BMI Practica 2

Leah Hadeed & Lorenzo Vela

1. Para esta práctica, hemos optado por hacer los siguientes ejercicios:

* 1.1 Metodo orientado a términos
  + Para este ejercicio, codificamos la función de search del TermBasedVSMSearcher. Recorremos la lista de los términos del query y usamos un diccionario para guardar los acumuladores de cada termino. Para luego ordenarlos, usamos un heap para sacar los primeros N términos hasta el cutoff dado.
* 1.2 Metodo orientado a documentos
* 1.3 Heap de ranking
* 2 Indice en RAM
  + Para este ejercicio, codificamos el Builder y el RAMIndex. Para el Builder, guardamos la lista de postings en un diccionario Python con clave termino y valor tupla docid y frecuencia. Usamos la librería de pickle para guardar los datos en fichero (POSTINGS\_FILE). Luego guardamos los nombres de los documentos en un fichero aparte (INDEX\_FILE). Cuando creamos el RAMIndex, leemos de los ficheros y metemos los documentos en el docmap y calculamos los módulos de los scores.
* 6 PageRank
  + Para este ejercicio, seguimos el algoritmo de clase de PageRank. Creamos dos diccionarios de conexiones (in y out), y los rellenamos leyendo del fichero de grafos dado con clave nodo de entrada/salida y valor lista de conexiones de entrada/salida. Tomamos la unión de claves de los dos diccionarios para encontrar los sumideros. Para calcular el valor P’ de los nodos, calculamos la suma del valor de las conexiones de salida con el siguiente calculo:

|  |
| --- |
|  |
|  | for k in keys: |
|  | self.p[k] = 1/N |
|  | #print(self.outConnections) |
|  | sinks = self.inConnections.keys() - self.outConnections.keys() |
|  | # Begin iterations |
|  | for n in range(n\_iter): |
|  | for k in keys: |
|  | self.p\_p[k] = (1-r) / N |
|  | for i in self.outConnections: |
|  | for j in self.outConnections[i]: |
|  | self.p\_p[j] += (r \* self.p[i] / len(self.outConnections[i])) + (r \* self.p[i] \* len(sinks) / N) |
|  | for k in keys: |
|  | self.p[k] = self.p\_p[k] |

1. El siguiente diagrama muestra las clases implementadas para la práctica.

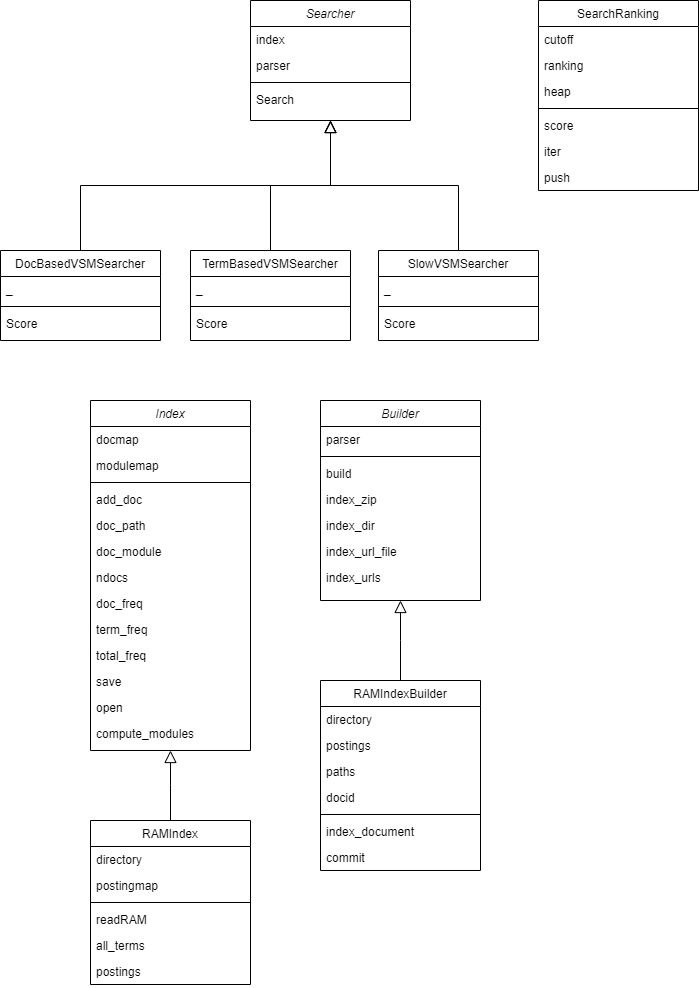


Figure 1: Diagrama de Clases

Los searchers heredan de la clase abstracta Searcher y en la función score, calcula el ranking utilizando la clase SearchRanking. Como tenemos implementado además el heap de ranking en la clase searchranking, ordenamos los documentos utilizando esa misma clase.

Para los índices, como solo creamos el RAMIndex y su Builder, estas clases heredan directamente de Index y Builder con algunas funciones sobreescritas para reflejar la nueva funcionalidad del índice. El RAMBuilder crea el índice y lo escribe en un fichero, mientras que el RAMIndex lee del fichero y rellena los campos de este, como el docmap y el modulomap.

1. A continuación mostramos una tabla de rendimientos:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Construccion del Indice | | | Carga del indice | |
|  | Tiempo de indexado | Consumo max RAM | Espacio en disco | Tiempo de carga | Consumo max RAM |
| Toy1 |  |  |  |  |  |
| Toy2 |  |  |  |  |  |
| 1K | 48.27505898 |  | 13943669 | 2.07007885 |  |
| 10K | 338.3613372 |  | 95847901 | 15.33669233 |  |
| urls | 2.071116686 |  | 347171 | 0.131609917 |  |