IRS

KP BP NF

Types of query

1) Keyword Query :- Order not important
2) Phrase Query :- Order of text is imp. Normally enclosed with double quote
3) Boolean Query :- For Text can be found with the help of operator
4) Proximity Query :- For eg. keyword 'Machine learning' is to be found
   If I use a proximity query with value 3
   then we can allow max 3 words to accomodate
   betn Machine & learning
5) NL Query :- For the one who have not enough knowledge of writing the query. Naive user. eg. Chat bots
6) Full document Query :- Usually provided URL's

⟹ Search Engines

Semantic Search Engine (IIT) :-

⟹ Text processing

for e.g. Corpus — e.g. This is the second week of the course

1) Tokenization is the process to split every word from the sentence & each word is taken in context of sentence.
   { This , is , the , second , week , of , the , course }
2) Stop Word Removal :- Words such as and, not, is, are, my, etc are present in most of the webpages, so to avoid searching for all majority of documents, these words are not taken into consideration. This does keywords just help to articulate the sentence but does not represent any content of the document.
   { second , week , course }
3) Stemming

→ It reduces the word to its root word after removing suffixes or prefixes.

e.g. 'Chocolates', 'choco', 'chocolatey' ⟶ Chocolate (Root word)
have ~~different~~ meaning

→ Sometimes root word ~~meaning~~ ~~does not exist~~ in dictionary, so to avoid blindly cutting of suffix another process is there which is known as Lemmetization.

{ second, week, course }

4) Case Conversion

→ It converts the text into same case

{second, week, course}

Eg. Doc 1 :- He is corona ~~+~~ positive

1) Tokenization :- { He, is, corona, +ve }
2) Stop Word :- { Corona, +ve }
3) ~~{ to~~ Stemming :- { Corana, +ve }
4) Case Conversion :- { corona, positive }

Doc 2 :- If positive, than go for testing

1) { If, +ve, then, go, for, testing }
2) { +ve, go, testing }
3) { positive, go, test }
4) { ~~corona~~ positive, go, test }

Doc 3 :- Be +ve through vaccination not corona

1)
2) { Be, +ve, vaccination, corona } → Not a stop word
3) { +ve, Be, Vaccine, corona }
4) { +ve, be, vaccine, corona }

Final List of acc. to Alphabetical order : {be, corona, go, +ve, test, vaccine}

These are called Terms. The list of terms is vocabulary.

Corpus → Preprocess → Vocabulary

Bag of words is represented in form of boolean

|V| = {be, corona, go, positive, test, vaccine}
(Vector)

'be' not present after preprocessing in Doc 1

Doc 1 :- <0, 1, 0, 1, 0, 0>
('corona' present)

Doc 2 :- <0, 0, 1, 1, 1, 0>

Doc 3 :- <1, 1, 0, 1, 0, 1>

These 3 can be combined to form a 3×6 Matrix.

Lets say Query is 'Corona Positive' : <0,1,0,1,0,0>

⇒ Limitations of Boolean Approach

1) Count is not recorded
2) Order is not recorded.
3) Rank to the documents will not be given.

O - order
C - count
R - ranking

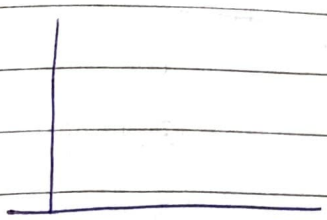
Now to find relevant documents find do AND

✓ DOC 1 AND QUERY VECTOR
× DOC 2 AND " "
✓ DOC 3 " "

Even though doc 2 contains similar statement this will not be the part of displayed doc.

→ If 2 vectors are ~~similar~~ nearer then it represents that they are somewhat similar.

Eg



A(2, 3)
B(5, 1)
C(4, 0)

K points → K documents
n dimensional vectors or n vectors → n diff vocab terms

$d(AB) = \sqrt{13}$
$d(AC) = \sqrt{13}$
$d(BC) = \sqrt{2}$

→ Now lets say A & B are documents & C is a query so the documents which are nearer to query will be displayed first.

d_1 <2, 3, 1, 0, 5, 0, 0, 3> → TF form

d_2 <1, 0, 0, 1, 0, 0> → Both Boolean & TF form

Normalized TF – It is used to avoid large numbers

Approach
1) Term freq

$$TF_{ij} = \frac{f_{ij} \text{ (Frequency)}}{\max\{f_{1j}, f_{2j}, \ldots f_{|V|j}\}}$$

f_{ij} → Count of i^{th} term in document 'j'.

Ex. d_1 → T1 = 5, T2 = 3, T3 = 10

$TF(t_1, d_1) = 5$ → we want to find

∴ Normalized TF $= \frac{5}{\max(5, 3, 10)} = \frac{5}{10}$

2) Normalized $TF(t_1, d_1) = 5 /$ No. of terms in doc

Need of Normalized TF :- As documents can have ~~different~~ length it is possible that a term would appear more freq in longer than shorter ones, because of this it will seem like a term is more imp to longer doc than shorter one.

$\left(= \frac{5}{5+3+10} = \frac{5}{18}\right)$

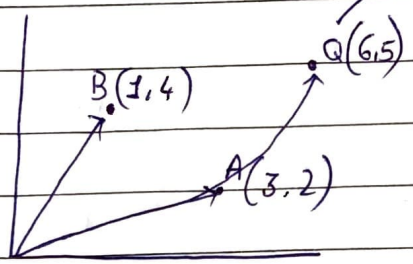
→ Ranking is possible, Count is recorded in TF representation.

→ If stop word is also coming more freq then it will also not be considered as discriminative.

document so find every pair of vector distance

This is ~~query~~ ~~to be found~~

Eg.



Two docs A & B

(3,2) represents count of words (word → t_1 appears 3 time in doc A, t_2 appears 2 time in doc A, t_1 appears 1 time in doc B, t_2 " 4 " " " ")

Soln

$QA = 3\sqrt{2} = \sqrt{18}$

$QB = \sqrt{26}$

$AB = \sqrt{8}$

We can infer that A & B are closest documents but we can see too that vector Q is being coincided with A, so Q & A ~~not~~ must be similar more than A & B.

→ So disadv of Euclidean dis is it does not consider vector direction & angle bet^n vectors

→ So to overcome this dis adv., cosine similarity is used.

~~→~~ Soln using cosine similarity

1) Length of a vector

$|A| = \sqrt{3^2 + 2^2} = \sqrt{13}$

$|B| = \sqrt{17}$

$|Q| = \sqrt{61}$

$x(1,0)$

$y(0,1)$

$|x| = 1$, $|y| = 1$, $dot(x,y) = 0$ (As x & y are 1 the dot product is 0.

~~This one~~ or

→ The 2 vectors should be orthogonal if their dot product is 0

→ ~~Orthogo~~ 2 vectors are orthonormal is they are both orthogonal & normal.

$$\text{cosine}(A,B) = \text{dot}(A,B) / ||A||B||$$

$$\text{Cos}(A,Q) = \frac{3\times6 + 2\times5}{\sqrt{13}\sqrt{61}} = \frac{28}{\sqrt{13}\sqrt{61}} = 0.99$$

$$\text{Cos}(B,Q) = \frac{26}{\sqrt{17}\sqrt{61}} = 0.81$$

$$\cos(A,B) = \frac{11}{\sqrt{13}\sqrt{17}} = 0.74$$

→ As cosine similarity is highest for A & Q so they are closest to each other.

Eg. There are 10 documents in the corpus.

1) A term t_1 is appearing in almost all ~~doc~~ documents.
2) A term t_2 is " " only one ".

Query contains the term t_1.
What will be the set of documents as a response from the search engine? ~~An~~

Ans:- 7 documents will be appeared.

If eg. 10000 docs are there & t_1 is appearing in 7000 docs then it will be difficult to filter out which docs to take. We can say that T_1 is not discriminative.

↳ We will resolve this query using IDF (Inverse document finder)

Date:

For eg. T2 is present in very few docs in the corpus so we can say T2 is discriminative

~~If~~

⇒ IDF technique

$$idf_i = \log \frac{N}{df_i}$$

N → Total number of documents

df_i → No. of documents in which t_i is present

Eg. 1000 docs, ~~tt~~

t1 : 200 → (t1 present in 200 docs)

t2 : 50

t3 : 700

Soln

$idf(t1) = \log(1000/200) = \log 5 = 0.6989$

$idf(t2) = \log(1000/50) = \log 20 = 1.3010$

$idf(t3) = \log(1000/700) = \log \frac{10}{7} = 0.1549$

~~If we~~ We should not use these type of keywords in search as it will display more than required documents. These type of keywords are not important or discriminative.

Eg.

	Book	Car	Drive	Go	Slow	Vaccine	Yesterday
D1	1	1	0	0	0	0	1
D2	0	1	1	0	1	0	0
D3	0	0	1	1	0	1	0
Q	0	1	1	0	0	0	0

D1 : I booked a ~~ye~~ car yesterday
D2 : Drive car slowly
D 3 : Vaccination drive is going on

Q : Car Driving

Soln N = 3 (Number of docs)

Find idf of vocab terms

$idf(book) = \log\left(\frac{3}{1}\right)$
$idf(car) = \log\left(\frac{3}{2}\right)$

:

$idf(yest) = \log\left(\frac{3}{1}\right)$

$tf(book, D1) * idf(book) = 1 \times \log 3 = 0.4771$

→ ~~Query Vect~~ From these values construct TFIDF matrix

	Book	Car	Drive	go	Slow	vaccine	yest
D_1	$\log 3$	$1 \log\frac{3}{2}$	0	0	0	0	$\log 3$
D_2	0	$\log\frac{3}{2}$	$\log\frac{3}{2}$	0	$\log\frac{3}{1}$	0	0
D_3	0	0	$\log\frac{3}{2}$	$\log\frac{3}{1}$	0	$\log\frac{3}{1}$	0
Q	0	$\log\frac{3}{2}$	$\log\frac{3}{2}$	0	0	0	0

TFIDF matrix.

→ Now if you want to find which doc is similar then do cosine similarity in TFIDF matrix

→ Disadvantage of this approach is that if for eg. term 'car' is present in all docs then tfi value of car will be $\log\frac{3}{3} = 0$. But this is rare case.

→ ~~log~~ Here if user mistakenly types any word in query which is not present in any of the doc, then $\log\frac{3}{0}$

exception will arise. This issue can be resolved by laplacian correction

Q

D1 :- Book story write book
D2 Book ticket train
D3 write read write write
Q:- D4 read write

Find
1) List of Vocab terms
2) Boolean Model
3) TF model
4) TFIDF model
5) Use boolean matching & cosine similarity on the corpus.

Soln
1) { Book , story , write , ticket , read , train }

2) Book read story ticket train write

D1 < 1 0 1 0 0 1 >
D2 < 1 0 0 1 1 0 >
D3 < 0 1 0 0 0 1 >
Q:- D4 < 0 1 0 0 0 1 >

⇒ Evaluation measures

Same things we did in ML.

⇒ Query :-

Search Engine

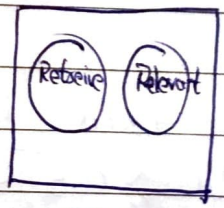
- Our goal :- list of webpages relevant to user's query.

But obstructions such as

1) All these webpages are really relevant to user's query?
→ No all may not be relevant.
→ The documents which search engine thinks relevant are called retrieved documents.
→ All the retrieved webpages may not be relevant.

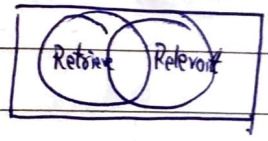
2) ~~It is~~ Is it possible to retrieve all the relevant documents?

Scenario 1:

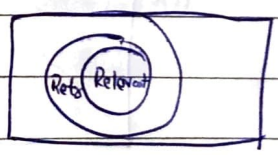


Rare case
If it is then search engine is bad.

Scenario 2:



Scenario 3:



As a user, what is your objective?
1) To get only relevant webpages as response
2) Fast retrieval
3) Sequence of webpages.

→ Also as a user, our intention is to find the relevant results as the very first set of responses.

→ Precision of IR system takes care that how many relevant documents are there at the beginning out of retrieved doc.

1st
2nd
3rd
:

→ Precision = $\frac{\text{Proportion of relevant doc}}{\text{Proportion of retrieved doc.}}$ (Just basic formula not the correct one)

* Recall

→ Out of all the relevant documents as per the ground truth, how many were actually relevant.

Q

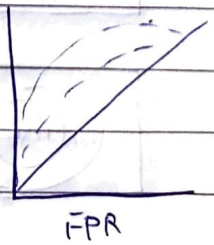
100 TP	50 FN
150 FP	200 TN

ROC ?

→ ROC can be found by threshold

if threshold $\geq 0.5 \longrightarrow$ Positive
" $< 0.5 \longrightarrow$ -ve

General value, we can take any



we want more TPR
less FPR

→ By increasing or reducing threshold we can control any one of FP or FN, but reducing both simultaneously is bit tough

Eg.

Ground truth doc = 5

+ ⟶ Relevant doc
− ⟶ Irrelevant doc.

Rank	Status	P(i)	R(i)
1	+	1	0.2
2	+	1	0.4
3	−	0.66	0.4
4	+	0.75	0.6
5	−	0.6	0.6
6	−	0.5	0.6
7	+	0.57	0.8
8	−	0.5	0.8
9	−	0.44	0.8
10	−	0.4	0.8

→ Precision (i) = $\frac{\#\text{ Rel. doc. retrieved so far}}{\#\text{ No. of doc retrieved so far}}$

$P(1) = \frac{1}{1}$ (For position 1 we will see till 1st doc only)

$= 1$

$P(2) = \frac{2}{2}$ $P(4) = \frac{3}{4}$ $P(6) = \frac{3}{6}$ $P(8) = \frac{4}{8}$

$P(3) = \frac{2}{3}$ $P(5) = \frac{3}{5}$ $P(7) = \frac{4}{7}$ $P(9) = \frac{4}{9}$

$P(10) = \frac{4}{10}$

→ Recall (i) = $\frac{\#\text{ No. of rel. doc retrieved so far}}{\#\text{ No. of rel. doc as per ground truth}}$

↳ will be given in question.

Recall (1) = $\frac{1}{5}$

→ Avg. Precision = Avg of relevant ~~results~~ doc

$$= \frac{100\% + 100\% + 75\% + 57\%}{4}$$

$$= 83\%$$

→ Rank Precision :- Precision at particular rank position.
for eg. RP(4) = 0.75
↳ Rank 4

→ ~~Break~~ P-R BreakEven point :- The rank position at which precision & recall is same is called P-R Breakeven point. (If that particular position is not found then consider position with similar precision & recall).

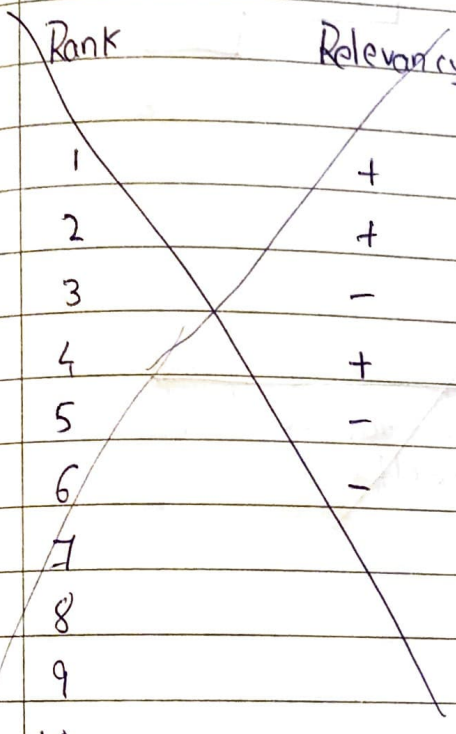
Q

Rank	Rel	P(i)	R(i)
1	+	1	0.2
2	-	0.5	0.2
3	+	0.66	0.4
4	+	0.75	0.6
5	-	0.6	0.6
6	+	0.66	0.8
7	+	0.7	1
8	-	0.625	1
9	-	0.55	1
10	-	0.5	1

Ground truth = 5

Avg. Precision = $\frac{100 + 66 + 75 + 66 + 70}{5}$

~~Ran~~ P-R Even Break point = 5 (Position)

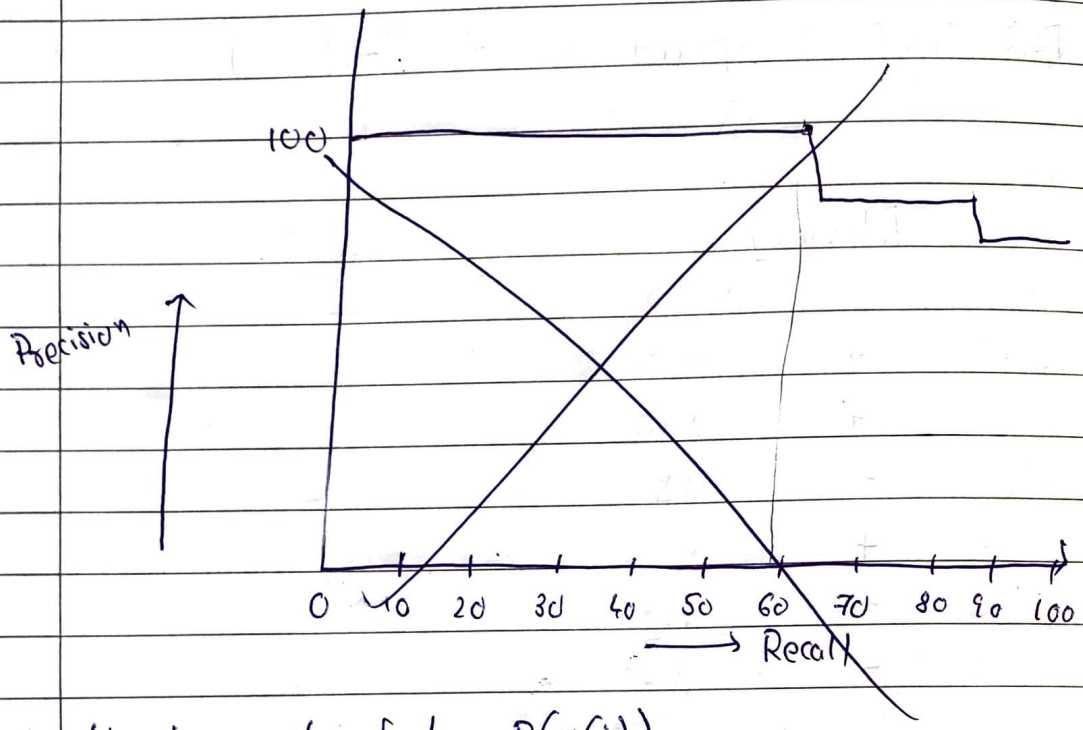


Q	Rank	Relevancy
	1	+
	2	+
	3	−
	4	+
	5	−
	6	−
	7	
	8	
	9	
	10	

Q	Rank	Relevancy	P(i)	R(i)
	1	+	$\frac{1}{1}$ = 100%	$\frac{1}{5}$ 20%
	2	+	100%	$\frac{2}{5}$ 40%
	3	+	100%	$\frac{3}{5}$ 60%
	4	−	75%	$\frac{3}{5}$ 60%
	5	+	80%	$\frac{4}{5}$ 80%
	6	−	66%	80%
	7	−	56%	80%
	8	+	63%	100%
	9	−	56%	100%
	10	−	50%	100%

→ Precision Recall curve

We have to find 11 equidistant recall points / levels



→ We have to find $P(\gamma(i))$

$\gamma(i)$	$P(\gamma(i))$	
0	0	
10	0	
20	100	($P(20) = 100$)
30	100	
40	100	
50	100	
60	100	($P(60) = 100$ or $P(60) = 75$ we have to use formula)
70	80	(Rank 5 to 10)
80	80	(Rank 5 to 10)
90	63	(It is between 80% & 100% b/w rank 5 & 8, so after rank 8 to 10 we will take max value)
100	63	

Calculate $P(\gamma(i))$ from bottom to top.

→ To plot the PR curve we use 11 standard recall levels, 0, 10, 20, ... 100.

→ We may not obtain exact values of these recall levels, therefore interpolation is used to obtain precision at such recall level.

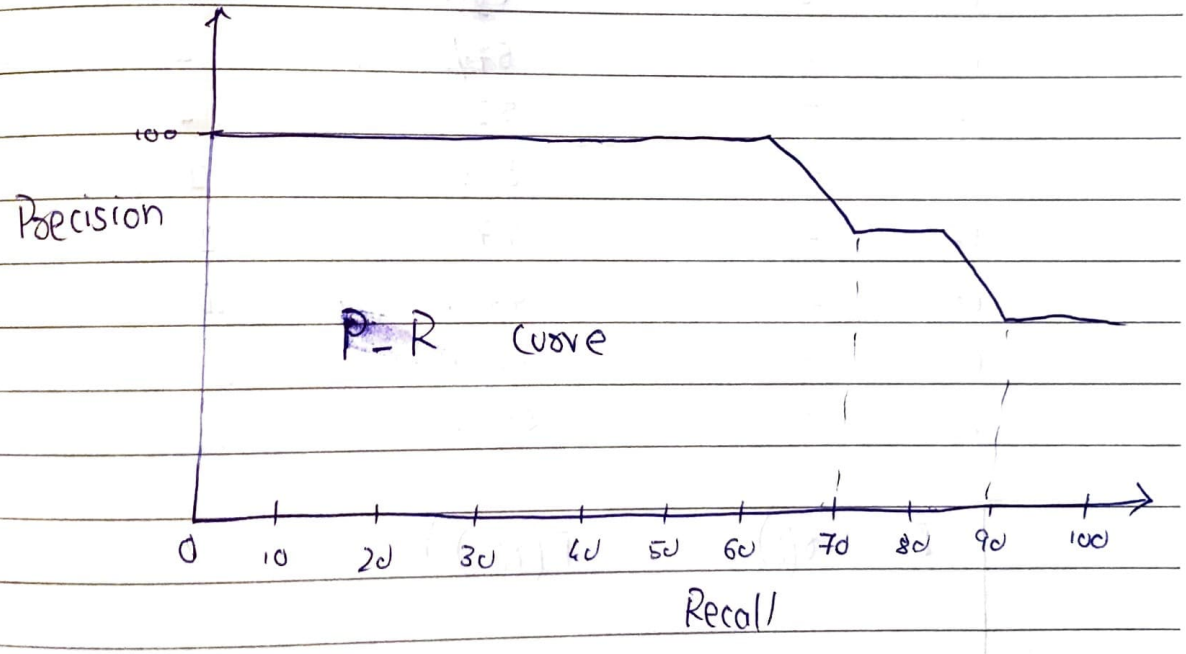
→ let $r(i)$ be a recall level, such that

$r_i \in \{0, 10, 20, \ldots 100\}$

$i \in \{0, 1, 2, \ldots 10\}$

$P(r_i)$ be the precision at recall level r_i where $P(r_i)$ is computed as.

$$P(r_i) = \max_{r_i \leq r \leq r_{10}} (P(r))$$



i	1	2	3	4	5	6	7	8	9	10
P(i)										

7/12 0.09 8/49 8

Q	Rank	Rel	Precision	Recall
	1	+	100%	0.125
	2	+	100%	0.25
	3	+	100%	0.375
	4	−	75%	0.375
Ground truth = 8	5	+	80%	0.5
	6	−	66%	0.5
	7	+	71%	0.625
	8	−	62.5%	0.625
	9	+	66%	0.75
	10	+	70%	0.875
	11	−	64%	0.875
	12	−	58%	0.875
	13	+	62%	1
	14	−	57%	1
	15	−	52	1
	16	−	50%	1
	17	−	47	1
	18	−	44%	1
	19	−	42%	1
	20	−	40%	1

γ(i)	P(γ(i))
0	100%
10	100%
20	100%
30	100%
40	80%
50	80%
60	71%
70	70%
80	70%
90	62%
100	62%