DS\_final\_report

姓名：鄧弘利

學號：111062107

# How you implement your code

我事先在每一次執行query搜尋項目前，先對傳進來資料夾中的所有txt檔案，先建立trie，並且，我額外建立了一個每一個字的輸入都是顛倒的trie\_reversed，這個的原因是因為我想要在做suffix查詢的時候，用trie\_reversed查比較快速。並且我在全域設了一個儲存trie還有trie\_reversed的vector。我有先對傳進來的資料檔名先進行排序處理，所以陣列儲存順序就是從0開始，一次加一(0.txt存在tries[0]還有tries\_reversed[0]，1.txt則是tries[1]還有tries\_reversed[1])。在建立Trie的時候，也有一些額外的處理，會先讓輸入的字串，去掉不是英文字母，還有都轉成小寫。針對每篇文章的標題，我是用vector<vector<string>> Titles去存。至於query的部分，我會每一次讀取query的一行搜尋項目，將遇到的+ - /先儲存在一個op的queue中。並且將被運算符分開的字串儲存在search\_words這個queue中。接下來要決定輸出符合條件的檔案名稱的部分，我會跑一個for迴圈，去一個一個跑剛剛建立的trie。每次先取search\_words的top字串並pop，接著丟進找尋字串是否在trie的function，該function回傳的是bool，如果是true代表有找到，false則是沒有。接下來跑一個while loop，直到search\_words變為empty才結束。每次從search\_words的top取出字串並pop後就丟進找尋的function，接著從op pop出指令，進行邏輯運算，並將結果儲存在valid。最後如果valid為1代表該篇txt有符合搜尋條件，否則沒有。當valid為1，就把該檔案的標題存在titles中。然後接著跑下一個trie，直到跑完。跑完之後把titles裡面符合搜尋條件的資料輸出。至於是怎麼搜尋的，我分別建立了三種的搜尋function。

第一個是找prefix的方式。Ex . graph。這個就比較基本，就是從root開始，針對我們丟給它要找的字串，去找，如果都找到就返回true。

第二個是找exact word。Ex .“graph”。這跟第一種方式差不多，但是找完之後還要去確認是不是一個字的結尾。

第三個是wildcard的搜尋。Ex . <com\*on>。這個主要是透過遞迴的方式去找，一個情況是跑忽略\*，另一個是先保留\*。

# Other implementations for optimization

在找suffix的時候，因為我已經建立trie\_reversed(建立資料時每一個輸入的字串都是翻轉的)，所以我只要將query要找的字串先翻轉，再丟入找prefix的search function，並且是找trie\_reversed，這樣就可以達到效果。另外在建立Trie的時候，我會先將文章中的文字轉為小寫，還有忽略不是英文字母。

一張含有 文字, 螢幕擷取畫面, 軟體, 作業系統 的圖片

自動產生的描述

# Challenges you encounter in this project ,or Conclusion

遇到困難的地方，主要在於一開始不熟悉Trie的運作方式，還有整體資料結構運作的方式。另外在做找wildcard的function時，也讓我困惱了很久，不知該怎麼做，是去網路上查詢別人是怎麼實作之後，才有了初步的概念。

# References that give you the idea (github/paper…)

Trie (字典樹)

<https://ithelp.ithome.com.tw/articles/10324306>

geeksforgeeks

<https://www.geeksforgeeks.org/trie-insert-and-search/>

stackoverflow

<https://stackoverflow.com/questions/8145552/simple-wildcard-search-algorithm-in-c>

github

<https://github.com/jamesonwilliams/wildcardtrie/blob/master/src/main/java/org/nosemaj/wildcardtrie/WildcardTrie.java>

<https://github.com/jamesonwilliams/wildcardtrie>