1. Given: 300 documents Relevant (9) = §4,8,04 = 3

System-1: NN ABN Rulsyst = JA, By = 2

System -2: NANNB Relsgs2 = {A,B} = 2

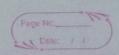
: MAP $_{3452} = 1 (1+2) = 1 \times 9 = 9$ = 0.45

System -3: ANNNN Relayes = GAG =1

.. MAPayes = 1 (1) = 1

.: MAPayes > MAPayes > MAPayes

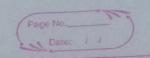
As system 3 has the highest Maximum A
posteriosi (MAP), it should be preferred.
MAP is a useful estimate which gives us
a point estimate of the guery results in
these systems.



2. Given: $t_1 = VNIT$, $t_2 = topper$, $t_3 = Medal$ $q_0 = (1,1,0)$ Relevant $\begin{cases} d_1 = (3,2,1) \\ d_2 = (3,2,1) \end{cases}$ Non $\begin{cases} d_2 = (6,0,0) \\ \end{cases}$ X=B=8=1 According to Rocchio's update rule, ->

=) 2m = xq' + B1 \(\frac{2}{2}\) \(\frac{1}{2}\) \(\fr - 1×1 [(6,0,0)] 2 (1,1,0) + (2,2,1) - (6,0,0) ·: qm = (0,3,1) All the documents given will have a lot of occurences of the term "VNIT", hence its weight will be irrelevant. The weights of The initial query q. (1,1,0) didn't consider the weight of terms "Medal" and hence was wrong. The new vector now considers the weight of "medal" and "topper" which also satisfies relevant documents. 3. Given: Doc1: Phase ring person happy person
Doc2: Dog pet happy run jump
Doc3: Cat pull pet person happy
Doc4: life Imile run happy
Doc5: life laugh walk run

(a)	Telm D	occum en	+ Free	mency!			
-4		D,	Di	123	D4	Ds	
-	cat	0	0	1	0	0	
F	dog	0		0	0	0	
4	happy			1	1	0	
K	jump	6		0	0	0	
	laugh	0	0	0	0		
1	life	0	0	0	1	1	
F	person	1	0	1	0	0	
1	pet	0		13 1	0	0	
	phone	1	0	0	.0	0	
	pure	0	0	1	0	0	
	ring	1	0	0	0	0	
	Tun	0	1914	0	1	1	
	Smile	0	0	0	(0	
	walk	0	0 1	0 1	0 1		
(b) Inverted index for ranked retrieval							
(9)	triving	mack of	or your	ed rel	rieval		
$cat \rightarrow (d3) =$							
dog -> [d2] +							
happy -> [di]-[d2]-[d3]-[d4]-1							
jump -> [d2)-1							
laugh -> (ds) =							
life -> [d4) [d5] =							
pelson -> [d] - [d3] -1							
$dog \rightarrow [dz] = 1$ $happy \rightarrow [dz] - [dz] - [dz] - [dz] - 1$ $jump \rightarrow [dz] = 1$ $laugh \rightarrow [ds] = 1$ $life \rightarrow [dz] - [ds] = 1$ $pelson \rightarrow [dz] - [ds] = 1$ $phone \rightarrow [dz] - [ds] = 1$ $phone \rightarrow [dz] - [ds] = 1$							
phone - tait =							
ring -> [di)-1							
run > (d2) (d4) (d5) 1							
smile -) (d4) =							
W	alk ->	(d5) = =					
		र इ					



When handling a query containing multiple terms, some terms should be given more impostance compaled others and the scores of all terms of the query along with the scores of all terms of the query along with the scores of documents contained these terms should be computed accordingly.

· Computing Coline Scoles:

For each query term, the weight of that telm in the query is calculated and its posting list is fetched. Then for every cole of the document, term frequency pair, scole of the document is updated to the previous scole plus the product of weight of term in query and the (wf) of term in the document. The topk such documents are returned to the user.

Fast cosine Scoles.

If weight of all the terms in a quely are considered equal, then instead of taking weights of term, it can be replaced with I tence, only (wf) of the term in a document is added to the scoles of a document for each term.

Fol top k document retrieval max-heap can be used. Construction of heap will take O(N) and top k entries can be read in O(logN). Thus, complexity becomes O(klogN). For further optimization we can avoid computing scores for all N documents and

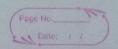
consider inexact k documents only, i.e. computing only for those k documents which are closes to k-best. Also, we can use shdex Elimination, where we consider only those documents which have at least one query term

A set of r documents are precomputed by every term in dictionaly. These r docs have highest term freq values. These r docs ret is the champions list for the given terms. Now, for every given query, union of champion list for all terms in query can be considered for cosine computation which improves efficiency to a good scale.

· Cluster Pruning:

Document vectors are dustered as a preprocessing step. Then at query time only that
duster is consider which is closer to the
query a handowly some leaders (docs) are
selected and at query time, the cluster
(leader + followers) closes to the query is
considered for top k document retrieval

5. Basic principle of KMP algorithm is minimized ing the manual matching which is done in the brute force method, having the worst complexity of O Coattern size * text size. This is done by keeping track of the



characters that are gonna match anyway once a nice match is found, or in other words we precompute the length of the longest prefix which is also a suffix at every index of the pattern.

Ex: for the patteen "AAAA"

Lps[] = [0,1,2,3]

for the pattern "ABCDE"

[ps[] = [0,0,0,0,0]

Now when we slide the pattern one by one and compare the character, we use the value from lps[] to decide the next character of the pattern to be matched Therefore, not matching a character that we know will anway match.

Exitxt = "AAAAA BAAABA", pat = "AAAA"

Lps [] = {0,1,2,3], i = txt pointer

j = pattern pointer

1=0 j=0
AAAAABAABA AAAABA
AAAA
AAAA

1 2 j = 3 AAAA BAABA AAAA BAABA AAAA BAAABA AAAABA

124 j=4

AAAABA J checking for only the

AAAAB last character of the pattern