Information Retrieval

Venkatesh Vinayakarao

Term: Aug – Dec, 2018

Indian Institute of Information Technology, Sri City

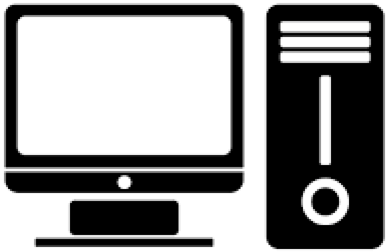
IR did not begin with the web. It began with searching library records. Nevertheless, in recent years, a principal driver has been the WWW.
– Adapted from Preface of Manning's book.

Review

Map-Reduce for Distributed Indexing

One System is Insufficient

- Problem: Petabytes of Data! Heavy Computation!!



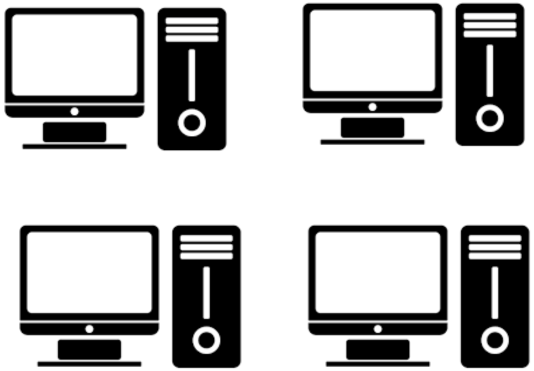
Two Solutions

- Super Computer

Sunway TaihuLight

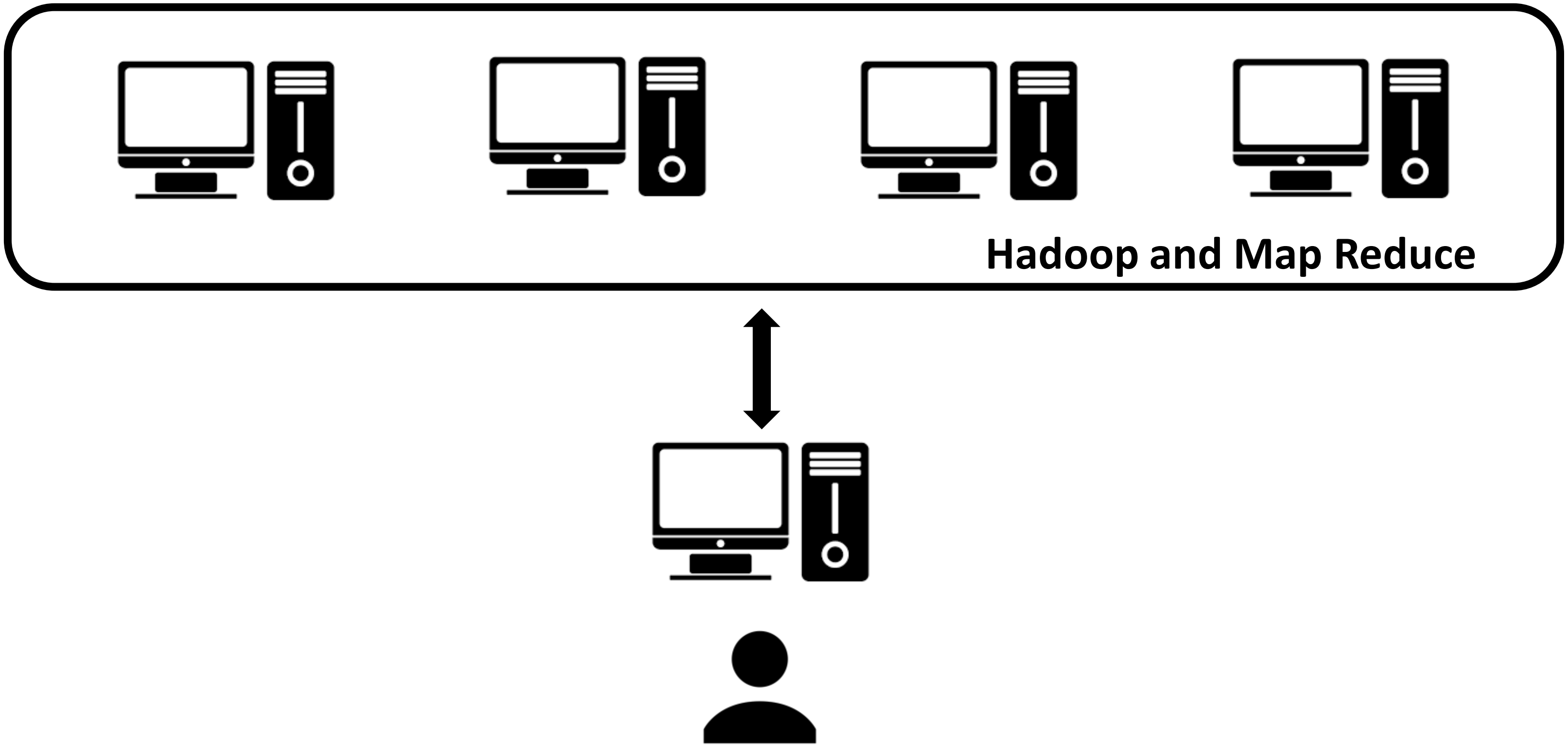
Active	June 2016
Operators	National Supercomputing Center in Wuxi
Location	National Supercomputer Center, Wuxi, Jiangsu, China
Architecture	Sunway
Power	15 MW (LINPACK)
Operating system	Sunway RaiseOS 2.0.5 (based on Linux)
Memory	1.31 PB (5591 TB/s total bandwidth)
Storage	20 PB
Speed	1.45 GHz (3.06 TFlops single CPU, 105 PFLOPS LINPACK, 125 PFLOPS peak)
Cost	1.8 billion Yuan (US$273 million)
Purpose	Oil prospecting, life sciences, weather forecast, industrial design, pharmaceutical research[citation needed]
Web site	http://www.nsccwx.cn/wxcyw/

- Use thousands of normal (commodity) systems



Exploring Solution Two!

- We interact with one. But it parallelizes our tasks!!



Fault Tolerance

- If in a non-fault-tolerant system with 1000 nodes, each node has 99.9% uptime, what is the uptime of the system?

Fault Tolerance

- If in a **non-fault-tolerant system** with 1000 nodes, each node has 99.9% uptime, what is the uptime of the system?

Answer: 37% = $(99.9\%)^{1000}$

*Assumption: System is up if all nodes are up.

Fault Tolerance

- If in a **non-fault-tolerant system** with 1000 nodes, each node has 99.9% uptime, what is the uptime of the system?

Answer: $37\% = (99.9\%)^{1000}$

- Consider a **fault-tolerant system** based on redundancy: 10 machines each with 50% uptime. What is the uptime of the system?

Fault Tolerance

- If in a **non-fault-tolerant system** with 1000 nodes, each node has 99.9% uptime, what is the uptime of the system?

Answer: 37% = $(99.9\%)^{1000}$

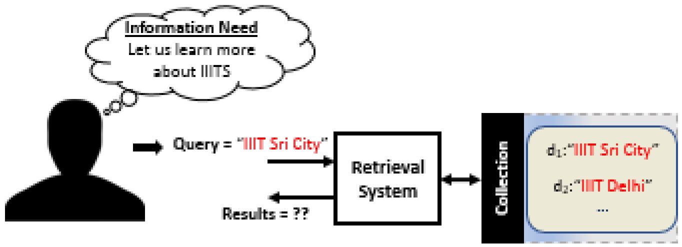
- Consider a **fault-tolerant system** based on redundancy: 10 machines each with 50% uptime. What is the uptime of the system?

Answer: Fails if all machines fail together.
$(1/2)^{10} < 0.1 \rightarrow > 99.9\%$ uptime.

Review

Term Weights & Scoring

Introduction to Retrieval

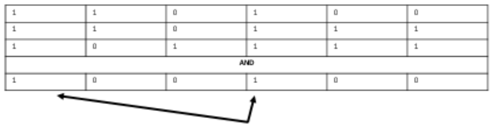


Documents

Terms

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

"Brutus and Caesar and not Calpurnia"

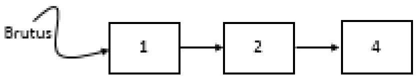


Document 1 and 4 satisfy our query.

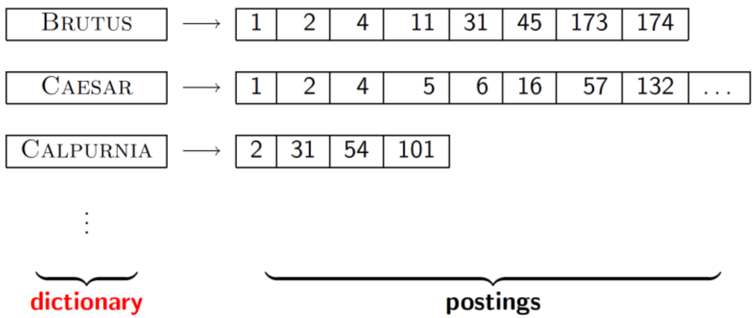
One (bad) Approach

- First match the **term** IIIT.
  - Filter out documents that contain this term.
- Next match the **term** Sri.
  - Filter out documents that contain this term.
- Next match the **term** City.
  - Filter out documents that contain this term.

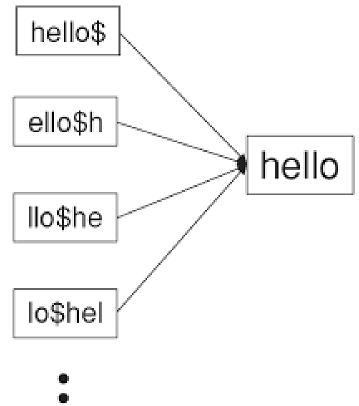
~~int[] A = {1,1,1};~~



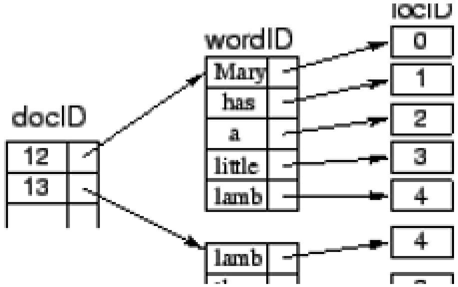
Indexing



Inverted Index



Permuterm Index

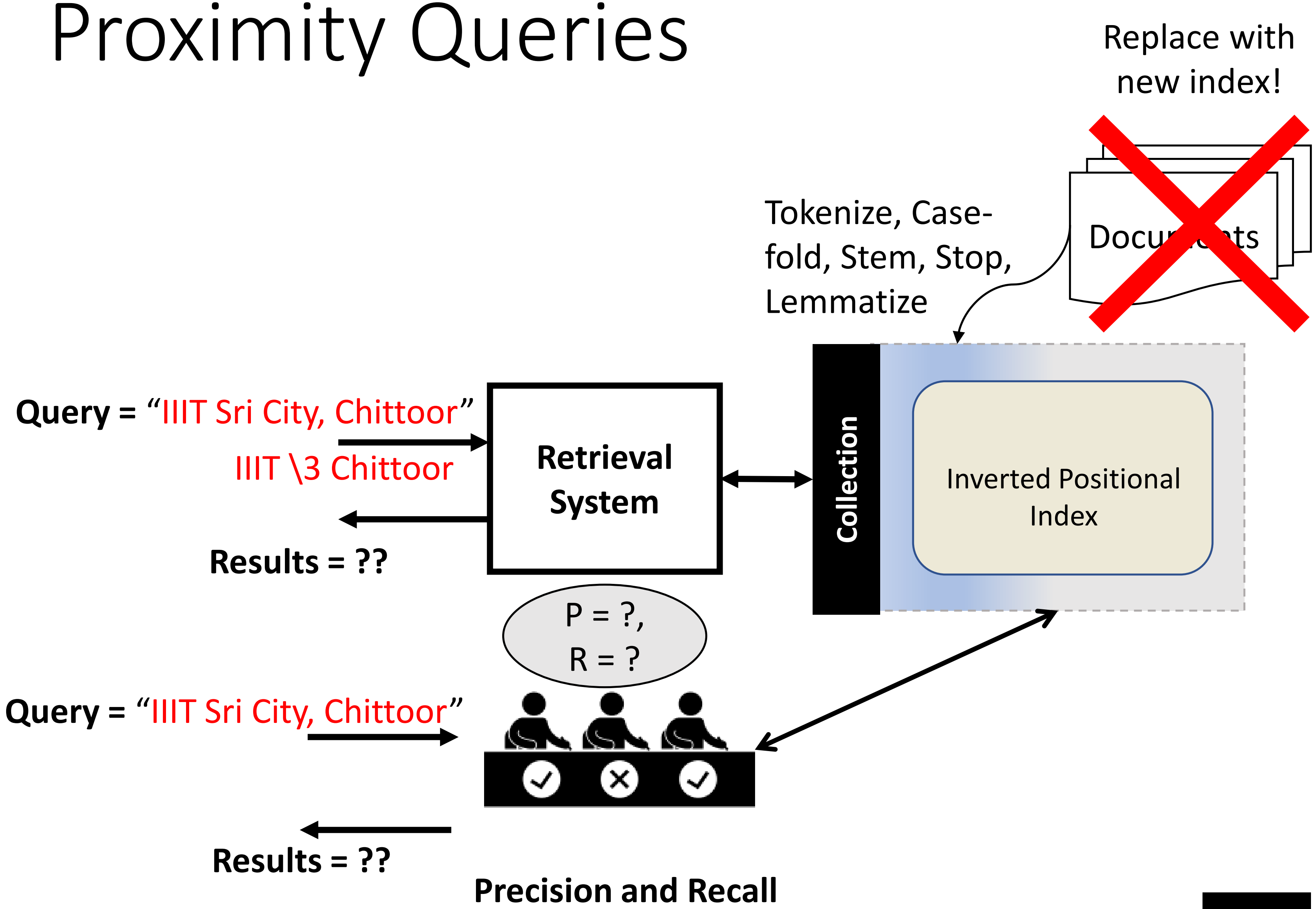


Forward Index

Term	Freq	Postings & Positions
and	1	(6,1) (6,6)
big	2	(2,3) (2,8) (3,8)
dark	1	(6,5)
did	1	(4,7)
gown	1	(2, 10)
had	1	(3,6)
house	2	(2,5) (3,2)
in	5	<(1,8)> <(2,1)> <(2,6)> <(3,3)> <(5,7)> <(6,3)> <(6, 8)>
keep	3	(1,7) (3,10) (5,6)

Positional Inverted Index

Proximity Queries



Cosine Similarity

Let query q = “BITS Pilani”.
Let document, d_1 = “BITS Pilani Goa Campus” and d_2 = “IIIT Delhi”.

	BITS	Pilani	Goa	Campus	IIIT	Delhi
q	1	1	0	0	0	0
d_1	1	1	1	1	0	0
d_2	0	0	0	0	1	1

In our VSM, q = (1,1,0,0,0,0), d_1= (1,1,1,1,0,0) and d_2 = (0,0,0,0,1,1)

similarity(d_1, q) $= \frac{d_1.q}{||d_1||\ ||q||} = \frac{1.1 + 1.1}{\sqrt{1^2+1^2+1^2+1^2}\ \sqrt{1^2+1^2}} = 0.71.$

similarity(d_2, q) $= \frac{d_2.q}{||d_2||\ ||q||} = 0.$

Length Normalization

- Suppose:
  - d1: IIITS is great.
  - d2: IIITS is great. IIITS is great.
- Assumption
  - d2 and d1 should get same similarity score since their ratio of **term frequencies (tf)** to **document length (|D|)** is same (tf/|D| is 1/3 or 2/6).
- Solution
  - Length Normalization.

Converting to Unit Vectors

- *Normalization*
  - $\frac{d_2.q}{||d_2||||q||} = \frac{d_2}{||d_2||} \times \frac{q}{||q||}$
  - $\frac{d_2}{||d_2||}$ and $\frac{q}{||q||}$ are unit vectors.

Length Normalization

- Converting to Unit Vectors

	Before Normalization				After Normalization		
term	**d1**	**d2**	**d3**		**d1**	**d2**	**d3**
affection	115.000	58.000	20.000		0.996	0.993	0.847
jealous	10.000	7.000	11.000		0.087	0.120	0.466
gossip	2.000	0.000	6.000		0.017	0.000	0.254
|| V(d) ||	**115.451**	**58.421**	**23.601**		**1.000**	**1.000**	**1.000**

- Before Normalization, V(d1) = (115, 10, 2)
- After Normalization, v(d1) = (0.996, 0.087, 0.017)

Calculating TF-IDF

- After Length Normalization

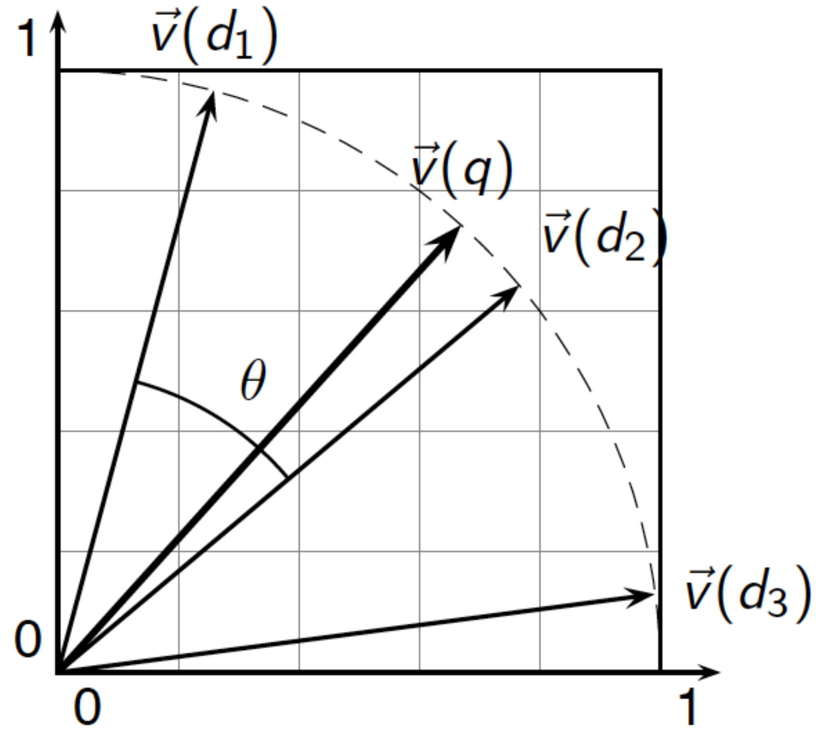
term	SaS	PaP	WH
affection	0.789	0.832	0.524
jealous	0.515	0.555	0.465
gossip	0.335	0	0.405
wuthering	0	0	0.588

Term frequencies (counts)

Unit Vectors

- Now, Similarity:

sim(d1,d2) = v(d1).v(d2)



Quiz

- Can you length normalize the vector (1,0,1,2) ?

Answer: (0.41, 0, 0.41, 0.82)

Hint: Normalization Factor = $\sqrt{1^2 + 0 + 1^2 + 2^2}$ = $\sqrt{6}$

Normalized Vector = (1/$\sqrt{6}$, 0, 1/$\sqrt{6}$, 2/$\sqrt{6}$)

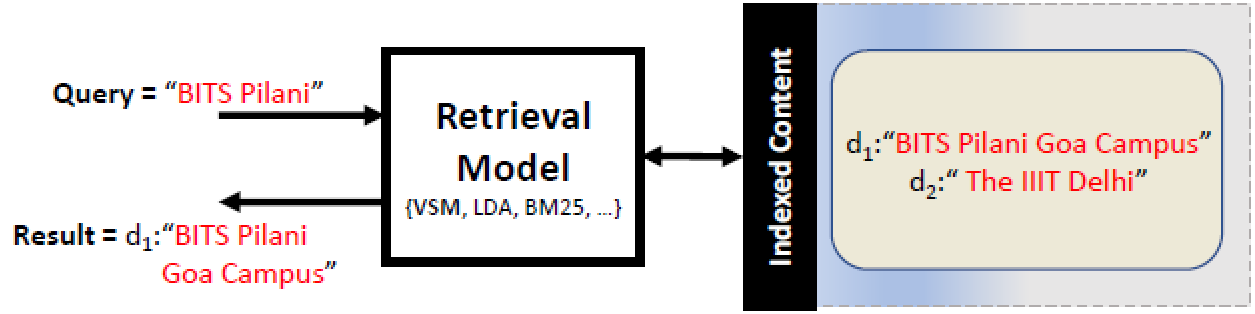
Quiz

- What is the cosine similarity between the **unit** vectors:
  - (0.996, 0.087, 0.017) and
  - (0.993, 0.120, 0)

Answer: 0.999

Hint: Simply take the dot product between the two unit vectors.

Not Every Term is Important



Let us add Term Weights

	BITS	the (* 0)	Pilani	Goa	Campus	IIIT	Delhi
q	1	1 * 0 = 0	1	0	0	0	0
d_1	1	0 * 0 = 0	1	1	1	0	0
d_2	0	1 * 0 = 0	0	0	0	1	1

$sim(q,d_1) = 0.71$

$sim(q,d_2) = 0$

Two Ideas

- Document containing more occurrences of query term is more relevant to the query.
- Terms that occur in fewer documents are more important in the query (for relevance computation).

$$\textbf{Relevance} \propto \textbf{tf}$$

$$\textbf{Relevance} \propto \frac{\mathbf{1}}{\boldsymbol{df}}$$

➡

Relevance ∝ TF * IDF

Scoring with tf-idf weighting

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

Compute the Scores

Question: Let N = 1,000,000 Documents. Let the weight of query term be its IDF. For documents, assume tf as the weight with Euclidean normalization.

term	query				document			product
	tf	df	idf	weighted	tf	weight	Normalized	
auto	0	5000	2.3	0	1	1	0.41	0
best	1	50000	1.3	1.3	0	0	0	0
car	1	10000	2.0	2.0	1	1	0.41	0.82
insurance	1	1000	3.0	3.0	2	2	0.82	2.46

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

Net Score = 0 + 0 + 0.82 + 2.46 = 3.28

Another Example

Question: Let N = 1,000,000 Documents.

term	query					document				product
	tf	df	idf	tf.idf	tf.idf(N)	tf	idf	tf.idf	tf.idf(N)	
auto	0	5000				1				
best	1	50000				0				
car	1	10000				1				
insurance	1	1000				2				

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

Net Score = 0 + 0 + 0.155 + 0.698 = 0.853

Another Example

Question: Let N = 1,000,000 Documents.

term	query					document				product
	tf	df	idf	tf.idf	tf.idf(N)	tf	idf	tf.idf	tf.idf(N)	
auto	0	5000	2.3	0	0	1	2.3	2.3	0.342	0
best	1	50000	1.3	1.3	0.339	0	1.3	0	0	0
car	1	10000	2.0	2.0	0.522	1	2.0	2.0	0.297	0.155
insurance	1	1000	3.0	3.0	0.783	2	3.0	6.0	0.892	0.698

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

Net Score = 0 + 0 + 0.155 + 0.698 = 0.853