多 50 個相關的重要名詞，點擊每個名詞標籤後會在上方顯示該名詞的解釋，名詞解釋是從 Google 搜尋中獲得跟維基相關頁面中，第一個維基頁面內的第一段 (`p` 標籤) 內的第一句話取得。另外也有提供使用者自行搜尋其他名詞的欄位
- Table of Contents(流程排序結果) 由上至下顯示流程排序結果，點擊每個標籤會在右方的 Relative Link 中列出有哪些書或線上課程中有出現這個主題。
- Relative Link 點擊每個標籤會展開內容，裡面會列出書或課程內跟從 Table of Contents 點擊的主題相關的標題，以及網頁連結。

前端用到 jQuery、Materialize，後端網頁伺服器是 Python Tornado，並且主要的網頁伺服器跟名詞解釋的伺服器是分開的，名詞解釋的部分是另一個 Tornado websocket 伺服器。

3 結論

這系統說真的，很煞氣。

4 致謝

老姜好威感謝老姜