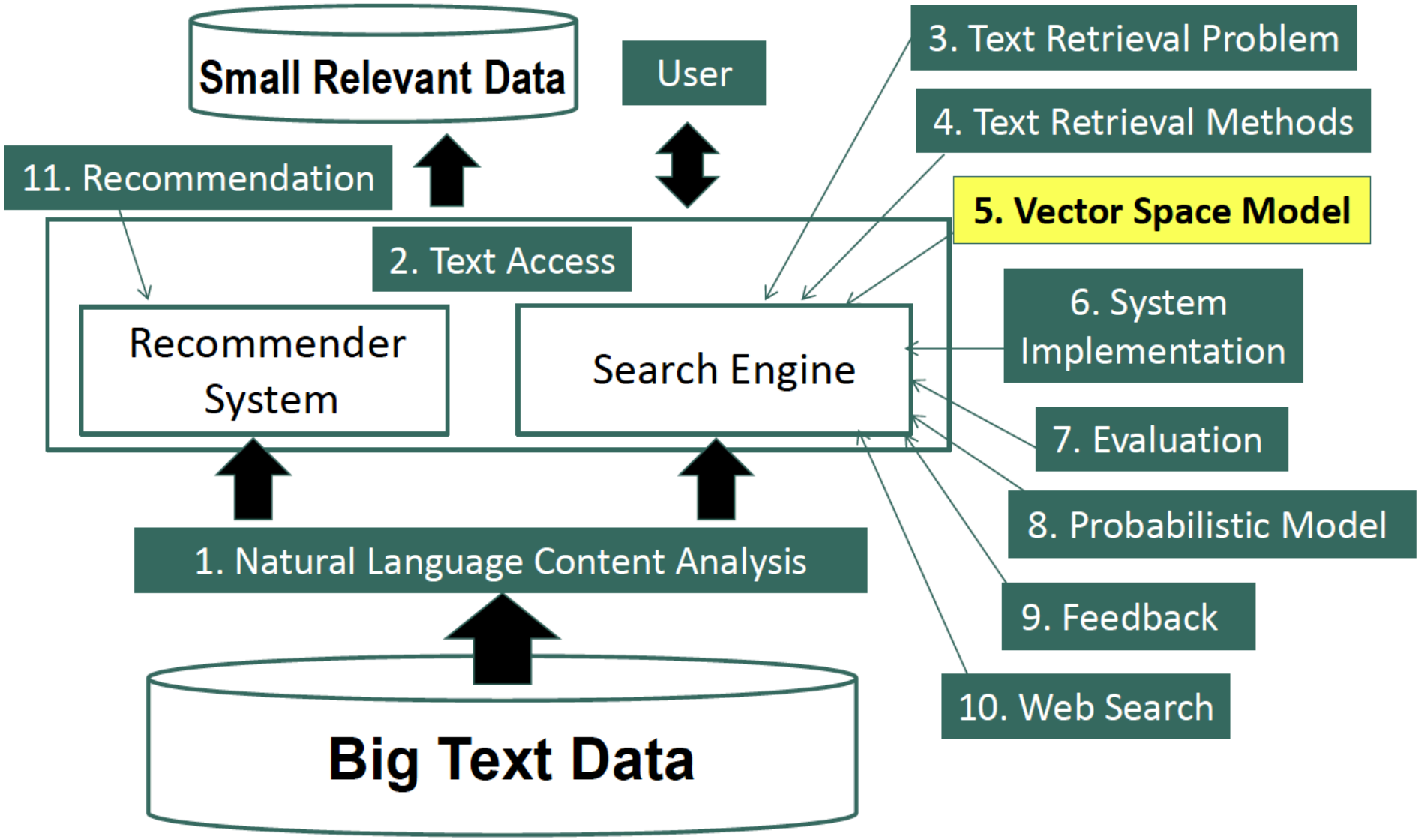
Information Retrieval & Text Mining

Vector Space Model
Improved Instantiation

Dr. Iqra Safder
FAST NUCES, Lahore

Course Schedule



An Example: How Would You Rank These Documents?

Query = "**news about presidential campaign**"

d1 … **news about** …

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

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

d4 … **news** of **presidential campaign** …
… **presidential** candidate …

d5 … **news** of organic food **campaign**…
campaign…**campaign**…**campaign**…

Ideal Ranking?

d4 +
d3 +

d1 -
d2 -
d5 -

Ranking Using the Simplest VSM

Query = "news about presidential campaign"

d1 | … **news about** …

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

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

q= (1, 1, 1, 1, 0, …)

d1= (1, 1, 0, 0, 0, …)

f(q,d1)= 1*1+1*1 +1* 0+1*0+0*0 + …= 2

d3= (1, 0, 1, 1, 0, …)

f(q,d3)= 1*1+1*0 +1*1 +1*1+0*0+ … = 3

Is the Simplest VSM Effective?

Query = "news about presidential campaign"

d1	… **news about** …	f(q,d1)=2
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
d5	… **news** of organic food **campaign**… **campaign**…**campaign**…**campaign**…	f(q,d5)=2

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**

?

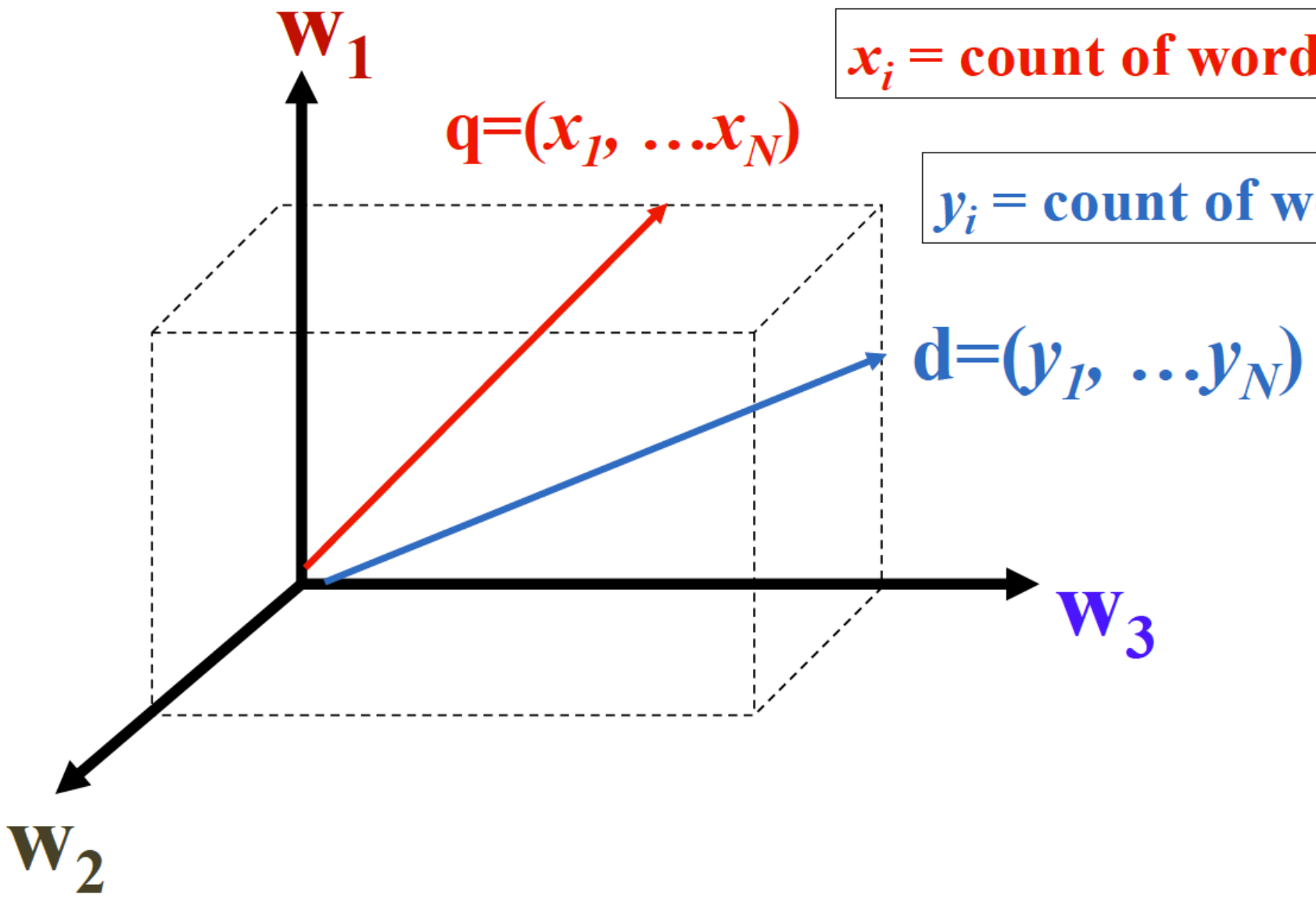
Two Problems of the Simplest VSM

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

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

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

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

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 …. }

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

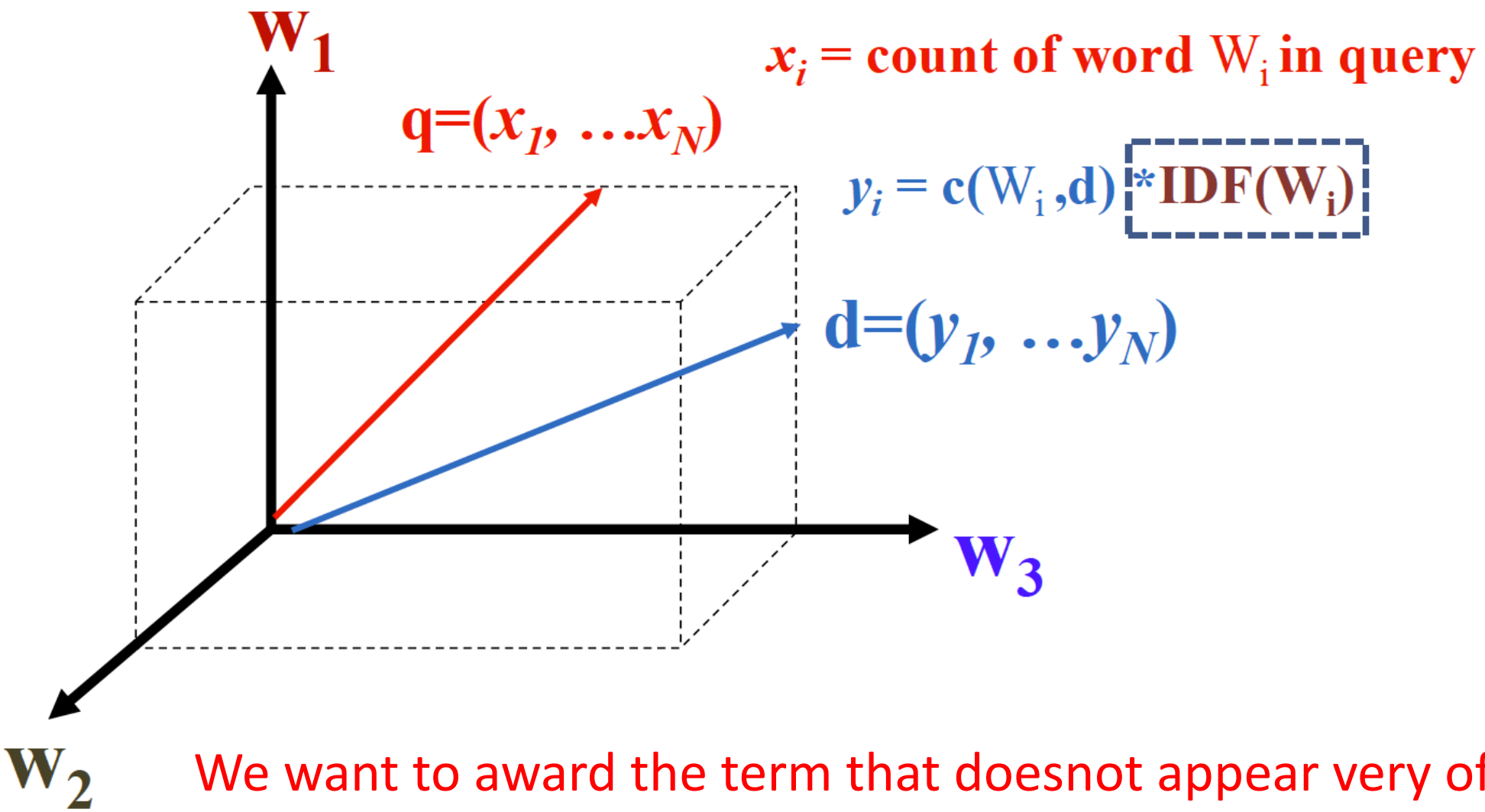
f(q,d2)<3

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

f(q,d3)>3

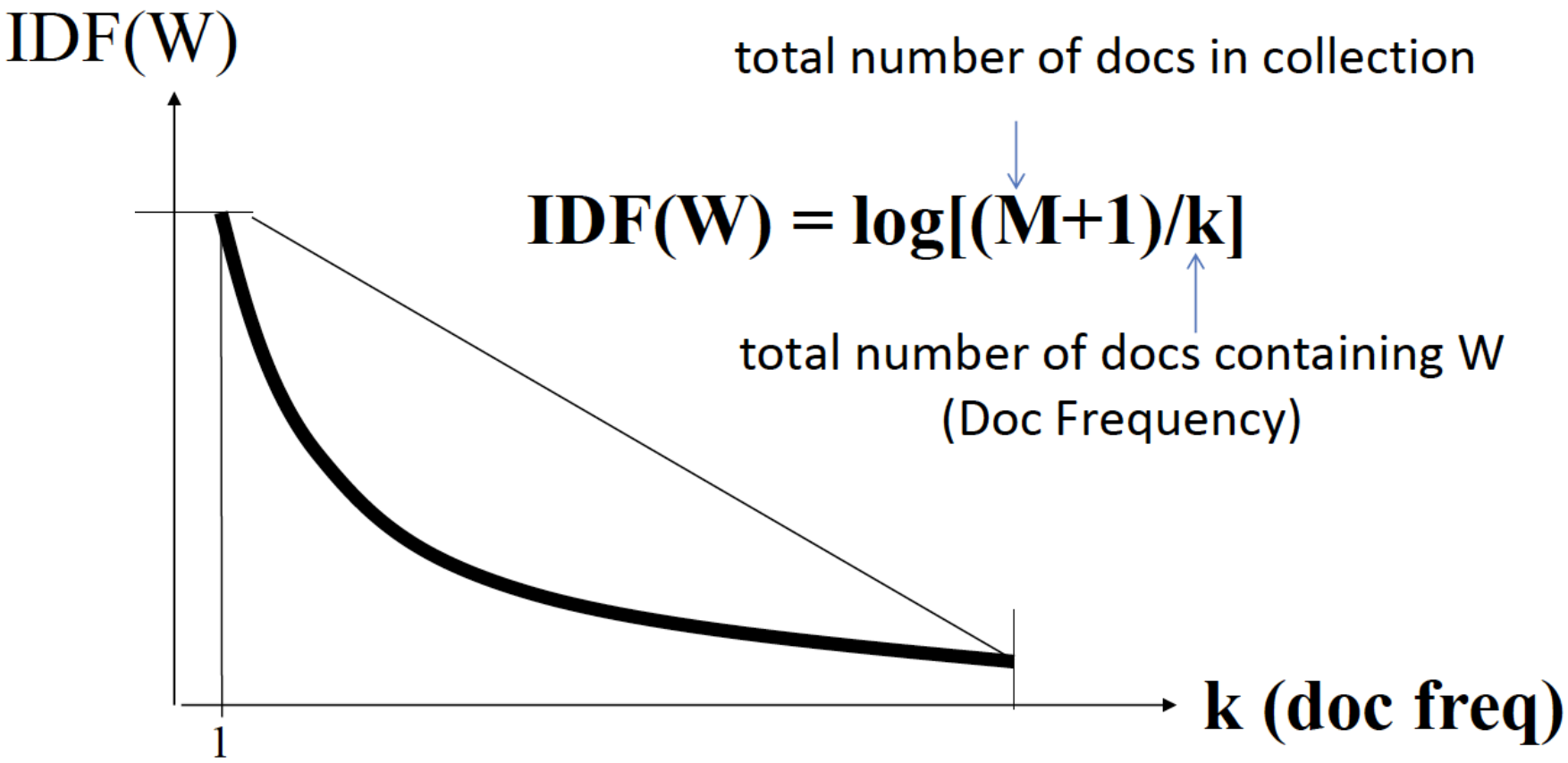
We can somehow use some global statistics of the term and some other information to down weight "about".

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



We want to award the term that doesnot appear very often.

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 …Why use log?

So here the first word "the", if our corpus is of 1000 documents will occur in almost every document but "serendipity" is a rare word and might occur is less documents, for instance we take as it has occurred in only one document.

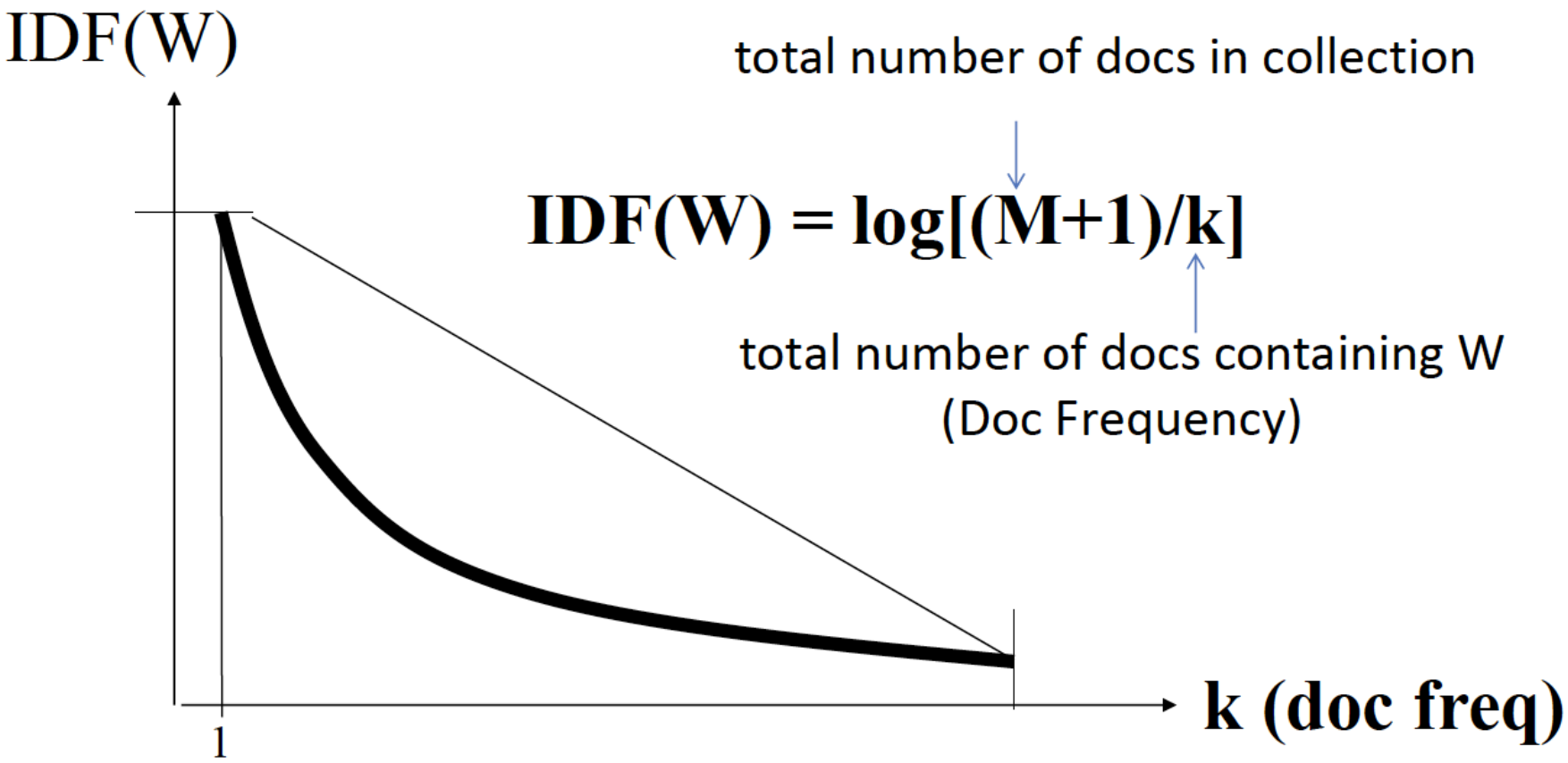
So, when calculating the IDF of both -

IDF	Log(IDF)
The = 1000/1000 = 0	0
Serendipity = 1000/1 =1000	~6.9

Now we see if we had a TF of range around 0-20 then if our IDF was not a log(IDF) then definitely it would have dominated the TF but if taken as log(IDF) then it would have a equal effect on the result as TF has.

Log is said to be used because it “dampens” the effect of IDF

IDF Weighting: Penalizing Popular Terms



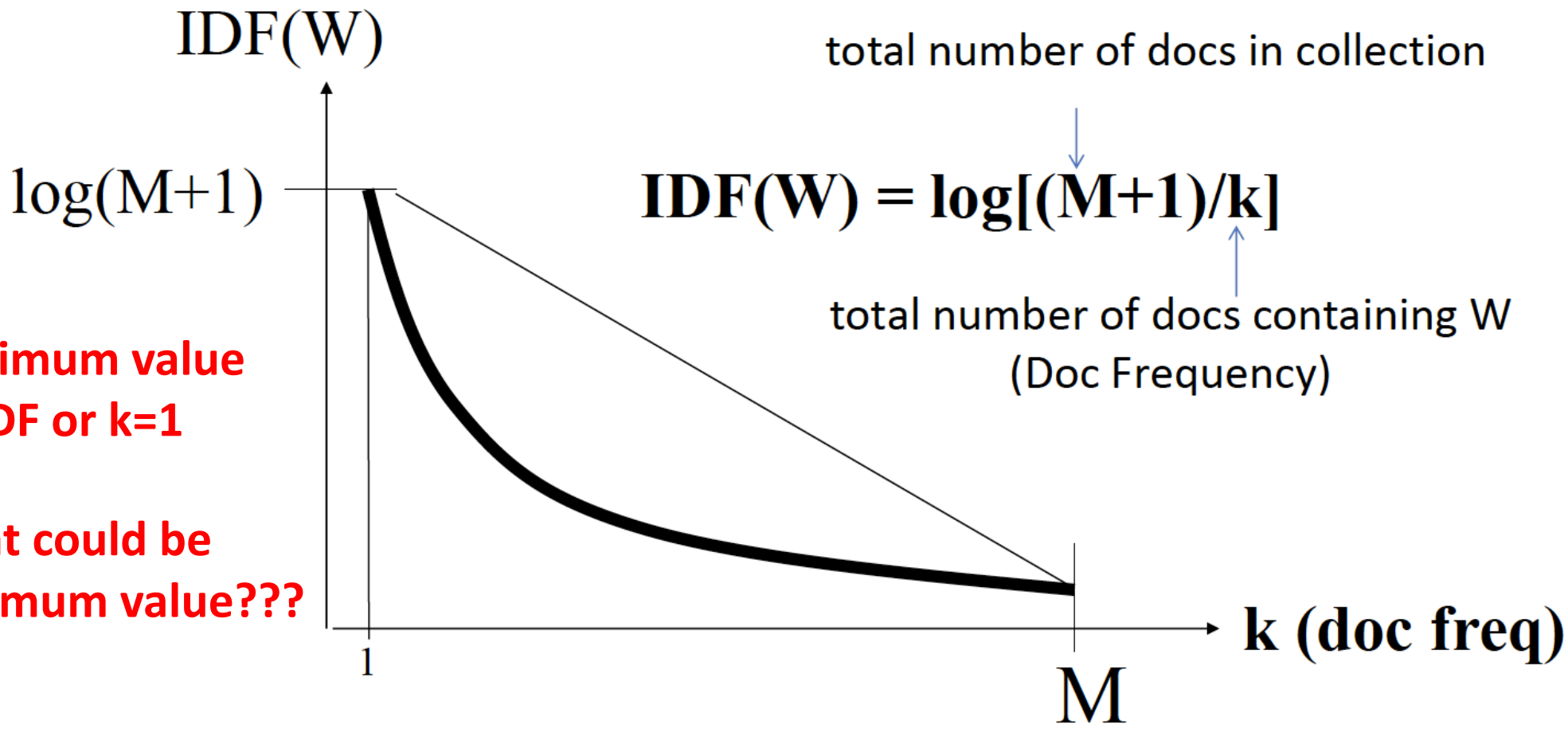
total number of docs in collection

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

total number of docs containing W
(Doc Frequency)

What will be maximum and the minimum value of this function?

IDF Weighting: Penalizing Popular Terms



total number of docs in collection

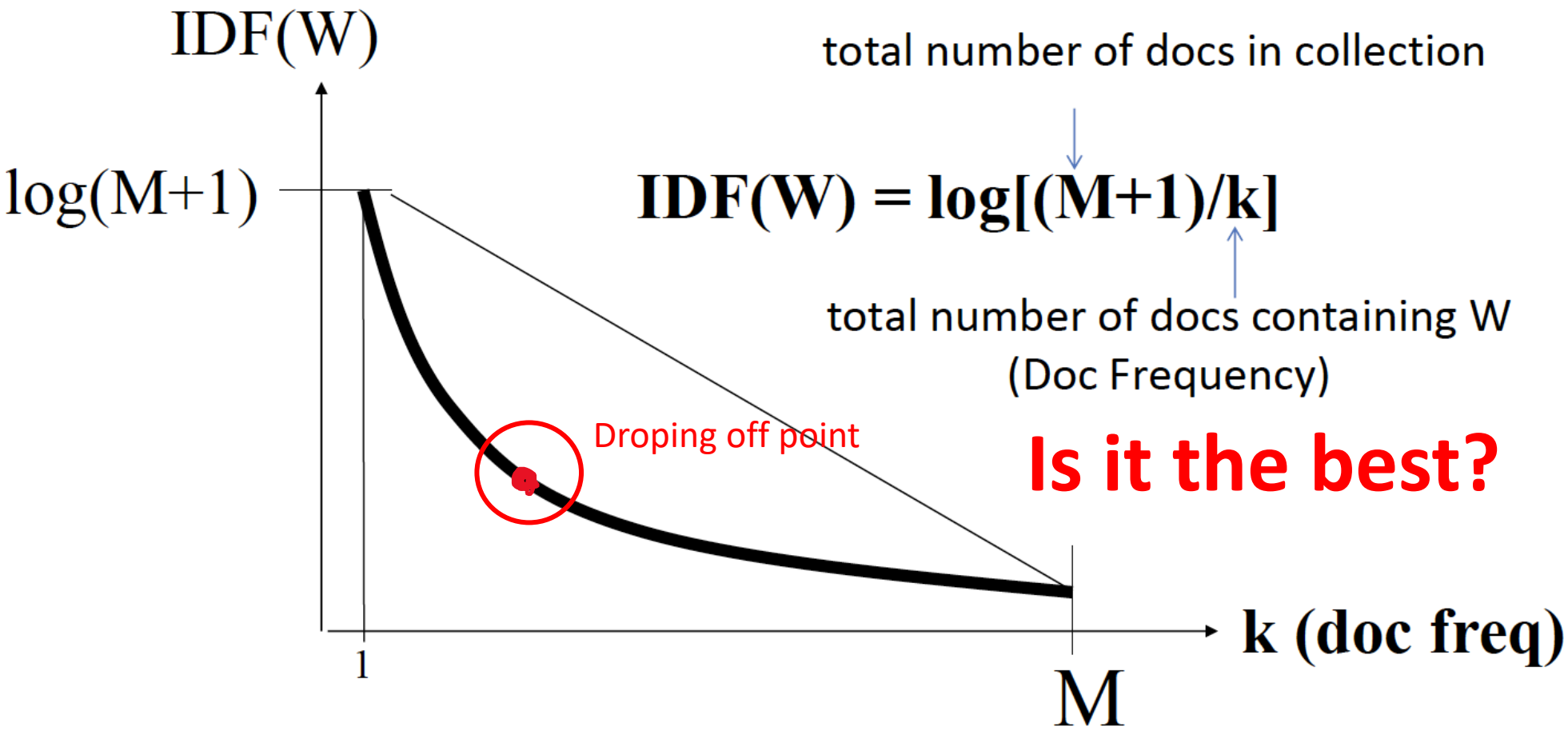
$$\mathbf{IDF(W) = \log[(M+1)/k]}$$

total number of docs containing W
(Doc Frequency)

Maximum value For DF or k=1

What could be minimum value???

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)

Is it the best?

This makes sense because when the term occur so frequently that it's unlikely to differentiate two documents relevance (since the term is so common). Intutively, we want to focus more on the descrimination of low df words rather than these common words.

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?

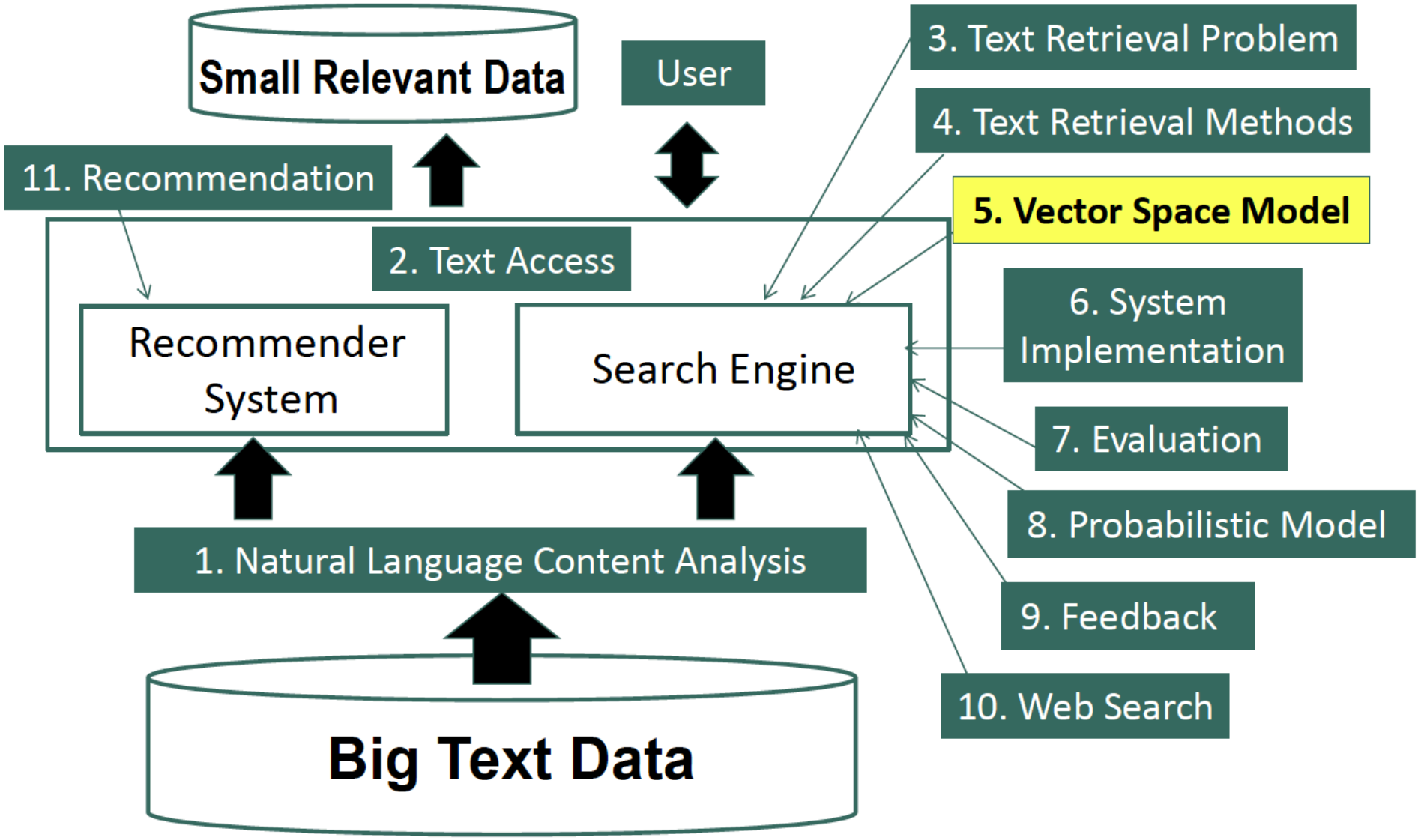
Query = "news about presidential campaign"

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

Course Schedule



VSM with TF-IDF Weighting Still Has a Problem!

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?**

When you try to fix one problem you tend to introduce another one!!

Ranking Function with TF-IDF Weighting

Total # of docs in collection

$$f(q,d) = \sum_{i=1}^{N} x_i y_i = \sum_{w \in q \cap d} c(w,q)\underline{c(w,d)} \log \frac{M+1}{df(w)}$$

All matched query words in d

Doc Frequency

d5

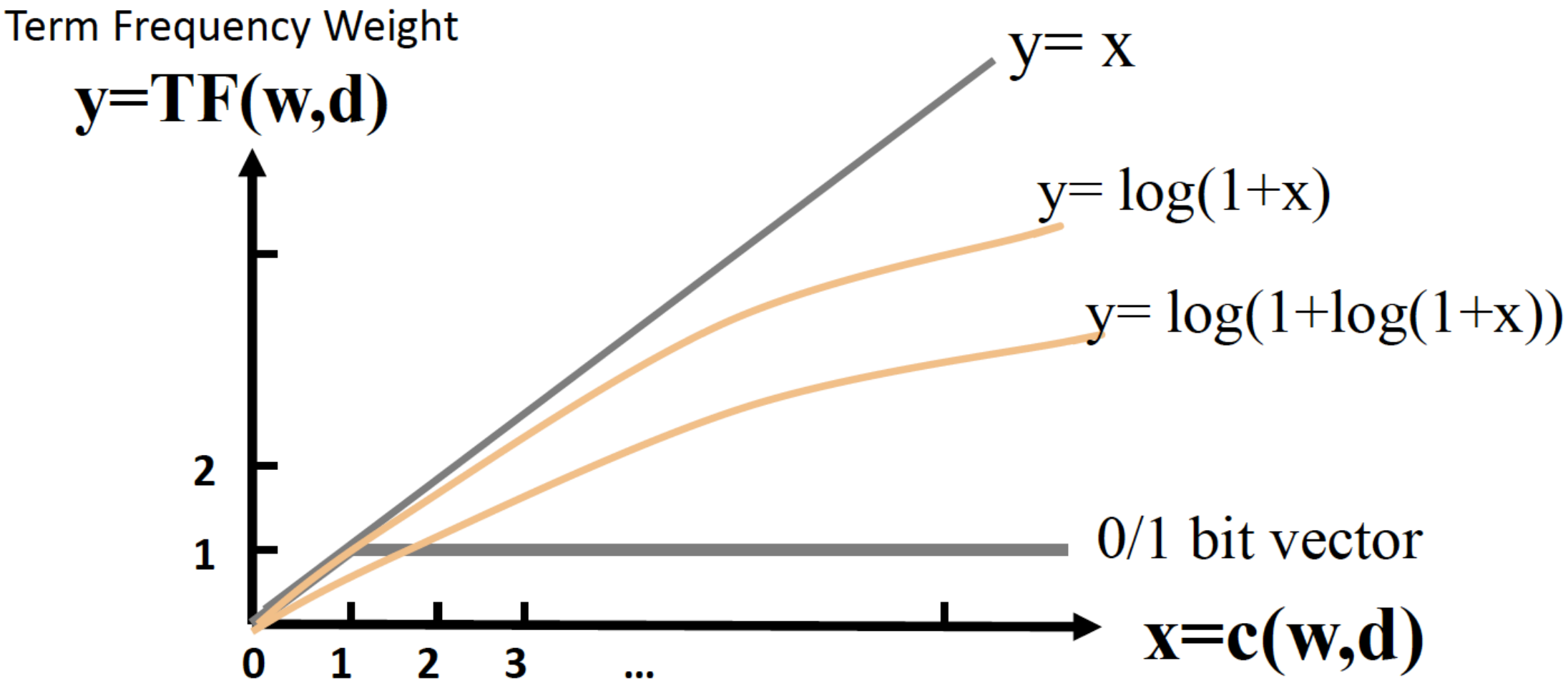
… **news** of organic food **campaign**… **campaign**…**campaign**…**campaign**…

c("campaign",d5)=4

➔f(q,d5)=13.9?

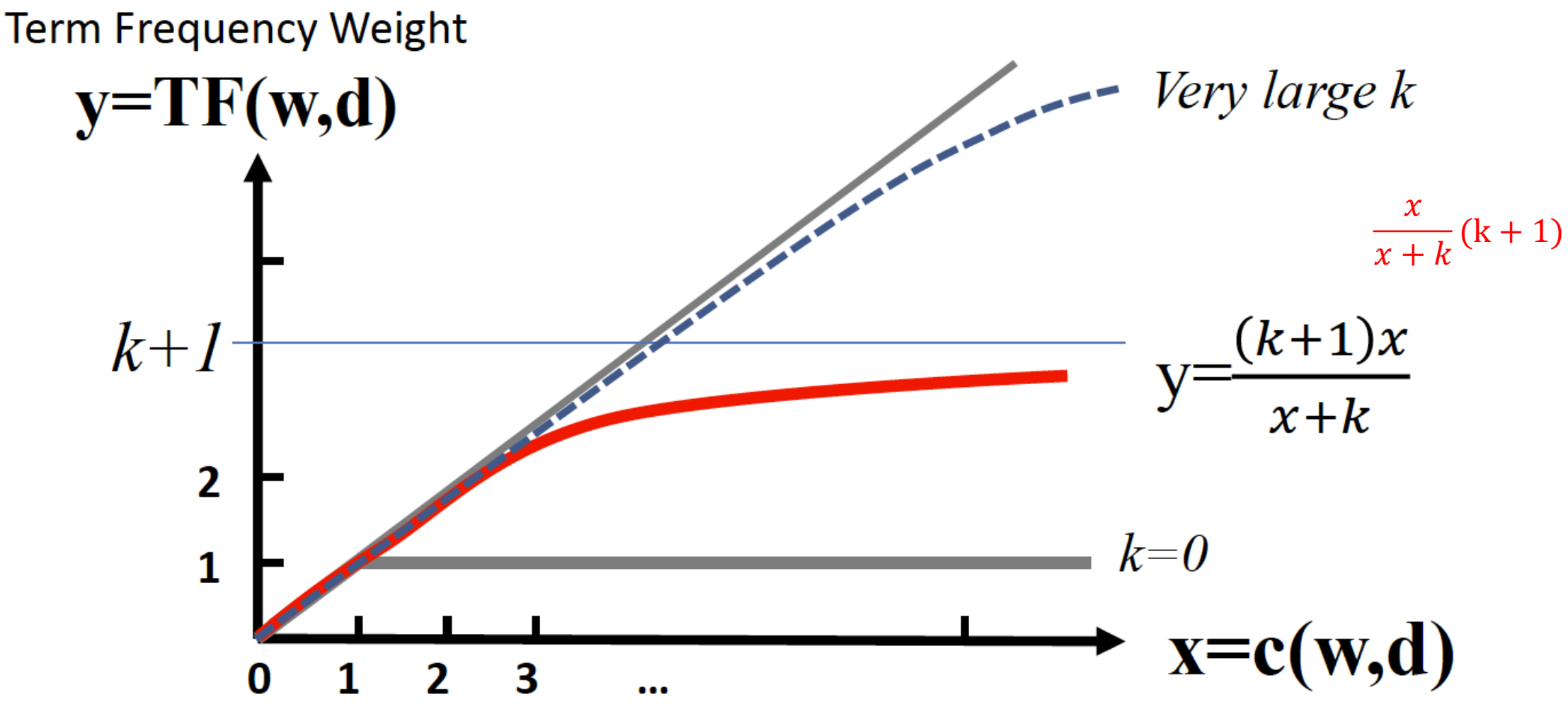
We should not award multiple occurances so generaously and we should restrict the contribution of the high-count term.

TF Transformation: c(w,d)➔TF(w,d)



We can have a sublinear transformation like the orange line here.

TF Transformation: BM25 Transformation



- In fraction (x/x+k) Numerator is always less than than denominator so it will always be less than 1
- Uper bound is usefull to control the influence of a particular term. And it will ensure that all terms will be counted when we aggregate the weights to compute a score.

Summary

- Sublinear TF Transformation is needed to
  - capture the intuition of "diminishing return" from higher TF
  - avoid dominance by one single term over all others
- BM25 Transformation
  - has an upper bound
  - is robust and effective
- Ranking function with BM25 TF (k >=0)

$$f(q,d) = \sum_{i=1}^{N} x_i y_i = \sum_{w \in q \cap d} c(w,q) \frac{(k+1)c(w,d)}{c(w,d)+k} \log \frac{M+1}{df(w)}$$