Omsairam

Prof chris pollet [chris@pollet.org](mailto:chris@pollet.org)

Video available after class

Syllabus - prerequisite

* How records are stored in file system
* B trees
* IR implementing and evaluating search engines – 2016 book
* measure effectiveness

Examples of IR– desktop search, Mekvyl – library catalog, google search, Turnitin.com-> ur paper vs other papers similar to you

Search engine can also be made in mysql -> difficult to scale

Ex-Fb had mysql IR earlier

Inverted index -> word and list and intersect the list

Conjuctive queries -> ‘and’ all words in search used for search

Precision and Recall: how many u got back and how many relevant

Ranking system for result retrieved

Query retrieval time faster : index compression techniqyes to sqeeuze search query and retrieve maximum

Log structured and merged tree : keep to add new data rather than recompute whole thing

chatGPT ; might not necessarly be correct so use better IR to get better result

augment google with language model to get better result

Homework:

Readme file wth people worked on i. less than 4

Lec-01

Slidy software used for ppt , it has HTML5 and javascript uses slidy to look like slides. It passes WAVE (online test for web accessibility check)accessibility.

Information Retrieval -> Representation, search and manipulation large amounts of data

File web desktop search, categorization, aggregation, clustering, answering, multimedia, summary – all examples of IR

Searching in finder

Archieve.org -> look what web pages looked like back in time

“Way back machine” is above example

Document filtering -> filter based on medical article like cancer etx, text clustering -> same types in one category

Category->Spam and others

Multimedia -> search in YT etc

Search engine has a list of documents so we index. Some documents are deleted , some added

Query might not give information so use query to fnd the topic of interest

Way a simple IR works :

User -> needs a ‘topic’-> queries to search engine -> search engine accesses its inverted index store(word: docs maps) -> ranks documents , removes duplicates and sends to user

Lec -02

Order information in the documents, position of each word , so size of inverted index is almost same like document size. Can reconstruct the original document from the index map.

Duplicate documents: Urls mapped to same documents. Last updated.

To remove duplicates :Relcolonical attribute, same page ur colonical version of the page should be indexed

Robots.txt file tells whether to index or crawl it or not but doesn’t say if that’s already crawled

Documets: cn be a web page, pdf etc, might be retruned to user as search result, email msg, etc

Sub-part can be also returned as a search result.

Snippet : title of web page , where term appeared on the webpage

Performance evalution :

Efficiency(sec/query) and effectiveness(byte/documents)

Effectiveness -> relevant

Binary -> 1 relevant , 0 not relevant

Graded -> how much relevant 0.x

Inverted index example:

Doc 1 The cat is hungry

Doc 2 The dog is barks

Barks : (2,3) -> doc 1 pos 3

Cat: 1,2

Dog: 2,2

Hungry: 1,4

Is: 1,3

The: (1,1) (2,1)

Fully positional index : word and its (documents, index position)

Number of bytes to write the inverted index is almost same as the bytes of the corpus.

Ranking a document: prob. Ranking principle. Decreasing order of relevance

Ex: prime minister: query

Order change per time for ranking the documents

Specificity -> specific part of the document, what is returned , only important thing specific of the query

Exhaustivity -> degree of information it retruns, how many results

Novelty -> how new

SCO -> search engine optimization manage online image of people. Bias score, using sentiment analysis

A search based on the type of data or filters to use based on the search . classification label to the search engine. Json to the search result

Text formats: unstructured or semi structured data

For strcuturd can use mysql etc,

How to create the inverted index :

Can use json too.

All Microsoft docments are zip of xml files.

Headless browser -> crawl the web page, slow. Indexed less frequently.

If dynamic web page then don’t get indexed normally. Crawlers are more specialized for frequently changing pages.

Take documents -> tokens -> create the inverted index

Remove formatting pages, font information etc, convert ASCII, Big-5 to UTF-8

Stremmer -> gets roots out of words

term-frequency table.

Content size

Descrption: most relevant 1 KB

Ranking per word

Timing used

Hash code to indicate documents

Ip address

Links extracted

Only words from summary gets into the inverted index

How important the word is to a document: language modeling

Probability distribution : function from sample space in range [0,1]

Lec -03

Unknown work predict:

V’= V U (unknown)

Extend our language model

Language model: M’(unknown) = B

…see book

Example:

A ball -> unknown

Known work

A cube

A pyramid

<a, ball, cube, pyramid>

As in the document half of the words are a and half of the words are ball

<0.5, 0.5, 0, 0> -> new text vector

<0.5, 0, 0.25, 0.25> -> corpus text vector

Euclidean distance between new text and corpus , then

Then standard deviation between vectors in corpus and view some fixed deviations to decide if that’s present in unknown

MLE (max likelihood expectations): 0th order model

First order Language m odel : see seq of terms

M(q|b) means what is the probability of seeing q that we have seen b

S=small, l = large corpus

Types of term distibutions:

1. language model
2. Markov model

Omsairam

Lec-04

url rewriting : same looking url, rewrite to a different url that says what actual url being used.

Cd /users/cpollet/

Php index.php

Localhost/8080

username : root , no password

robot names shows but no the instance.

Robot.txt -> can be configured to crawl

Queue-> what to crawl next urls

Doubt: unable to get data from Wikipedia from hw1 > updated

omsairam

Lec-05

Name server -> nodes all the queue servers -> like a web app

Queue nodes -> keeps part of the queue for a web, what to crawl and index the stuffs

Web apps -> in between queue and fetch

Fetchers -> download and process pages using urls, then result sent back to queue server

Send folder -> tries made to download batch,

Fetcher gives web app what all are downloaded that’s moved to receive then to index to index the pages.

Robots -> robot.txt

Etag -> how long a page will remain before expiry

Analytics about crawl (#urls, speed, how many urls < new updated pages, ),

Querying analysis : on the search, (query info: what linked clicked (page number) )

Media updater does this :Search sources : website rssp, like tedtalks download : everyday and appear as link on the wiki page of yioop

Steps to process a page :

1. Downloaded web page -> has html. Mediawiki underline software of Wikipedia, wordpress is another etc. create wen scrapers to extract content of a type, xpath -> real info of the page. delete xpath-> removed unwanted info. Challenge : pages change underlying important xpath, so needs to be periodically updated. Scraper for media content -> fb open graph type to get media content, inject meta word if see video, can search using meta word. And can add the meta word to the index matrix , for better query search. Conversion standard: mp4 for video, jpeg for images. Webp more efficient for image processing than jpeg. News feeds: RSS feed (relies simple syndication)-> has sequence of tags like stories and contents, display sequence of podcast items. Regex-> html broken then use regex, feed podcast -> scrape RSS feed, download and wiki page to download the items. Scrape podcast -> if difficult to identity url like Video has different types of time sequence segments. Trending value -> price tracker etc check. Description source -> who did the video, how long etc.
2. Pico <file name.php> -> create php file, to close use control e then Y then enter key

Hello world -> simple copy mode print

Php hello.php -> run php file

Copy mode -> helloworld

Interpreter mode -> <?php ?>

Can switch between interpreter and copy model

Like below:

<?php

function foo()

{

?>hi2<?php

}

Don’t close as some issue like sending data etc

Downloaded web page -> scraper -> get main text -> summarizer -> most important text

Omsairam:

Php -i same as phpinfo()

In php any array is like a key value pair

[a,b,c] is actually 0->a, 1->b, 2->c

Print\_r better shows, but doe4snt show empty array’

var\_dump -> not pretty but shows empty array

only function def takes by ref

so, inc($a) caller

function (&$a)

need to use global $b to use global variable in a function

when u enlist an index, the index itself gets removed

omsairam

posix based system -> sun microsystem, linux, MAC, unix, windows. Env ironment variables used in the process.

$\_ENV example

Form : lke login , payment

1. Browser gives DNS msg to yahoo, isp knows who browser wants to contact
2. Browser gets ip address, open port sends SSL handshake, ip doesn’t know what browser wants. This is encrypted with SSL , so don’t need to re-encrypt
3. If browser understands zipped files then domain can provide compressed data
4. Standard HTTP now is 3
5. Request after ? of the url
6. ./ process where php is running . its home directory
7. Fgets -> either the line or the max bytes provided in the command
8. .= means concatenation
9. Search for the fox in woods, in the ADT example

t[1] = the, t[2]= fox, t[3]=in t[4]=woods, position = current position

lti = number of times ti occurs in the corpus

n = number of terms in the phrase

L = number of times most frequent term occurs in the phrase list

k = number of candidate phrases

l = number of times least frequent terms occurs in the phrase list

binary search can be used to next and prev O(n. k . log(L)) or O(n. l. log(L))

when can linear occur better efficiency as binary -> when all posting list are of same size

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| First and second term frequency | Binary search O(n.l.log(L)) | Sequential search O(n.L) | Galloping search  O(n.l.(log(L/l)+1)) |  |  |  |  |
| F=l and s=l | O(n.l.log(l)) | O(n.l) | O(n.l.(log(1)+1)) |  |  |  |  |
| F=l and s=3l | O(n.l.log(3l)) | O(n.3l) | O(n.l.(log(3)+1)) |  |  |  |  |
| F=l and s=6l | O(n.l.log(6l)) | O(n.6l) | O(n.l.(log(6)+1)) |  |  |  |  |
| F=l and s=9l | O(n.l.log(9l)) | O(n.9l) | O(n.l.(log(9)+1)) |  |  |  |  |

Omsairam

Binary Search = l = L

Scan when all are equal length

To reduce scan size ->

Types of inverted indexes:

Separate documents

1. Depending on the tags
2. Xpath – query for particular nodes in the documents
3. Split at doc level : use n:m, n= docid and m=mth tern within the document

Starting the term at 1 index not 0

The quick brown fox , doc =23 then briwn lies 23:3

1. TREC 45 corpus more relevenat to judge topics
2. Recall what fraction of relevant result is returned
3. Precision fraction of results returned were relevant
4. 167/881 precision in slides 167/202 recall
5. nextDoc(t,current) will return doc where the term t is found after document current, so cannot be the same current doc
6. use positional index for HW2
7. Vector space model : the fox <0,…0> -> this vector represents any real number for each vocabulary value . 1 position per word and a score per word. Vector fr doument and vector for terms.

1 if fox is present and the f fox is present

Multi dimensaional space . but required terms at only 1 rest all 0s

1. Similar then cos theta= 1, else its 0
2. Similarity between doc and query = unit vector doc \* unit vector query
3. Value to use in above vector = TF-IDF

Term freq – inverse document freq

Avg length of query -> 3 to 4

Log base 2 always here

Measure of how mnay bits of information is given by whether how many docuemnts have that term IDF

Small number get more information as log is used to get TF

1. Cosine rank : query vector create ho> frequency vector or TF-IDF difficukt as has to keep track of all th queries that’s ever been made
2. Use decibels as the unit for the log(N)/N .
3. Cover use the smallest sequence that has all the required terms
4. Why can all Boolean function be written as AND OR NOT ? use truth table to create operation function table . output is either a 0 or 1.
5. Positive Boolean query doesn’t use NOT

rankProximity(“all”, “the”, 2)

d1 = "Which is correct: all the people, all of the people, or all people?" and d2="All the king's horses and all the king's men".

Score=0

Next cover:[1:4,1:5] , u=4, v=5, d=1, score += 1/(5-4+1) = ½

Next cover: [1:5, 1:7], u=5, v=7, d=1, score += 1/(7-5+1) = 1/2+ 1/3

Next cover: [1:7, 1:9], u=7, v=9, d=1, score +=1/(9-7+1) = ½ + 1/3 + 1/3

Next cover: [1:9, 1:12], u=9, v=9, d=1, score+= 1/(12-9+1) = ½ + 1/3 + 1/3 + ¼ = 17/12 = 1.4

Score = 0 as document is different

Next cover: [2:1, 2:2], u=1, v=2, d=2 , score += 1/(2-1+1) = 1/2

Next cover: [2:2, 2:6], u=2, v=6, d=2, score += 1/(6-2+1) = ½ + 1/5

Next cover: [2:6, 2:7], u=2, v=7, d=2, score += 1/(7-6+1) = ½ + 1/5 + 1/ 2= 12/10 = 1.2

Ran function returns documents based on above score in descending order [1,2] // as score(1)= 1.4 score(2)= 1.2

Omsairam

To make recall 1 , return all the documents,

In slides recall = 167/202 . precision = 167/881

When we dint return anything, everything we returned is relevant so precision at k=0 is 1

Every point x on the graph, =k results returned

Res = d1, d2, d3, d4, d5

Rel = d2, d4 = 2

Avg. Precision(k) = ½ (P@2 + P@4)

= ½ (|rel and res{1,2}|/ |res{1,2}|) + (|rel and res{1,4}|/ |res{1,4}|)

=1/2 (1/2 + 1/2) = ½

Better AP means we found all relevant documents in the beginning itself. How close to front of the list we got the result.

Omsairam

At creation time of inverted index and query time we need to change the word .

What to use C++ to make it computer friendly -. Write in full names like cplusplus or cpp

Database indexing -> full text index or use spinx search , index a column and add quick search into it. Use to remove stopwords

Normalized : fearless furry flies

stemming: fearless furri fli

Example of recall increase and precision decrease when stemming used

Query1 Lung cancer -> stremmed to -> lung canc

Query2 Lung cancer treatment -> stemmed to -> lung canc treat

Say there are 100 relevant documents for lung cancer

Total retrieved for each query is 150

,

80 actually has relevant terms

20 is false positive

For Q1

True positive : 80 (80 relevant docs retrieved)

False negatives: 20 (20 relevant documents not retrieved)

Recall = 80/100 = 0.8

False positive: 70 (70 irrelevant documents retrieved)

Precision = 80/(80+70) = 0.533

For Q2

True positive : 90 (90 relevant docs retrieved)

False negatives: 10 (10 relevant documents not retrieved)

Recall = 90/100 = 0.9

False positive: 100 (100 irrelevant documents retrieved)

Precision = 90/(90+100) = 0.474

From above we can see that recall increased as it retrieved more variation of the query terms.

Precision decreased as it retrieved more irrelevant documents.

Php :

\_\_construct for constructor creation

Require command : cant find class then fatal,

Require “File.php” -> execute the php file.php

x number of times u call require File.php -> will execute x times

require\_once “File.php” -> execute only once

php namespace = java pkg

Same class twice define then error

Public or var = public visibility

omsairam

Define keyword for global constant

Static is for class , cant have within function only at class level

PhP

Instance of a class

Class A{// some func}

$a= new A();

//Primitive types

$a=1;

(immediate new memory)$b=$a;// new variable for b in memory occupied is a new one

//array

$a=[1,2,3]

(copy on write)$b=$a // no extra memory for b, if change b or a then create on copy and make the change. Copy on write semantics

$a= new A();

(only copies reference)$b=$a; // pointing to a

Different files can be put under same namespace

Default namespace = \

Strcmp or \strcmp

Built in functions are from default, if overridden then use namespace\func

Same function different namespace without declaring infront then error.

File import = require , require\_once just import once, include then only warning if no class found

Use : does load into memory ? does nothing . used in web env, so not actually importing .

Autoload -> brings into memory, figures out where files are

Need to use

Library::init() // if need to use yioop as a library

Omsairam

ASCII -> 7 bits

Nibble -> 4 bits

Hexadecimal : 4 bits, so 4 bits hexadecimal can represent = 2^16 bits

U+516B = binary for 0101 0001 0110 1011 = 5 1 6 8

In UTF-8 – first 4 determines how many decimals

1110 0101 10 000101 10 101011

First start first 4 digits from binary Con t. first 6 digits from binary cont. last 6 digits from binary

n-grams = n=5 , start .xxxx xxxxy xxxyy xxyy.

if 3 digits then .the.

if less ..is.

omsairam

hw3

yioop as library -> fetchURL.php to be used

stemmers and summarizers etc -> phraseParser

need compressed format use packedTableTools.php

use hex editor to see that bits are written to dick correctly -> file listings have raw bytes and can play games using hex. In unix : man hexdump to check.

Writes hexadecimal for every byte of the file, and prints as characters

Sort based . more general searching , but slower than hash based

Per-term indexing

PL: x1, x2, ….

At start has per-term index which has 2 components : offset to the 5000th index, between documents where to search then search in range , the range will be pulled to memory

Can fit in memory the entire per-term index.

This is one term in the pl: <D1:<p1,p2>>

So for per-term index min 5000 documents are present to even start with

Per-term index will point to 5000th document and its indexes

Omsairam

buildIndex

insert at back -> insert at end new term, as less freq for new entry less likely than earlier entries O(len of chain)

move to front -> if seen t’ again , move to front of chain as it may appear again o(len of chain)

* Suppose we have a hash-table of size 3, our terms are just integers, and we use the hash function n%3.
* Suppose our corpus looks like: 8 7 5 5 5 7 3 6 8 5 2 3.

1. 8%3 =2

0->

1->

2->(8:1) //insert at end for 8

1. 7%3=1

0->

1->(7:2)//insert at end for 7

2->(8:1)

1. 5%3=2

0->

1->(7:2)

2->(8:1), (5:3) //insert at end for 5

1. 5%3=2

0->

1->(7:2)

2->(5:3,4),(8:1) //move front for 5

1. 5%3=2

0->

1->(7:2)

2->(5:3,4,5),(8:1) //move front for 5

1. 7%3=1

0->

1->(7:2,6)//move front for 7

2->(5:3,4,5),(8:1)

1. 3%3=0

0-> (3:7)//insert at end for 3

1->(7:2,6)

2->(5:3,4,5),(8:1)

1. 6%3=0

0-> (3:7),(6:8)//insert at end for 6

1->(7:2,6)

2->(5:3,4,5),(8:1)

1. 8%3=2

0-> (3:7),(6:8)

1->(7:2,6)

2->(8:1,9),(5:3,4,5)//move to front for 8

1. 5%3=2

0-> (3:7),(6:8)

1->(7:2,6)

2->(5:3,4,5,10),(8:1,9)//move front for 5

1. 2%3=2

0-> (3:7),(6:8)

1->(7:2,6)

2->(5:3,4,5,10),(8:1,9),(2:11)//insert at end 2

1. 3%3=0

0-> (3:7,12),(6:8)//move to front for 3

1->(7:2,6)

2->(5:3,4,5,10),(8:1,9),(2:11)

Disk based sorting

Disk blocks = 4096 bytes

In RAM enough memory to store n disk blocks

Pass 1 sort n disk blocks

n-way merge sort

omsairam

for merge bases soring needed 2 runs max:

first pass to count runs

second pass to merge

how many open file handles can your system have ? constant set by the OS use ulimit

don’t open too many file handles

merge based indexing parallelly check for the file handle that has least next term to output

2 types of query processing:

Term at a time

doc at a time

binary heap is used to implement a priorityqueue

max heap highest value at root

min heap smallest value at root

to put to an array leave first slot and add the elements one at a time

if remove an element, cut an element and put to root and push it down in log time

i is root then 2i and 2i+1 are children

omsairam

genGCList(list){

 sort list by end index //nlogn

 start, end = list[0] // store first element of the list in start and end

len=list.len //length of list

 for i = 1 to len-1{ //n

   if end<list[i][1]{

    res.add(start, end)//create a new interval

    start, end = list[i] // put the start and end of the current list

   }

   else if(start<list[i][0]) {

start = list[i][0] //update start if start < new start

}

}

 return res

}

time complexity: O(nlogn)

//for term at a time query processing

Acc=accumulator tells score so far for each document

Acc’=score after term I so far for each document

First while loop sees for all docs that doesn’t have term I but present in acc’ because they can be part of top k

Inpos docid score outpos docid t[i]score

If docid isa same in both and the score is computed for all terms before t[i] then we add score to acc’ for that docid

**T := argmin\_x{x in Nat|**

**sum\_(j=x)^{max\_f}(tfStats[j] \* q) < quotaLeft**

estimated number of remaining documents with tf >=x look for smallest x such that this sum is < quotaleft.

Threshold : T , all documents whose score is >x estimated.

Omsairam

Operators

A <| B means a contained in B

“It was a”

{[1,3], [9,11]}

“it was”

{[1,2], [9,10]}

GC “it was” <| GC “it was a”

A^/ B where ^/ is a triangle

A={[3,5]}

B={[4,6]}

A^/ B = step 1 S={[3,6]} .. union of intervals

Step 2 G(S) = means remove the subset intervals from S to make it a GC

Av/ where v is upside down triangle

… check book

Lines in speech tag , speaker within speech too, and witch inside speaker

A…B means A before B

A[]

B[]

A…B = least u greater than u’ use that as u’’ and v’ as one interval where u,v are from A and u’,v’ are from B

Omsairam

Huffman coding

//sorted in decreasing order

(a4 {0.34}) (a3{0.31})(a1{0.18})(a2{0.11})(a5{0.06})

(a4 {0.34}) (a3{0.31})(a1{0.18})(a2,a5{0.17})

^

(a2{0.11})(a5{0.06})

(a1,a2,a3,a4,a5 {1.00})

^

(a4,a3{0.65})(a1,a2,a5{0.35})

^ ^

a4 {0.34}) (a3{0.31}). (a1{0.18})(a2,a5{0.17})

^

(a2{0.11})(a5{0.06})

Omsairam

Unary representation of 1 = 1

2 is 01

3 is 001

So y code is divided into uanary representation of length f binary counter part followed by binary counter part

Ek 16 is represented n binary as 10000 so its length is 5 and unary 5 is 00001

So y code for 16 is 00001 10000 and we premve redundant 1 from second string

00001 0000

Generalized compression techniques

Seen symbol before use that to compress. Remembering when at least 2 occurrence of symbols then use some rules to that occurrence and u can compress the whole data

Random string cannot be compressed.

Markov models types.

NP hard – halting problems cannot find sol to this.

Shortest prog that takes no arg and outputs that string: undecidable.

But same thing using turning machine is doable,

Why We know that the Golomb codeword for integer k+M is 1 bit longer than the codeword for the integer k

?

Because we dvide k+M by M which gives +1 values in golomb.

Vbyte

For 8

Binary 8 is 1000 can fit in 1 byte

So we put first bit of the byte as 0 as ot continue to next byte , last bits same as binary for 8

Remaining in btween is filled with 0

0 000 1000

For 128

Binary is 1000 0000

First 1 bit means > 7 bits needed . next byte says no need of extra byte

Last 7 bits from binary is taken and put to below first byte.

Second byte wll have first bit is 0 means can fit remaining bits then 1 followed by 0

10000000 01000000

Simple-9 first 4 bits say how next bits are used

Omsairam

Doc map has # hash of url and use that as the docid

LLRUN – length of gap in unary. Accumulating gap lengths, Huffman code first len of gap followed by len of gap in binary

Deletion using remerge technique -> we mark the documents as tombstone records and not use them.

Immediate merge :

In disk index:

“a”: pl “a”

“the” : pl “the”

In mem index:

“a”: pl ”a”

“the”: pl “the”

Then have an output index that will have the size disk and mem, so extra space used.

If same terms in in mem and in disk , then append to disk index from in mem index

So linear scan of both the indexes, mem used is twice.

In place merge :

In disk index:

“a”: pl “a” [has an extra space per posting list]

“the” : pl “the” [has an extra space per posting list]

In mem index:

“a”: pl ”a”

“the”: pl “the”

Complexity : Length of in mem index if in mem fits in extra space of in disk pl

Check if in mem pl <= extra space in in disk pl, then directly put to in disk

If in mem pl > extra space in disk pl, use a new pl that copies the 2 pl from in mem and in disk. And replace with the in disk pl. so time complexity : len of in disk + len of in mem pl

Extra space needed is a complete new pl.

Log merging

In mem

In disk

(folder structures where we store disk based indexes)

Gen1

Gen2

Gen3

When In me. Fills, I fill into A. ind, then write to B. ind, merge A and B to form A. ind, then B, ind comes, then merge if more than 1 some gen index, merge to create the next gen index.

Binary means 0 1 where I store in gen 1 slot for 0 slot and then gen 1 slot in 1 slot, if both filled , merge them and pull to next gen2 slot in 0 slot of gen2

* Suppose we can index 200,000 documents before we run out of memory and need to merge with a disk based index that uses logarithmic merging.
* We want to index a billion pages, using 4 machines each responsible for having the index for 1/4 of the documents.
* Suppose we can index 300,000 documents before we run out of memory and need to merge with a disk based index that uses logarithmic merging.
* We want to index a two billion pages, using 3 machines each responsible for having the index for 1/3 of the documents.

total documents-  2B

Machines count- 4

N= 1B/4= 250M = 250000000

M=200000

1B / 4 = 250 M = 250000000

N/M = 250000000/200000

2^t \*2 = 2500

t log(1275) = 11

A) latest generation is 10

B) 9 merges required

total documents-  3B

Machines count- 3

N= 2B/3= 1B

M=300000

2B / 3 = 2/3 B

2^t \*2 = 2500

t log(1275) = 11

A) latest generation is 10

B) 9 merges required

N.log(N/M)

 In m/y limit = 200000, total docs = 1000000000, docs per machine = 250000000

gen t = 2^t \* 200000 , t = log\_2 (250000000 / 200000) = 10

Largest generation = 10, Most generations that will need to be merged = 9

(**Edited: 2022-04-20**)

1 billion/4 machines = 250 million

250 million/200000 documents = 1250

log(1250) = 10.28 ~ 10 generations

The largest generation is 10.

Max number of generations merged is 9

Limit = 300000,

tot doc = 2000000000

total m/c = 3

docs per m/c = 2000000000/3

gen t = 2^t \* 300000, t = log\_2 (2000000000/(3\*300000)) = 11

most gens merged = 11-1 =10

6464917119 – grant call-based number. Grants confirm 15k.

NC717

Omsairam

BM25F only change in BM25 where we check only titles

Each title acts as a document

S= Sample space = collection of our events, is a set

P = map from set of events =[0,1], probability distribution

Sum P(s) where s E S

Random variable : R = function from sample space into real value X : S-> IR

E(X)= sum P(s)X(s) where s E S

p(r∣D,Q) prob of document D given the document in the query Q

omsairam

pick terms from corpus at random and add to query : called query expansion : Kullback\_Leibler Divergence. Check for p(q|d) approach

DFR use:

autocomplete for query , see previous queries used for the IR ranking system of queries.

norm ftd = ftd \* log (1 + lavg/ld)

# This is a sample Python script.  
  
# Press ⌃R to execute it or replace it with your code.  
# Press Double ⇧ to search everywhere for classes, files, tool windows, actions, and settings.  
  
import math  
  
words = {  
 "Federal": 75,  
 "Income": 701,  
 "Tax": 354,  
 "Return": 707  
}  
  
corpus = 350000  
avg\_len\_queries = 4  
  
  
def score\_DFR(new\_query, given\_query):  
 score = 0  
 for term in given\_query:  
 ftd = new\_query.count(term)  
 norm\_ftd = ftd \* math.log(1 + avg\_len\_queries / len(new\_query), 2)  
 num = math.log((1 + words[term] / corpus), 2) + norm\_ftd \* math.log((1 + corpus / words[term]), 2)  
 den = (norm\_ftd + 1)  
 score += num / den  
 print(score)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 given\_query = ["Federal", "Income"]  
  
 new\_query = ["Federal", "Income", "Tax"]  
 score\_DFR(new\_query, given\_query)  
  
 new\_query = ["Federal", "Income", "Tax", "Return"]  
 score\_DFR(new\_query, given\_query)

Split index in terms of term or documents partition and use intra query processing

Omsairam

Parallel processing queries

* 1. Documents partitioning

User has query -> Receptionist m/c (sends to each m/c) get results like scored documents and merges them and get top m results from them

/

\/

M1 = docid%3 ==0 . each m/c sends some top k results

M2 = docid%3 ==1 . each m/c sends some top k results

M3 = docid%3 ==2 . each m/c sends some top k results

Top M results needed, n machines, and each machine will give top k results.

∑l=0kb(n,m,l)⋅(p(n−1,m−l,k))

The first machine will give us l results, then rest of the n-1 machines will give the result the will accomplish remaining m results.

Hash document wise and partition documents

* 1. Term partitioning

Words beginning with term in different machines

For ranking : Need to do term at a time processing using accumulator termAtATime with accumulator

M1 = a-f =

M2 = g-t =

M3 = u-z =

For query = joe biden,

receiptionist request m1 for biden and m2 for joe and gets result and merges them to get top m results

if multiple terms in same m/c sends both to same m/c

final : doc partition plus replication of documents partitions

fault tolerance also achieved and query time processing improved

MAP REDUCE

Type theory

0th order object is simple 1 object

Map that links 0th order object-> 0th order object is called first order object

Ex:

Map function f

F: x-> x2

L = <1, 2, 3>

Map(f,l) = <1, 4, 9>

Reduce function g

L=<1, 2, 3>

Reduce(+, l) = 1 + reduce(+, <2, 3>)

= 1 + 2 + reduce(+, <3>)

= 1 + 2 + 3 = 6

Type 1 function of arity 2 = so a second order function

Args: Takes first order function of arity 2 and a list

Number of arguments a function take is its arity

K=doc id

V =document

Map(k,v)

Ex: bob-> 1, 1,1, 1,1 number of times bob seen in document v

Reduce(k, <v1, v2, v3>)

Where k doc id

<v1, v2..> = <1,1 ,1, 1, 1>

Finally counting the total times bob present in document v

Some output from map goes to m/c or reducer

m/c A say computes key bob and gives 2000

m/c B say computes key bob and gives 50

combiner will do A and B bob and give 2000+50 , running its worn reducer

and give 2050

combiner is on a map m/c that sends its result to reducer m/c

ensure fault tolerance how?

map -> distributed file system (checks all files present)-> reducer if fails -> reads from distributed file systema and does reduce again.

Ex: Hadoop example of distributed file system, zookeeper

omsairam

How does website know where I came from

telnet : DNS lookup -> gets ip address -> gets socket connection at that ip address -> tells from host id where it came from

browser then refers then web page comes back

telnet

referrer tells where I clicked from , browser sences this

* + given a web graph

links between pages 1 2 3

1-

-2 -3

1 , 2 and 3 are linked

1 goes to 2 and 3

2 goes to 1 only

3 goes to 2 only

Matrix creation

0 ½ ½

1 0 0

0 1 0

Power method doesn’t work if a page leads to no where.

Image page is a black hole page sinks.

Omsairam

Xi(0) = col matrix with 1/num of pages with row number of pages

Same matrix for yi(0) but also multiply the matrix with Lr for yi(0)

Xi(1)=A. xi(0)

Yi(1)=H.yi(0)

Update adjacency list and dangling node etc