Week 2 - Scoring, Term Weighting and the Vector Space Model

在了解如何將字從文件中抓出來後，接下來要**index**文件。btw, 文件會將其代號(docID)。

**Boolean Retrieval Model**

1. **Incidence matrix**

* Boolean query: xxx AND yyy AND NOT zzz

🡪 110100 AND 110111 AND 101111 (**0 to 1, 1 to 0**)

* 最不合適的方法，column和row分別為docID以及term，優點是極為直觀，但缺點就是會存了一堆0。(the matrix will be very sparse)
* Example: 1M篇doc、500k個term，假設每個entry存1bit，這樣一個matrix就要用掉58GB。()

1. **Inverted index**
   * Record only the things that do occur
   * 前處理完會產生Dictionary(lots of terms)，後面會有postings list，紀錄該字出現在哪些文章。
   * 假設今天你的IR system是fixed的，你的dictionary不會動，則postings list 會放在memory的array(sorted)來做。(文章太大則會放在disk裡)
   * 但假設是incremental的話，則會用B Tree。
   * Why called inverted?

🡪以前看文章to meet user query，但現在是反過來做，是看使用者要的東西在哪裡，回推到文章中。

* + 建dictionary和postings list的順序為每篇文件先normalize tokens，並且刪掉重複的，但在刪掉重複的時候時，紀錄該文字出現在這文章幾次(**之後會要討論權重問題**)
  + Implement: Postings list要先sort過(也就是說sorted by docID)，接著是**不相等時動小的**，小的往下一格，當他們都一樣時，把答案加進去，一起下一格。

1. **Problems:** 第一，當資料量很大時，你的結果也只是丟出一個次大的東西，使用者也沒辦法用。第二，假設user query = a AND b AND c，但可能c不是很重要的東西，這樣就濾掉像是豐富記載a和b的文章。簡單說**二分法只能分對錯，不能partial match、沒辦法排序**。

**Then how to assign a score to a query?**

1. **Parametric and zone indexes**
   * Usually used by the **library**
   * Metadata: data about data(document), for example, a book has the authors, title, and so.
   * Generally, there are two types of metadata:
     1. Field: possible value is **finite**, such as date. 🡪 通常是y/n，不用排序
     2. Zone: **free text**, such as title. 🡪 要排序
   * **Weighted Zone Scoring**
     1. 給各項一個權重，舉例來說書有三個zones: author, title, body。並且給author 0.3的權重，title 0.2的權重，body 0.5的權重，最後看誰分數高。
     2. 權重怎麼得? 🡪 1. 專家 2. ML
     3. 所以這其實也是y/n 問題，有的話就拿這佔比，沒有的話就0，所以又稱為 **ranked Boolean retrieval**。
2. **Term weighting and vector space model**
   * **Term Frequency (TF) and Weighting 頻率**
     1. TF 的下標有d是因為他**針對single document**
     2. 依照關鍵字出現頻率來決定重不重要。
     3. Free text query: 使用者隨意去打，看他的關鍵字match到哪些文章。
     4. It will produce **the bag of words.**
     5. Problems: **the ordering is ignored!** 🡪 ‘Mary loves dogs’ == ‘Dogs loves Mary’

* **Inverse Document Frequency (IDF) 鑑別力**
  1. DF (document frequency) 的下標沒d是因為他是**針對全部的document**
  2. 有些字沒有鑑別力，像是and、the等等
  3. idf = log(N/df)，其中N是定值，表文本數量。若每篇文章都有某個字的話，則idf = log(N/N) = 0，則該字最沒鑑別力。
  4. 取log是因為要維持差異性，但不要這麼大。(真正在操作上N通常超大，取完log df的影響比較大)
  5. Collection frequency (CF): 全部N當中，**出現了幾次**。但其實不常用，比較常用DF。
* **TF-IDF weighting:** tf-idf = tf \* idf
  1. **夠不夠頻繁\*夠不夠有鑑別力**，可避免stop word.
  2. Bias: 長文章易擺在前面，因為其tf易較高。
* **Vector Space Model**

1. View each document(query) as a vector
2. Each term corresponding to one dimension 🡪 通常有20000+ dimensions
3. Use the concept of TF-IDF to weigh
4. Lose the relative ordering (identical to the bag of words)
5. How to quantify the similarity between two documents? (why ask for similarity? **for ranking**)

🡪 V(d1)-V(d2)向量相減，概念就是:**你的第一維term和我的第一維term權重差多少，你的第二維…**繼續下去取絕對值&summation

🡪 disadvantage:長文章有高權重(又回到TF的概念，畢竟用到其概念去做權重)

1. **Cosine similarity**: 分子為向量內積，分母為兩向量的長度，也就**normalize**它的長度，如此一來就能避免長文章的優勢。

🡪何時會大?你大我也大，你重要我也重要

🡪=1? 兩個向量一樣；=0? orthogonal