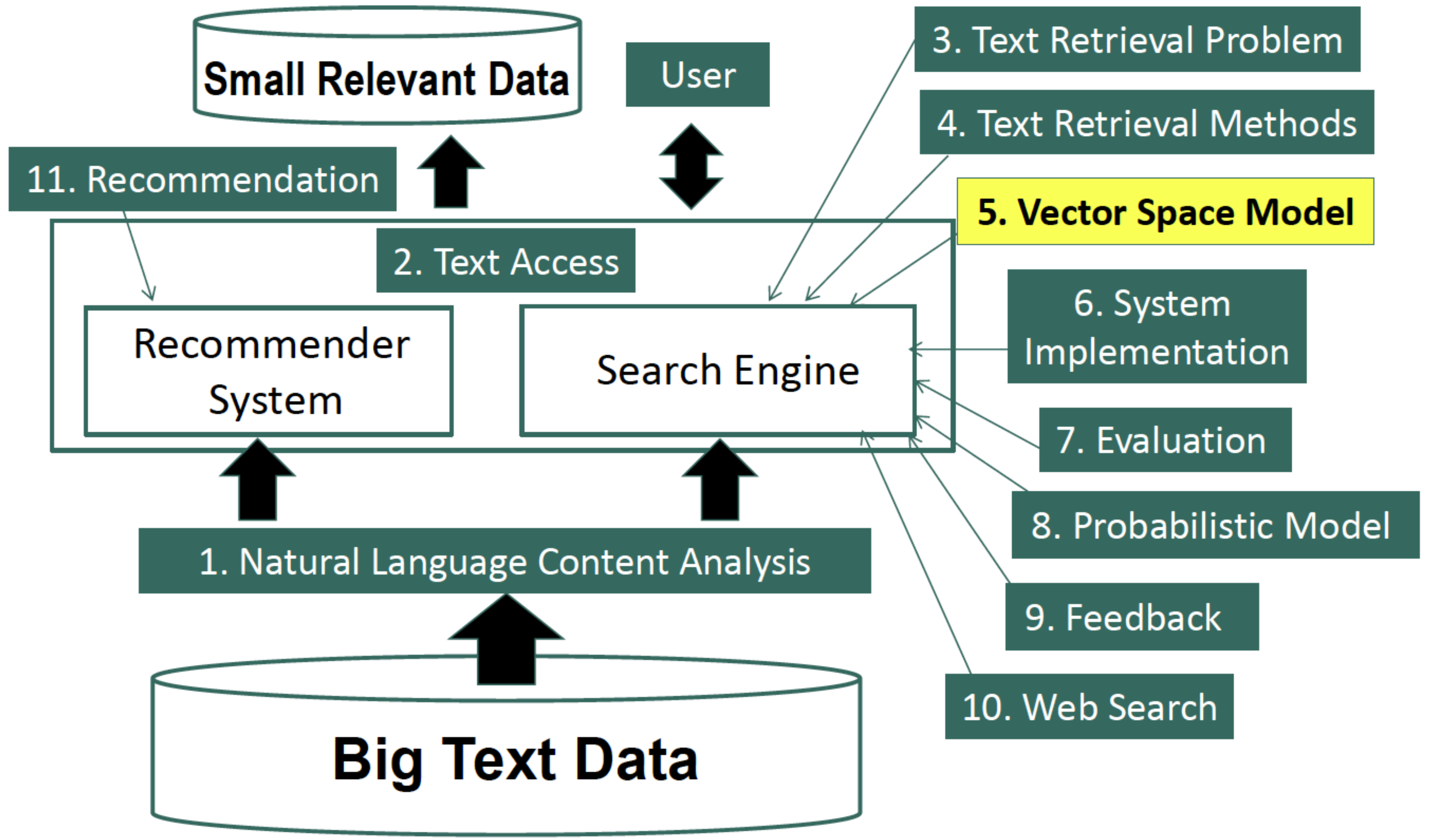
Information Retrieval & Text Mining

Vector Space Model
Improved Instantiation

Dr. Saeed Ul Hassan
Information Technology University

Course Schedule



Two Problems of the Simplest VSM

Query = "news about presidential campaign"

d2	… **news about** organic food **campaign**…	**f(q,d2)=3**
d3	… **news** of **presidential campaign** …	**f(q,d3)=3**
d4	… **news** of **presidential campaign** … … **presidential** candidate …	**f(q,d4)=3**

?

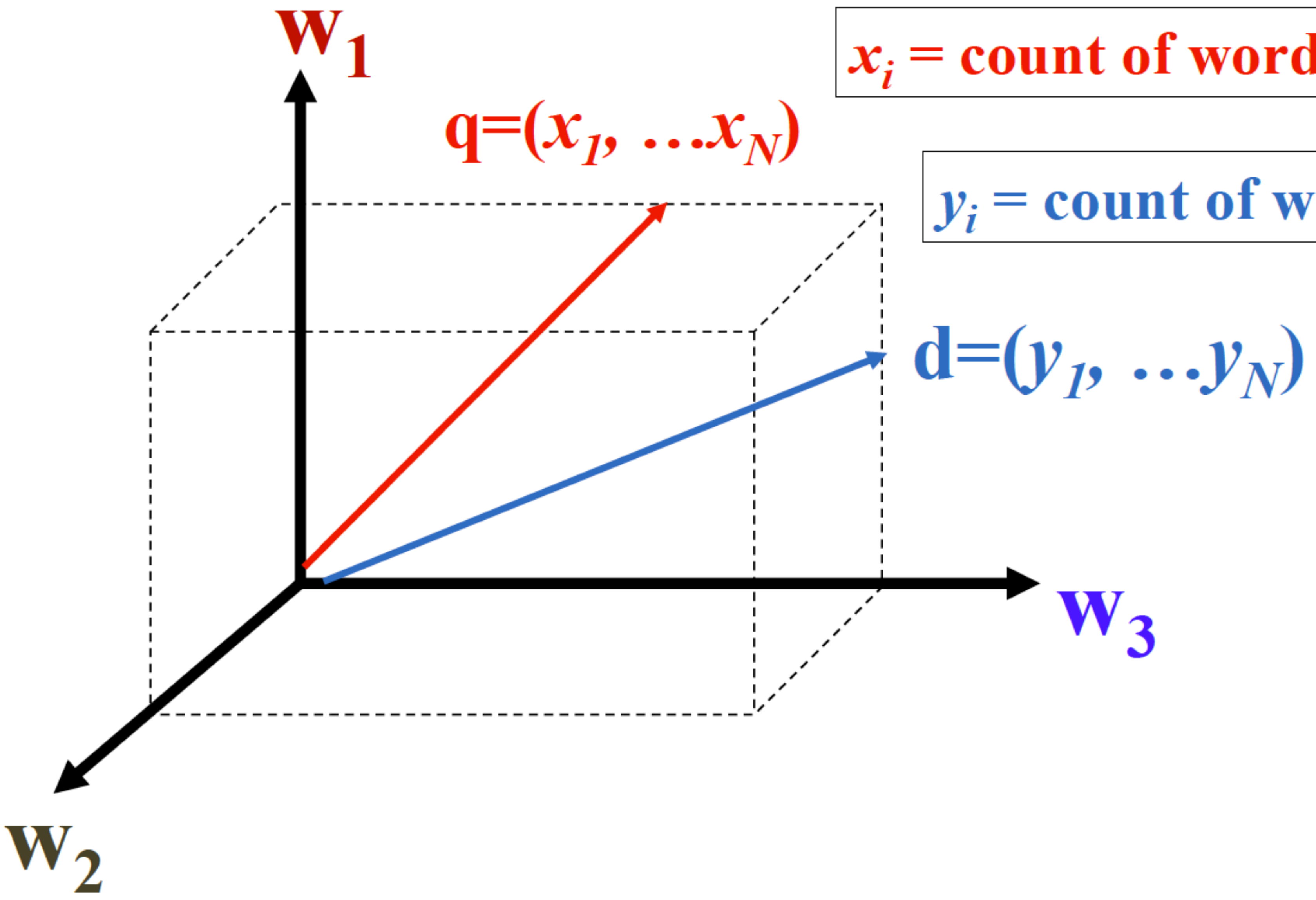
Two Problems of the Simplest VSM

Query = "news about presidential campaign"

d2	… **news about** organic food **campaign**…	**f(q,d2)=3**
d3	… **news** of **presidential campaign** …	**f(q,d3)=3**
d4	… **news** of **presidential campaign** … … **presidential** candidate …	**f(q,d4)=3**

1. Matching "presidential" more times deserves more credit
2. Matching "presidential" is more important than matching "about"

Improved Vector Placement: Term Frequency Vector



x_i = count of word W_i in query

y_i = count of word W_i in doc

Improved VSM with Term Frequency Weighting

$q=(x_1, \ldots x_N)$ x_i = count of word W_i in query

$d=(y_1, \ldots y_N)$ y_i = count of word W_i in doc

$$Sim(q,d)=q.d= x_1 y_1 + \ldots + x_N y_N = \sum_{i=1}^{N} x_i y_i$$

What does this ranking function intuitively capture?

Does it fix the problems of the simplest VSM?

Ranking Using Term Frequency (TF) Weighting

d2 … **news about** organic food **campaign**… f(q,d2)=3

q=	(1,	1,	1,	1,	0,	…)
d2=	(1,	1,	0,	1,	1,	…)

d3 … **news** of **presidential campaign** … f(q,d3)=3

q=	(1,	1,	1,	1,	0,	…)
d3=	(1,	0,	1,	1,	0,	…)

d4 … **news** of **presidential campaign** …
… **presidential** candidate … **f(q,d4)=4!**

q=	(1,	1,	1,	1,	0,	…)
d4=	(1,	0,	2,	1,	0,	…)

How to Fix Problem 2 (“presidential” vs. “about”)

d2 … **news about** organic food **campaign**…

d3 … **news** of **presidential campaign** …

V= {news, about, presidential, campaign, food …. }

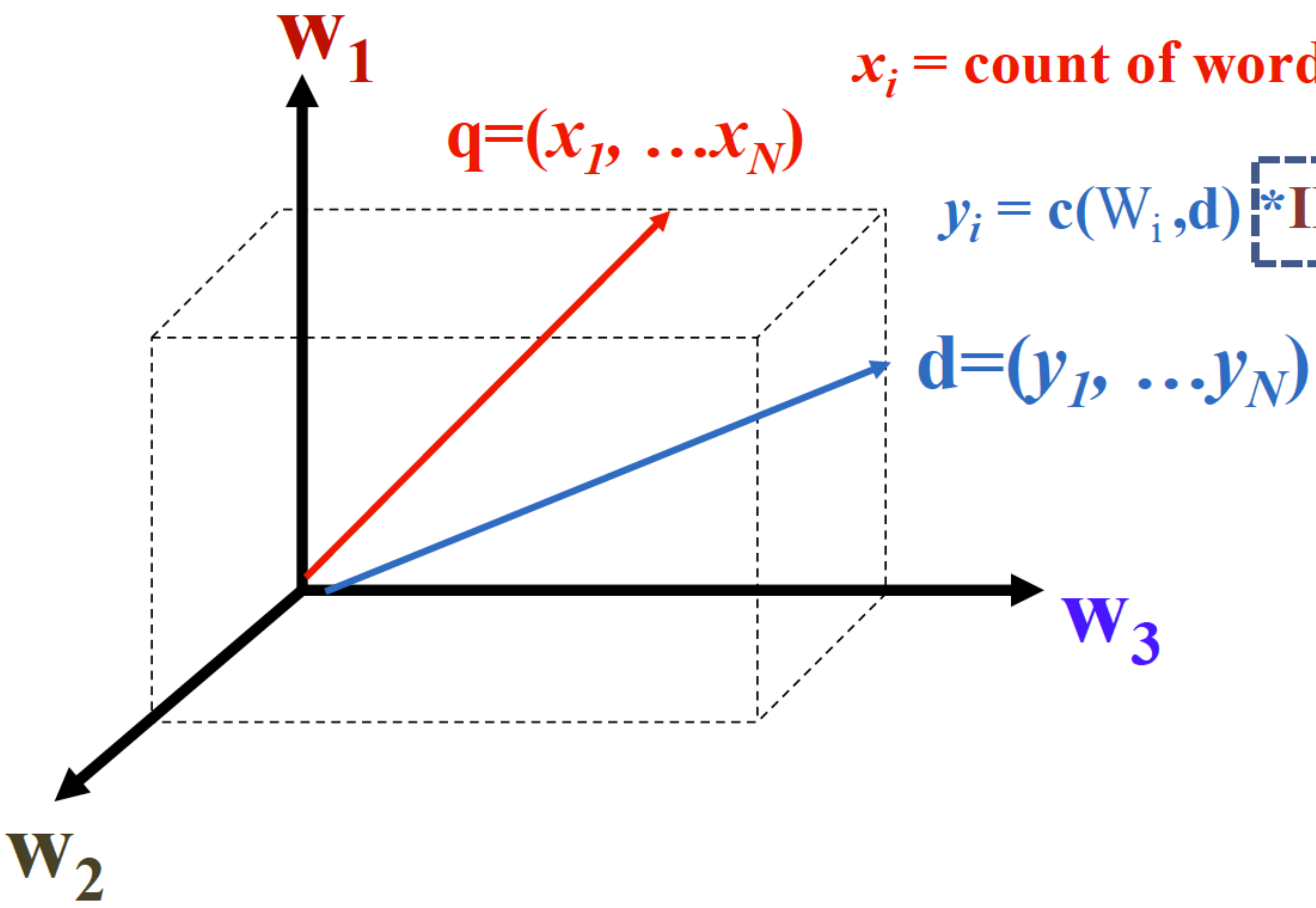
$$
\begin{array}{lllllll}
q= & (\boxed{1,} & \boxed{1,} & 1, & \boxed{1,} & 0, & \ldots) \\
d2= & (\boxed{1,} & \boxed{1,}\Downarrow & 0, & \boxed{1,} & 1, & \ldots)
\end{array}
$$

$f(q,d2)<3$

$f(q,d3)>3$

$$
\begin{array}{lllllll}
q= & (\boxed{1,} & 1, & \boxed{1,} & \boxed{1,} & 0, & \ldots) \\
d3= & (\boxed{1,} & 0, & \boxed{1,}\Uparrow & \boxed{1,} & 0, & \ldots)
\end{array}
$$

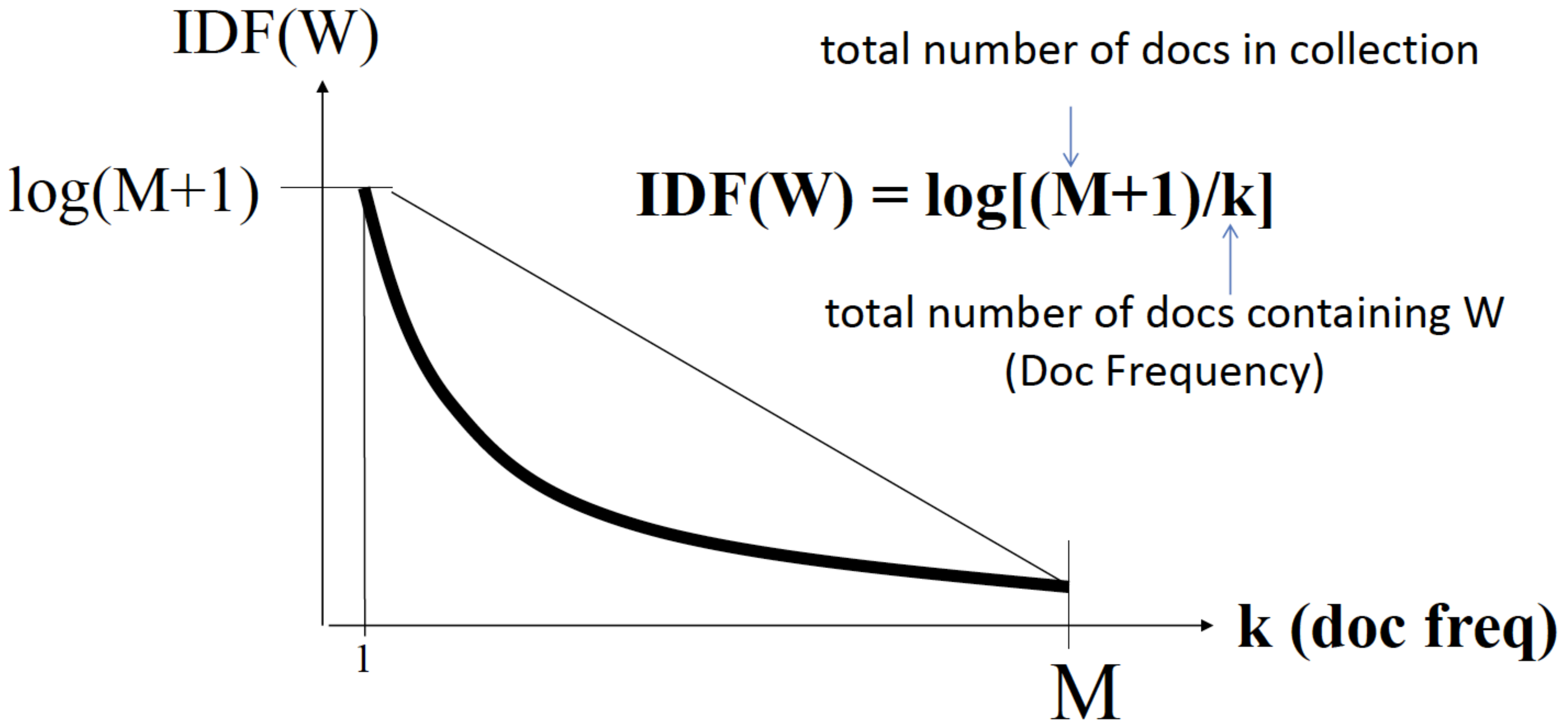
Further Improvement of Vector Placement: Adding Inverse Document Frequency (IDF)



x_i = count of word W_i in query

$y_i = c(W_i, d) \boxed{* IDF(W_i)}$

IDF Weighting: Penalizing Popular Terms

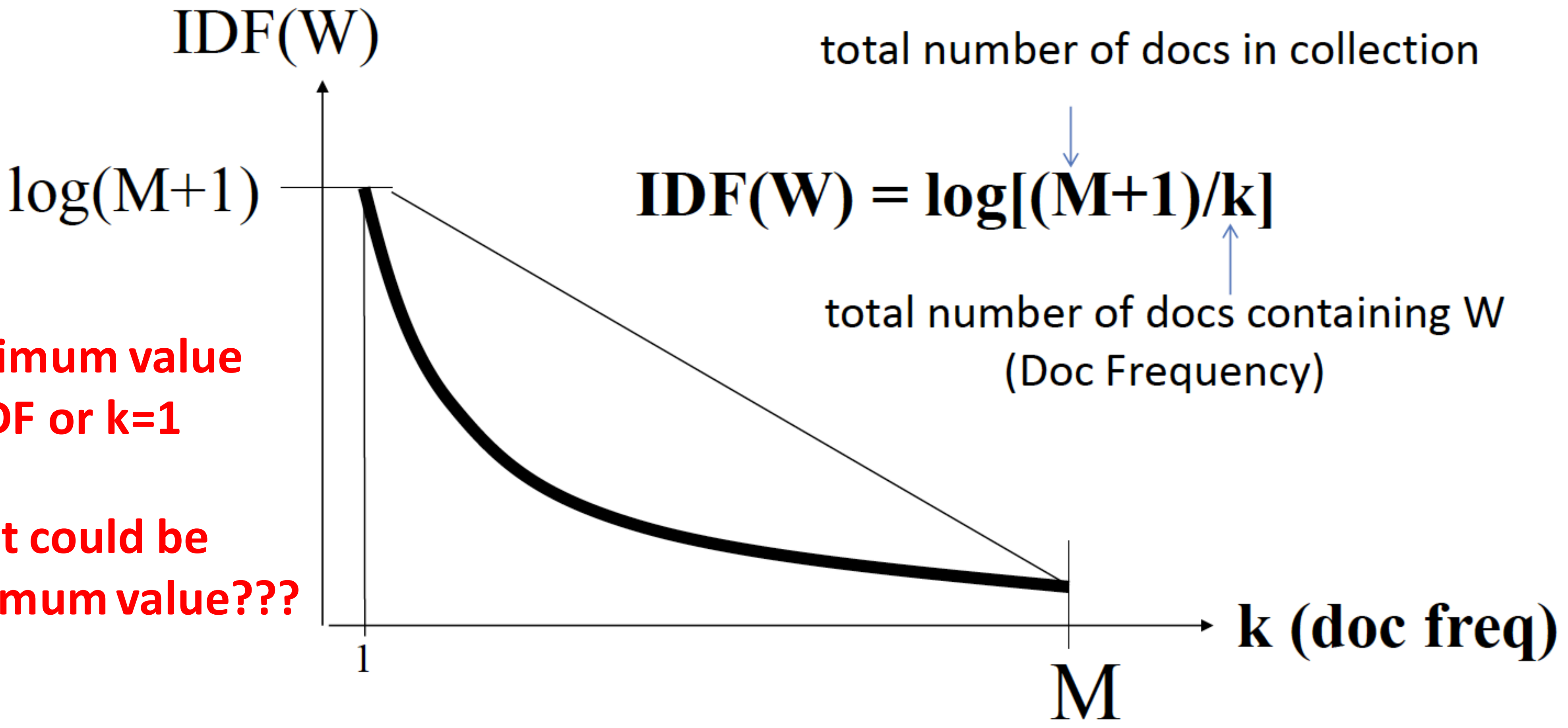


total number of docs in collection

$$\mathbf{IDF(W) = \log[(M+1)/k]}$$

total number of docs containing W
(Doc Frequency)

IDF Weighting: Penalizing Popular Terms



total number of docs in collection

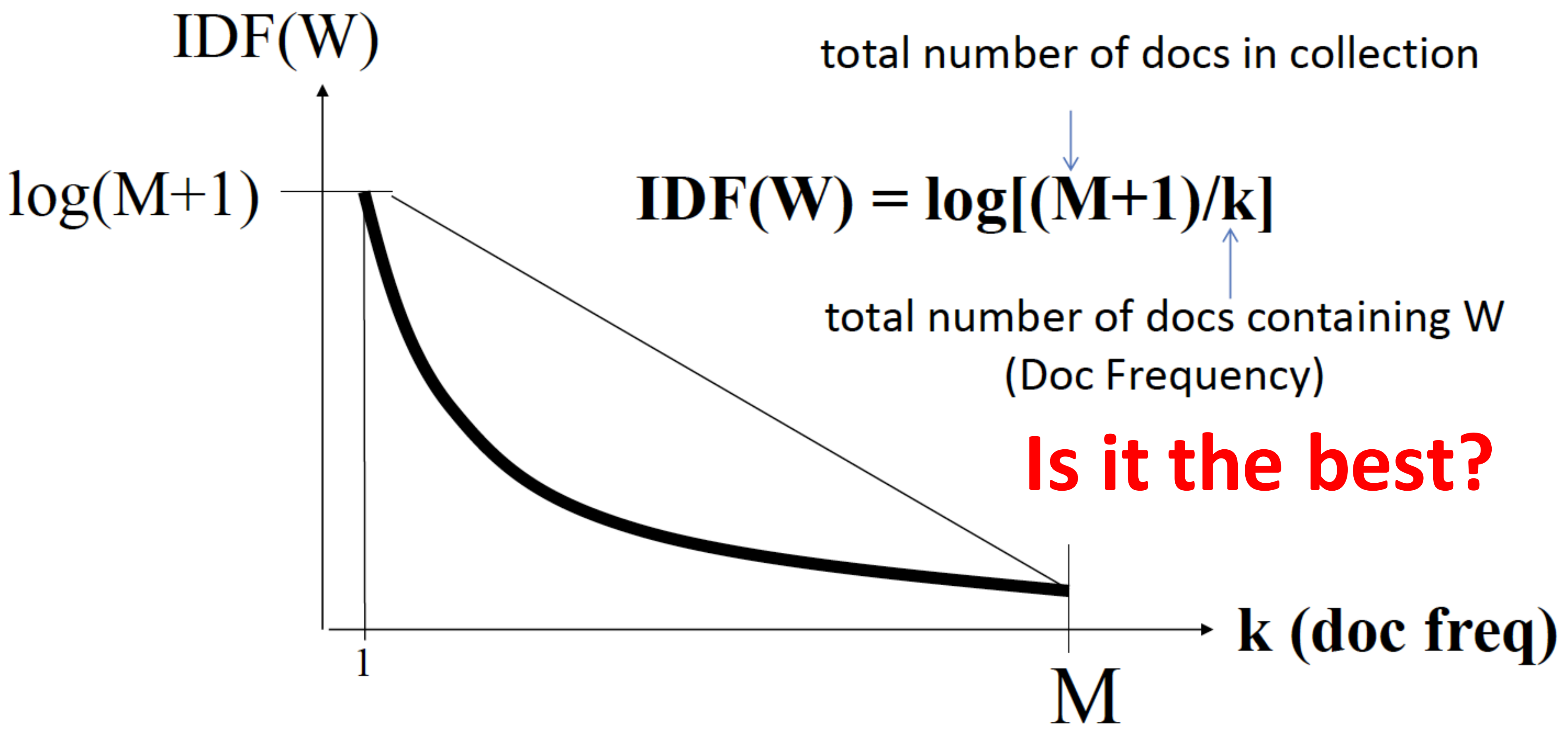
$$IDF(W) = \log[(M+1)/k]$$

total number of docs containing W
(Doc Frequency)

Maximum value For DF or k=1

What could be minimum value???

IDF Weighting: Penalizing Popular Terms



total number of docs in collection

$$IDF(W) = \log[(M+1)/k]$$

total number of docs containing W
(Doc Frequency)

Is it the best?

Solving Problem 2 (“Presidential” vs “About”)

d2 … **news about** organic food **campaign**…

d3 … **news** of **presidential campaign** …

V=	{news,	about,	presidential,	campaign,	food	…. }
IDF(W)=	**1.5**	**1.0**	**2.5**	**3.1**	**1.8**	
q=	(1,	1,	1,	1,	0,	…)
d2=	(1*1.5,	1*1.0	0,	1*3.1,	0,	…)
q=	(1,	1,	1,	1,	0,	…)
d3=	(1*1.5,	0,	1*2.5	1*3.1,	0,	…)

f(q,d2) = 5.6 < f(q,d3)=7.1

How Effective Is VSM with TF-IDF Weighting?

Query = "news about presidential campaign"

d1	… **news about** …	f(q,d1)=2.5
d2	… **news about** organic food **campaign**…	f(q,d2)=5.6
d3	… **news** of **presidential campaign** …	f(q,d3)=7.1
d4	… **news** of **presidential campaign** … … **presidential** candidate …	f(q,d4)=9.6
d5	… **news** of organic food **campaign**… **campaign**…**campaign**…**campaign**…	**f(q,d5)=13.9!**

Summary

- Improved VSM
  - Dimension = word
  - Vector = TF-IDF weight vector
  - Similarity = dot product
  - Working better than the simplest VSM
  - Still having problems