

I R S

Types of query

- 1) Keyword Query :- Order not important
- 2) Phrase Query :- Order ^{of text} is imp. Normally enclosed with double quotes
- 3) Boolean Query :- Text can be found with the help of operators
- 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 accommodate
betn Machine & learning
- 5) NL Query :- For the one who have not enough knowledge of writing
the query. Naive user. e.g. Chat bots
- 6) Full document Query :- Usually provided URL's

⇒ Search Engines

Semantic Search Engine :- ^(CSC)

⇒ 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 token 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)
 → Sometimes root word meaning does not exist in dictionary, so ^{avoid} blindly cutting off suffix another process is there which is known as Lemmatization.
 {second, week, course}

4) Case Conversion

→ It converts the text into same case.
 {second, week, course}

e.g. Doc 1 :- He is corona + positive

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

Doc 2 :- If positive, then go for testing

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

: of 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

$|V| = \{ \text{be, corona, go, positive, test, vaccine} \}$
Vector

'be' not present after preprocessing in Doc 1

Doc 1 :- $\langle 0, 1, 0, 1, 0, 0 \rangle$
↳ 'corona' present

Doc 2 :- $\langle 0, 0, 1, 1, 1, 0 \rangle$

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

Doc 3 :- $\langle 1, 1, 0, 1, 0, 1 \rangle$

lets say

Query is 'Corona Positive'
 $\langle 0, 1, 0, 1, 0, 0 \rangle$

⇒ Limitations of Boolean Approach

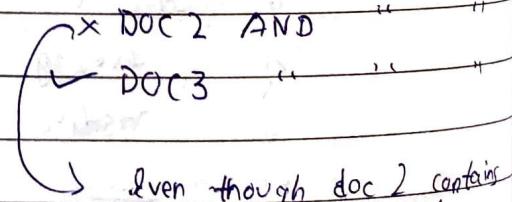
- 1) Count is not recorded
- 2) Order is not recorded.
- 3) Rank to the documents will not be given.

Now to find relevant documents
find doc AND

✓ Doc 1 AND QUERY VECTOR

✗ Doc 2 AND " "

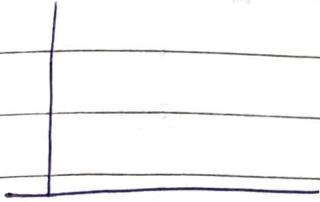
✗ Doc 3 " "



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

→ If 2 vectors are ~~similar~~^{nearby} then it represents that they are somewhat similar.

Eg.



$$A(2, 3)$$

$$B(5, 1)$$

$$C(4, 0)$$

k points $\rightarrow k$ documents

n dimensional vectors $\rightarrow n$ diff vocab terms
n vectors

$$d(A, B) = \sqrt{13}$$

$$d(A, C) = \sqrt{13}$$

$$d(B, C) = \sqrt{2}$$

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

$$d_1 <2, 3, 1, 0, 5, 0, 0, 3> \rightarrow \text{TF form}$$

$$d_2 <1, 0, 0, 1, 0, 0> \rightarrow \text{Both Boolean \& TF form}$$

Normalized TF - It is used to avoid large numbers,

$$\therefore d_1 \rightarrow T_1 = 5, T_2 = 3 \\ T_3 = 10$$

Approach
i) Term freq

$$TF_{ij} \leftarrow \frac{f_{ij}}{\text{Frequency}}$$

$$\max\{f_{1j}, f_{2j}, \dots, f_{nj}\}$$

Count of i^{th} term in document j

$$TF(t_1, d_1) = \frac{5}{10} \rightarrow \text{We want to find}$$

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

ii) Normalized TF(t_1, d_1) = $5 / \text{No. of terms in doc}$

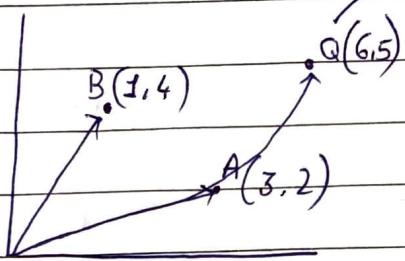
Need of Normalized TF:- As documents can have different length it is possible that a term would appear more freq in longer than shorter ones, because of this it will seem like a term is more imp to longer doc than shorter one.

$$= \frac{5}{5+3+10} = \frac{5}{18}$$

→ 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.

Q.



This is document so find
pair of vectors to be found

Two docs A & B

(3, 2) represents count of words
(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 must be similar more than
A & B.

- So disadv of Euclidean dis is if it does not consider
vector direction & angle betⁿ vectors
- So to overcome this adv., cosine similarity is used.

→ Solⁿ using cosine similarity

i) Length of a vector

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

$$|B| = \sqrt{17}$$

$$|Q| = \sqrt{61}$$

$$x(1, 0)$$

$$y(0, 1)$$

$$|x| = 1, |y| = 1, \text{dot}(x, y) = 0 \quad (\text{As } x \& y \text{ are } \perp \text{ the dot product is } 0.)$$

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

→ ~~Ques.~~ If 2 vectors are orthogonal & normal, are they both orthogonal & normal.

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

$$\text{Cos}(A, Q) = \frac{3 \times 6 + 2 \times 5}{\sqrt{13} \sqrt{61}} = \frac{28}{\sqrt{13} \sqrt{61}} = 0.99$$

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

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

→ 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 docs.
- 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? Ans.

Ans.: 7 documents will be appeared.

→ If e.g. 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 index)

For eg. T_2 is present in very few docs in the corpus
 so we can say T_2 is discriminative.

\Rightarrow IDF
 \Rightarrow IDF technique

$$idf_i = \log \frac{N}{df_i} \quad \begin{matrix} \text{Total number of documents} \\ \downarrow \end{matrix}$$

\downarrow No. of documents in which f_i is present

e.g. 1000 docs, f_i .

$t_1 : 200 \rightarrow (t_1 \text{ present in } 200 \text{ docs})$

$t_2 : 50$

$t_3 : 700$

$$\text{soln} \quad idf(t_1) = \log \left(\frac{1000}{200} \right) = \log 5 = 0.6989$$

$$idf(t_2) = \log \left(\frac{1000}{50} \right) = \log 20 = 1.3010$$

$$idf(t_3) = \log \left(\frac{1000}{700} \right) = \log \frac{10}{7} = 0.1549 \quad \leftarrow$$

~~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.

e.g. Book Car Drive Go Slow Vaccine Yesterday

	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

	Book	Car	Drive	Go	Slow	Vaccine	Yesterday
Q	0	1	1	0	0	0	0

D1: I booked a ~~taxi~~ car yesterday

D2: Drove car slowly

B3: Vaccination drive is going on

Q: Car Driving

Solⁿ

N=3 (Number of docs)

Find idf of vocab terms

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

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

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

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

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

	Book	Car	Drive	go	Slow	vaccine	yest
D1	$\log 3$	$\log \frac{3}{2}$	0	0	0	0	$\log 3$
D2	0	$\log \frac{3}{2}$	$\log \frac{3}{2}$	0	$\log \frac{3}{1}$	0	0
B3	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.

→ 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

- i) {Book, story, write, ticket, read, train}
- ii) 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

Some things we did in ML.

⇒ Query :-

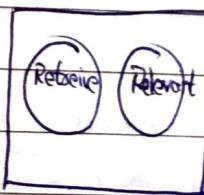
Search Engine

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

But obstructions such as

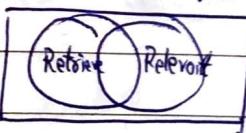
- 1) All these webpages are really relevant to user's query?
 → Not all may not be relevant.
 → The documents which search engine thinks relevant are called retrieved documents.
 → All the retrieved webpages may not be relevant.
- 2) ~~Is it~~ Is it possible to retrieve all the relevant documents?

Scenario 1 :

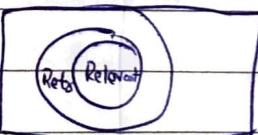


Rare Case
If it is then search engine is bad.

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 documents.
- 1st
2nd
3rd*

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

* Recall

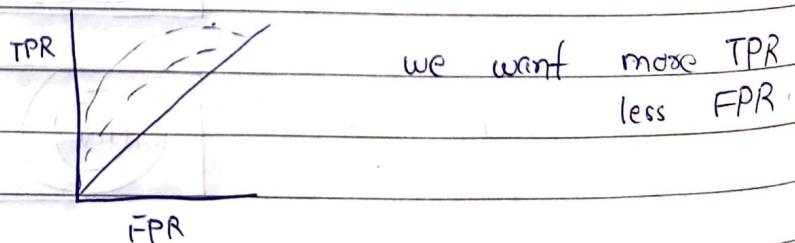
- Out of all the relevant documents as per the ground truth, how many were actually relevant.

Q

100	50
TP	FN
150	200
FP	TN

ROC ?

- ROC can be found by threshold if threshold $\geq 0.5 \rightarrow$ Positive ^{General value, we can take any}
- " $< 0.5 \rightarrow -ve$.



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

Fig.

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

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

$$= 1$$

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

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

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

→ 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.

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

→ 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

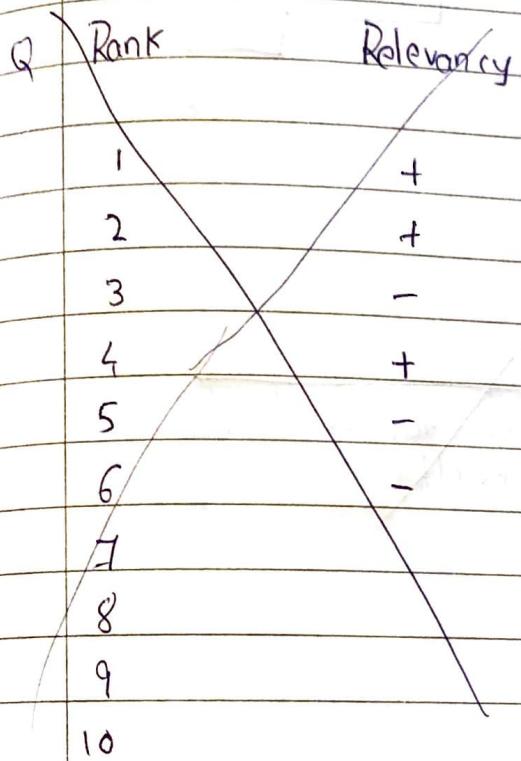
→ ~~P-R~~ BreakEven point :- The rank position at which precision & recall is same is called P-R BreakEven point. (If that particular position is not found then consider position with similar precision & recall).

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

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

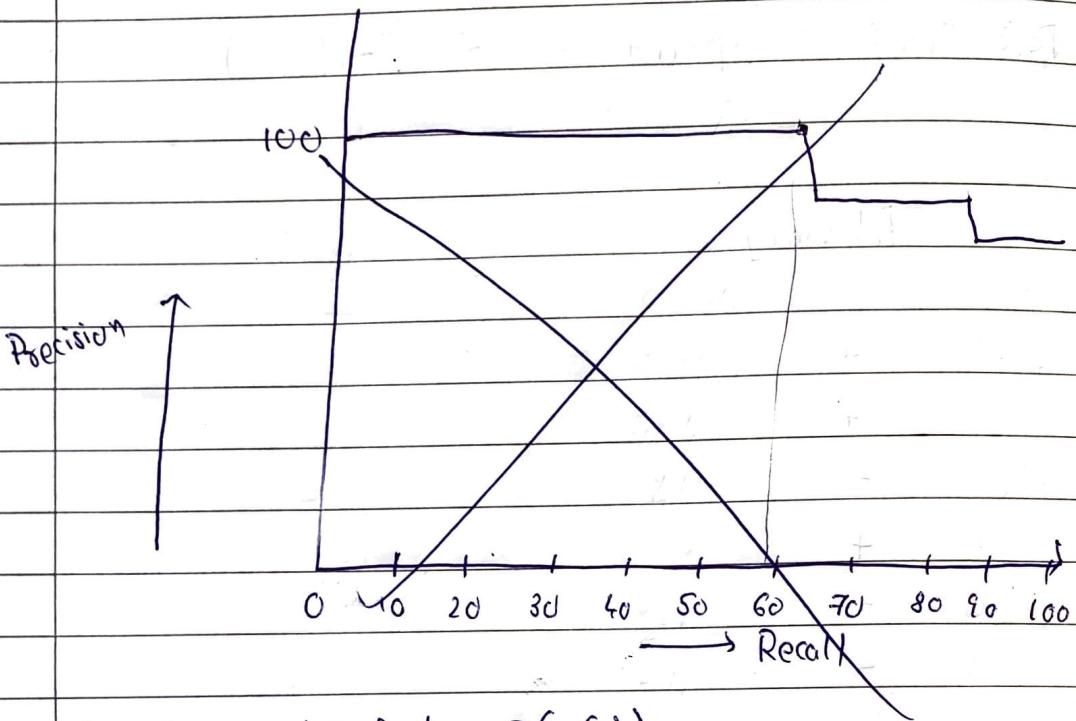
Re-R P-R Even Break point = 5 (Position)



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 $P(\delta(i))$

$\delta(i)$ $P(\delta(i))$

0 0

10 0

20 100% ($P(20) = 100$)

30 100

40 100

50 100

60 100

70 80

80 80 (Rank 5 to 10)

90 63 (It is between 80% & 100%. b/w rank 8 & 9, so after rank 8 to 10 we will find more value)

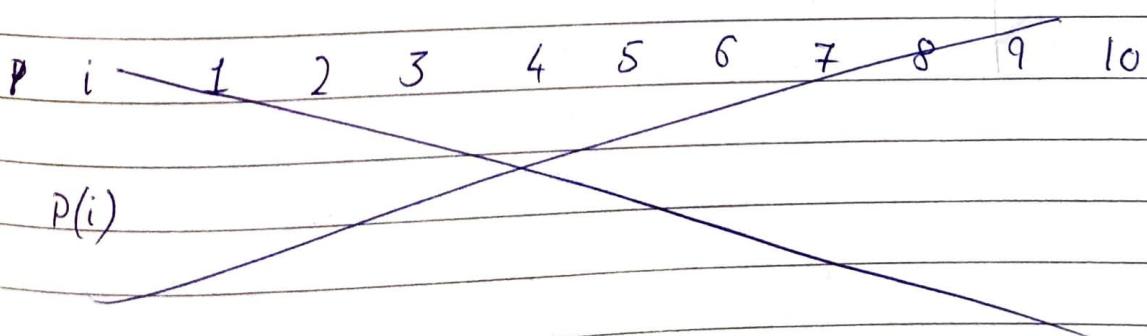
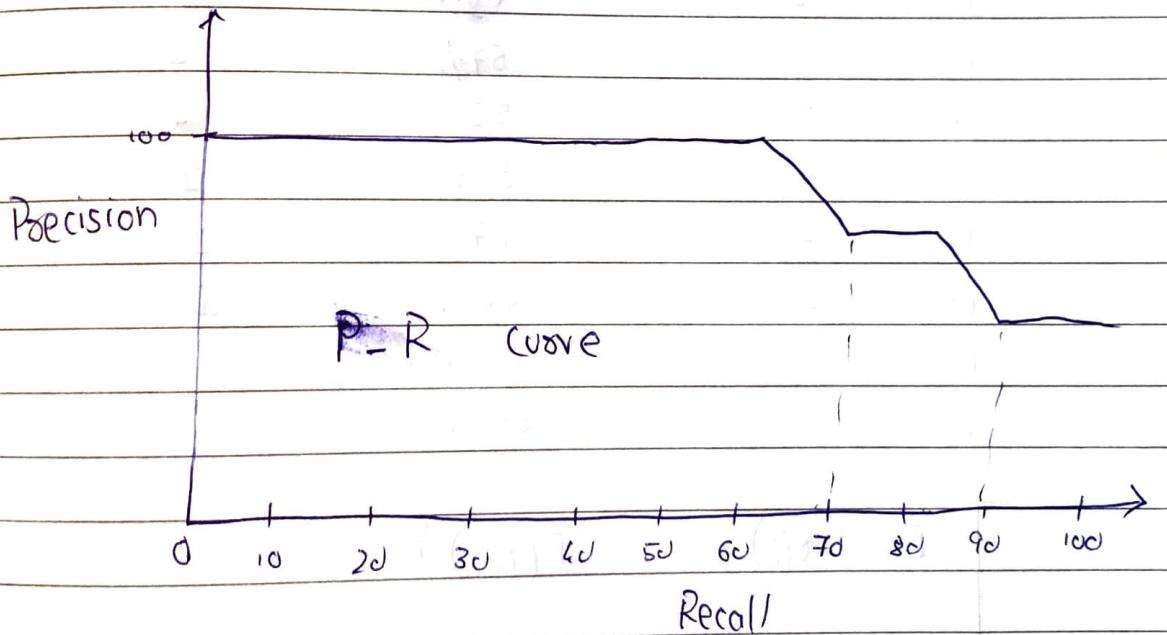
100 63

Calculate $P(\delta(i))$ 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 levels.
- Let $\delta(i)$ be a recall level such that
 $\delta_i \in \{0, 10, 20, \dots, 100\}$
 $i \in \{0, 1, 2, \dots, 10\}$

$P(\delta_i)$ be the precision at recall level δ_i , where $P(\delta_i)$ is computed as.

$$P(\delta_i) = \max_{\delta_i \leq \delta \leq \delta_{10}} (P(\delta))$$



Q	Rank	Rel	Precision	Recall
Ground				
Truth = 8	1	+	100%	0.125
2	+	100%	0.25	
3	+	100%	0.375	
4	-	75%	0.875	
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	-	56%	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	

$$\boxed{P(\delta(i))} \quad P(\delta(i))$$

0	100%
10	100%
20	100%
30	100%
40	80%
50	80%
60	71%
70	70%
80	70%
90	60%
100	60%

