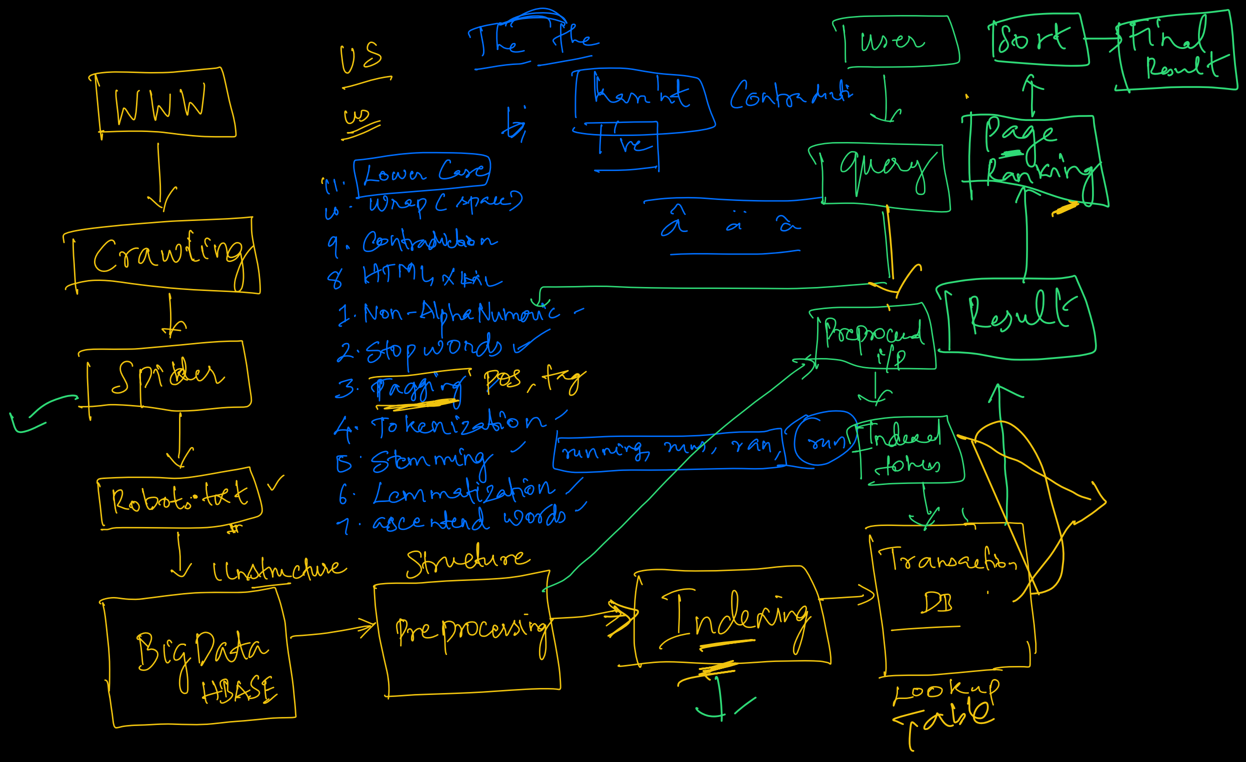


Founder of Google

Larry Page + Ranking → Page Ranking

1. Hi how are you! bro

2. I am good bro

3. I. am very very good

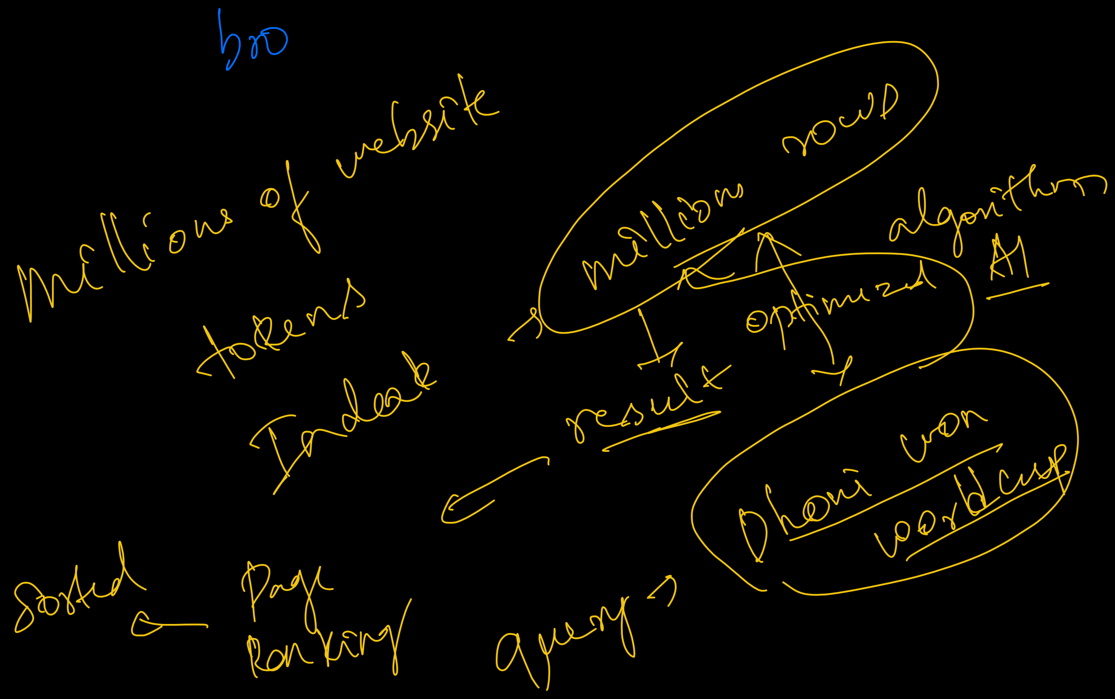
hi how you bro

i good bro

i very very good

	hi	how	you	bro	i	good	Very
1	1	1	1	1	0	0	0
2	0	0	0	1	1	1	0
3	0	0	0	0	1	1	2

indexing


bro

Millions of website

tokens

Index

millions rows

algorithm

AI

optimized

result

Dhoni won worldcup

query →

sorted ← Page Ranking

Limitation ✓

1. Frequency Count is not captured ✓

2. Order is not captured ✓

word1 : {0, 2, 5, 6}

word2 : {1, 3, 5, 6}

word3 : {2, 5, 8, 9}

✓

— | movie is very bad ✓ —

— | movie is very very bad ✓ —ve

3. Context is not captured

4. Meaning is not captured

8.

1. Count vectorizer ✓ One hot

Doc1 = I like cats. do you like ?

Doc 2 = I like dogs to

Vocabulary = [I, like, cats, dogs, too, do, you]

	I	like	cats	dogs	to	do	you
doc 1	1	1	1	0	0	1	1
doc 2	1	1	0	1	1	0	0

Structured

2. Bag of words

	I	like	cats	dogs	to	do	you
doc 1	1	2	1	0	0	1	1
doc 2	1	1	0	1	1	0	0

Structured

3. Term Frequency - Inverse Document Frequency

TF-IDF

DOC1 = I like cats. do you like ?

DOC2 = I like dogs to

Term Frequency DOC1 => 5

DOC2 => 4

log IDF

	TF(DOC1)	TF(DOC2)	IDF
I	1/5	1/4	2/2 = 1 => 0
like	2/5	1/4	2/2 = 1 => 0
cats	1/5	0/4	2/1 = 2 = -3
do	1/5	0/4	2/1 = 2 = 0.3
you	1/5	0/4	2/1 = 2 = 0.3
dogs	0/5	1/4	2/1 = 2 = 0.3
too	0/5	1/4	2/1 = 2 = 0.3

Inverse document Frequency

(Total number of documents) / (Number of doc which has word)

IDF => log(Tot nos doc / Nos of doc which has word)

TF·IDF

=> Doc1 = I like cats. do you like ?

Doc2 = I like dogs to

0.3

like

TF IDF

DOC 1

1 ✓

TF·IDF(I) => 1/5 * 0 = 0 ✓

TF·IDF(like) = 2/5 * 0 = 0 ✓

TF·IDF(cats) = 1/5 * 0.3 = 0.06

TF·IDF(

* TF·IDF

= TF * log(IDF) ✓

= TF * (1 + log(IDF))

why?

1000 document

10000 The

$$\left(\frac{1000}{1000}\right) = 1$$

$$\log(1) = \boxed{0}$$