Vector Space Model

Presenter

Quách Đình Hoàng

Slides are obtained from ChengXiang Zhai and Sean Massung book

Outline

1. Text Retrieval Problem
2. Vector Space Model
   i. Vector Space Framework
   ii. Simplest VSM instantiation
   iii. Improved VSM

1. Text Retrieval Problem

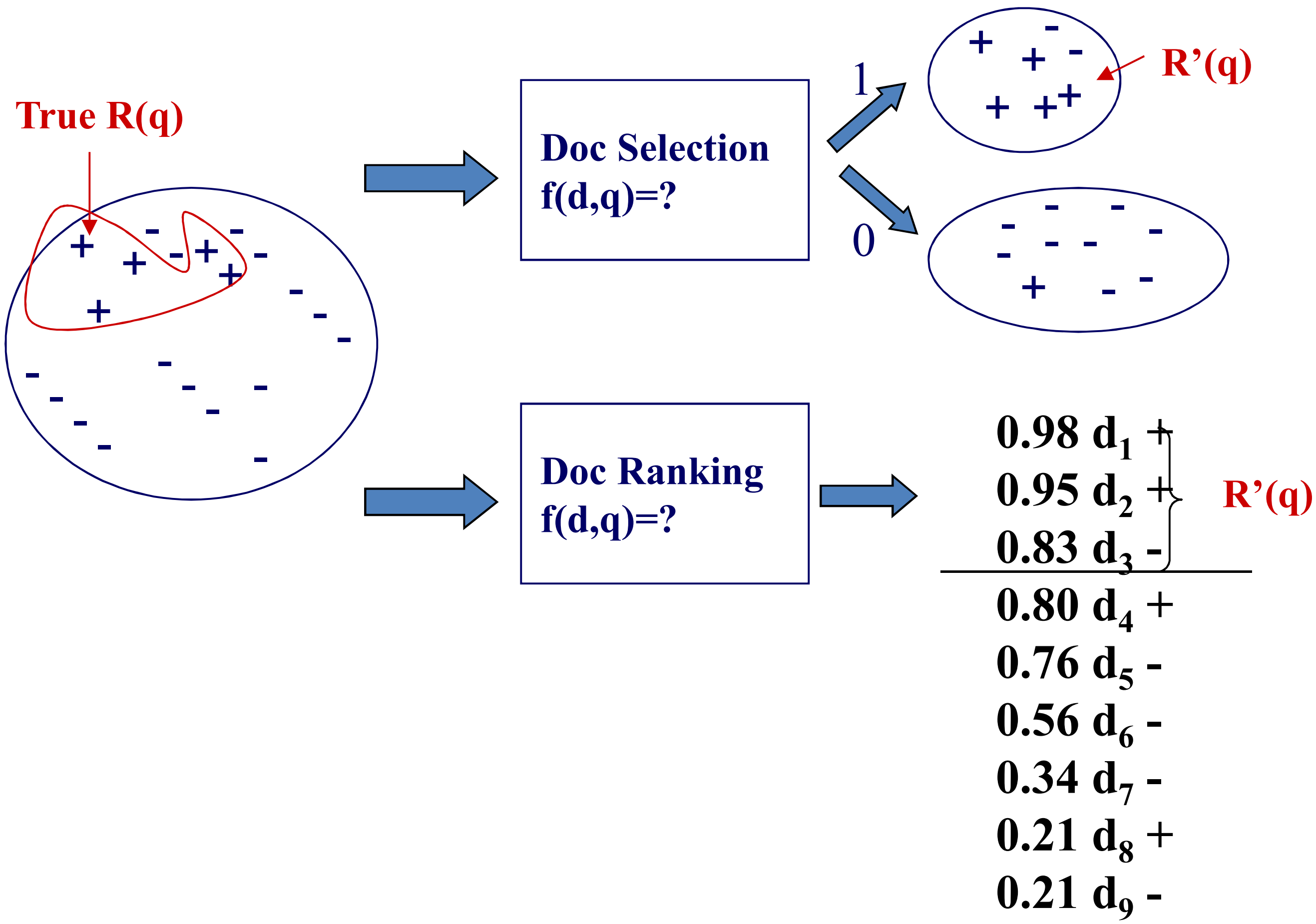
Text Retrieval Problem

- **Query:** q = q1,…,qm, where qi $\in$ V
- **Document:** d = d1,…,dn, where di $\in$ V
- **Ranking function:** f(q, d) $\in \Re$
- A good ranking function should rank relevant documents on top of non-relevant ones
- **Key challenge**: how to measure the likelihood that document d is relevant to query q
- **Retrieval model** = formalization of relevance (give a computational definition of relevance)

Computing $R(q)$

- Strategy 1: Document selection
  - $R(q) = \{d \in C \mid f(d,q) = 1\}$, where $f(d,q) \in \{0,1\}$ is an indicator function or classifier
  - System must decide if a doc is relevant or not ("***absolute relevance***")
- Strategy 2: Document ranking
  - $R(q) = \{d \in C \mid f(d,q) > \theta\}$, where $f(d,q) \in \Re$ is a relevance measure function; θ is a cutoff
  - System must decide if one doc is more likely to be relevant than another ("***relative relevance***")

Document Selection vs. Ranking



Retrieval Models

- **Similarity-based models**: f(q,d) = similarity(q,d)
  - Vector space model
- **Probabilistic models**: f(d,q) = p(R=1|d,q), where $R \in \{0,1\}$
  - Classic probabilistic model
  - Language model
  - Divergence-from-randomness model
- **Probabilistic inference model**: f(q,d) = p(d→q)
- **Axiomatic model**: f(q,d) must satisfy a set of constraints
- These different models tend to result in similar ranking functions involving similar variables

Common Form of a Retrieval Function

$f(q = \text{"presidential campaign news"}, \boxed{d})$

"Bag of Words"

$g(\text{"presidential"}, \boxed{d})$ $g(\text{"campaign"}, \boxed{d})$ $g(\text{"news"}, \boxed{d})$

How many times does "presidential" occur in d?
Term frequency (TF): $c(\text{"presidential"}, d)$

How long is d? **Document length:** $|d|$

How often do we see "presidential" in the entire collection?
Document frequency: DF("presidential")
P("presidential"|collection)

Which Model Works the Best?

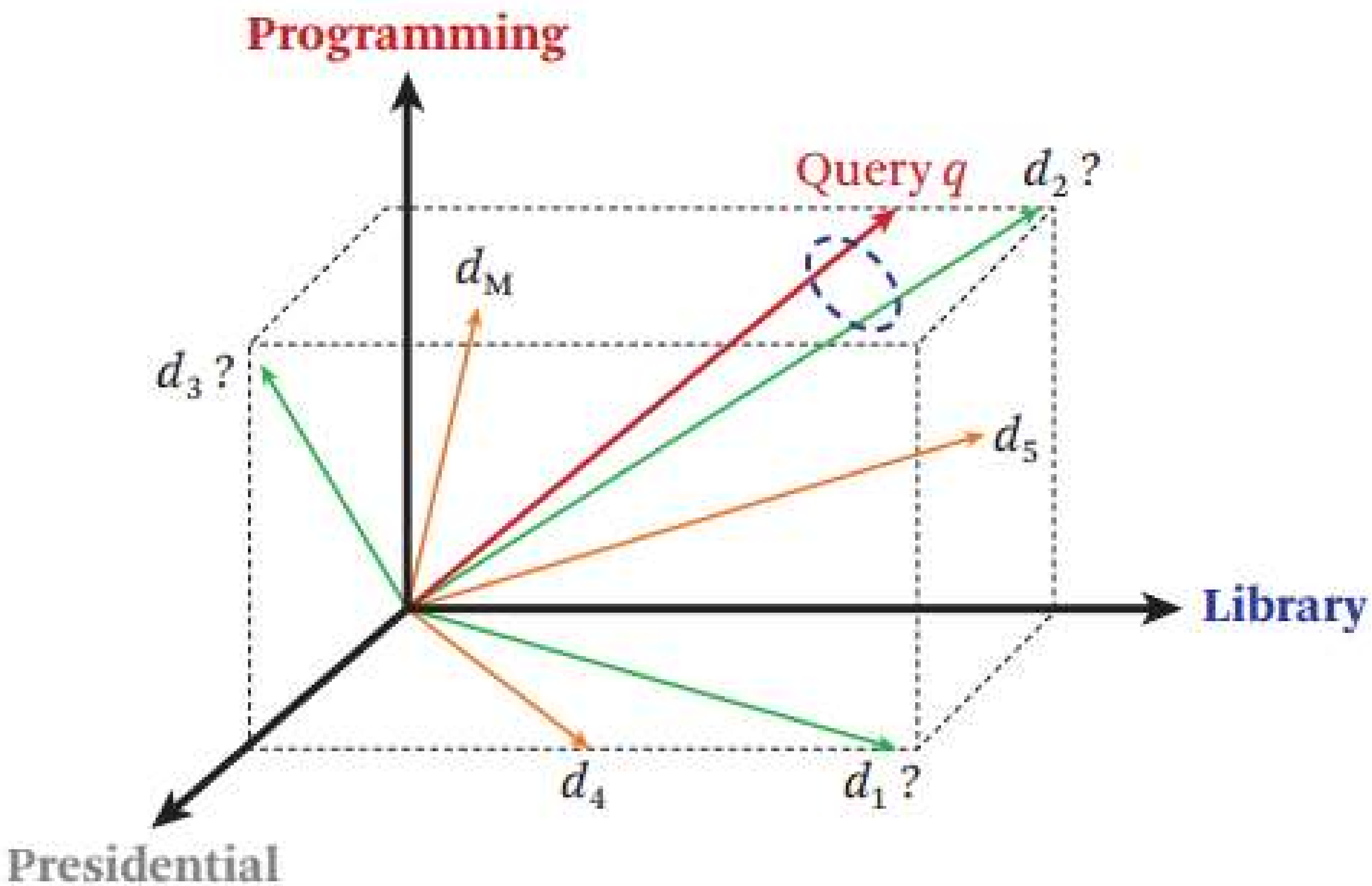
- When optimized, the following models tend to perform equally well [Fang et al., 2011]:
  - **Pivoted length normalization** [Singhal et al., 1996]
  - **BM25** [Robertson and Zaragoza, 1994]
  - **Query likelihood** [Ponte and Croft, 1998]
  - **PL2** [Amati and Van Rijsbergen, 2002]
- BM25 is most popular

2. Vector Space Model

Vector Space Model (VSM)

- Represent a document/query by a ***term vector***
  - ***Term***: basic concept, e.g., ***word*** or ***phrase***
  - Each term defines one dimension
  - N terms define a high-dimensional space
  - Element of vector corresponds to term weight
  - E.g., $d = (x_1,\ldots,x_N)$, x_i is "importance" of term i
- Measure relevance by the distance between the query vector and document vector in the vector space

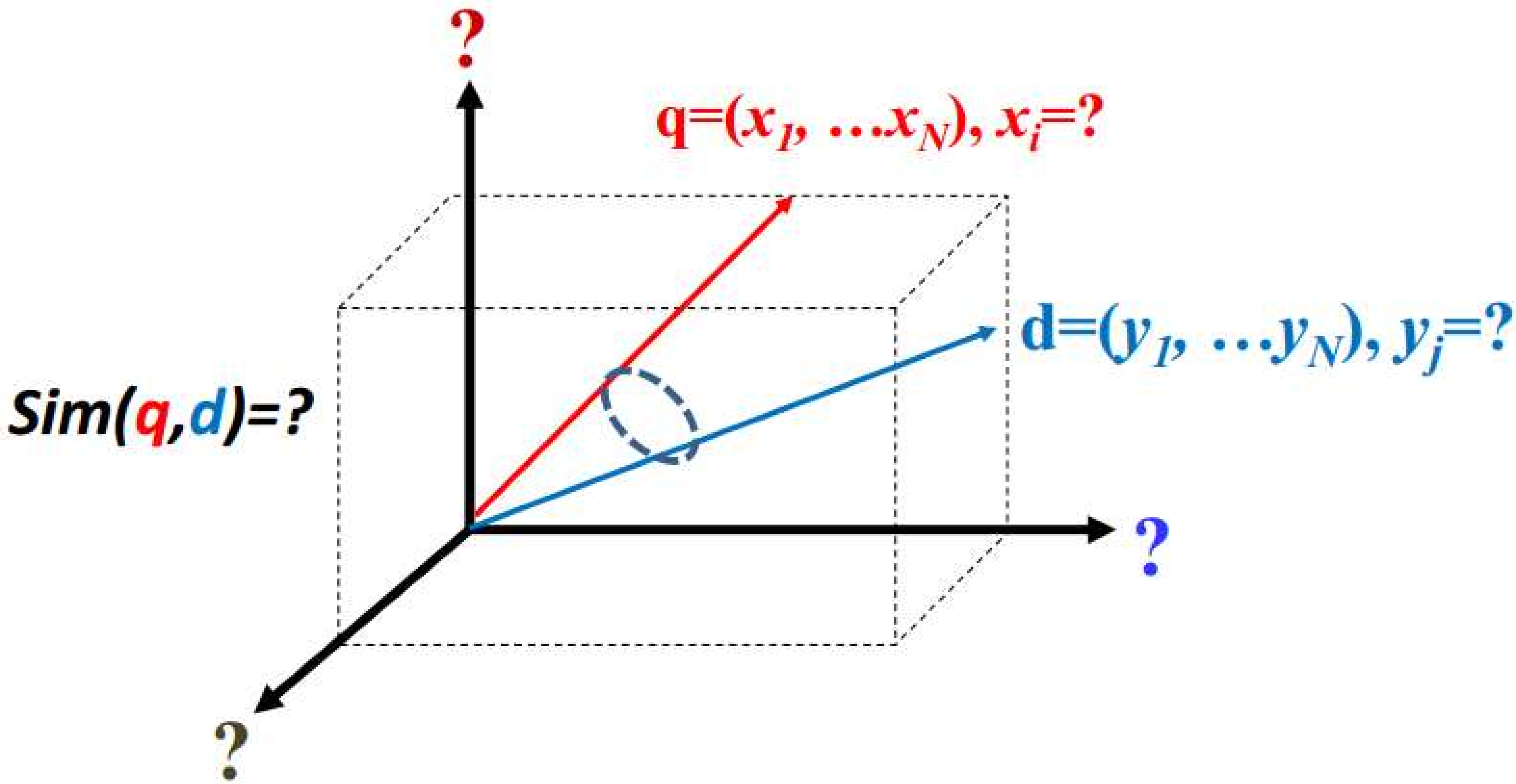
Vector Space Model (VSM) Illustration



VSM is a Framework

- How to define/select the terms
  - Terms are assumed to be linearly independent
- How to assign term weights
  - Weight in query indicates importance of term
  - Weight in doc indicates how well the term characterizes the doc
- How to define the similarity/distance measure

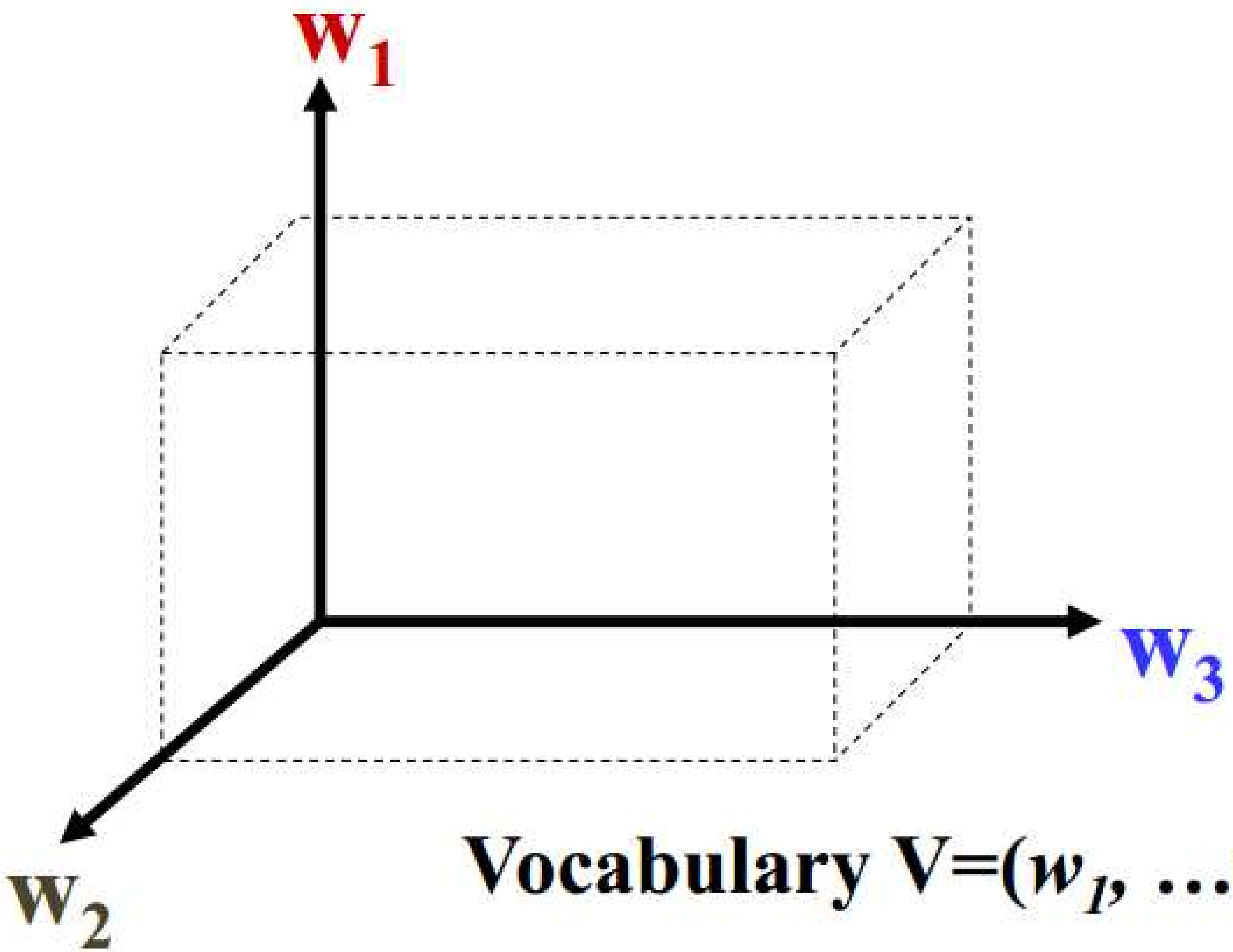
What VSM Doesn't Say



Simplest VSM instantiation

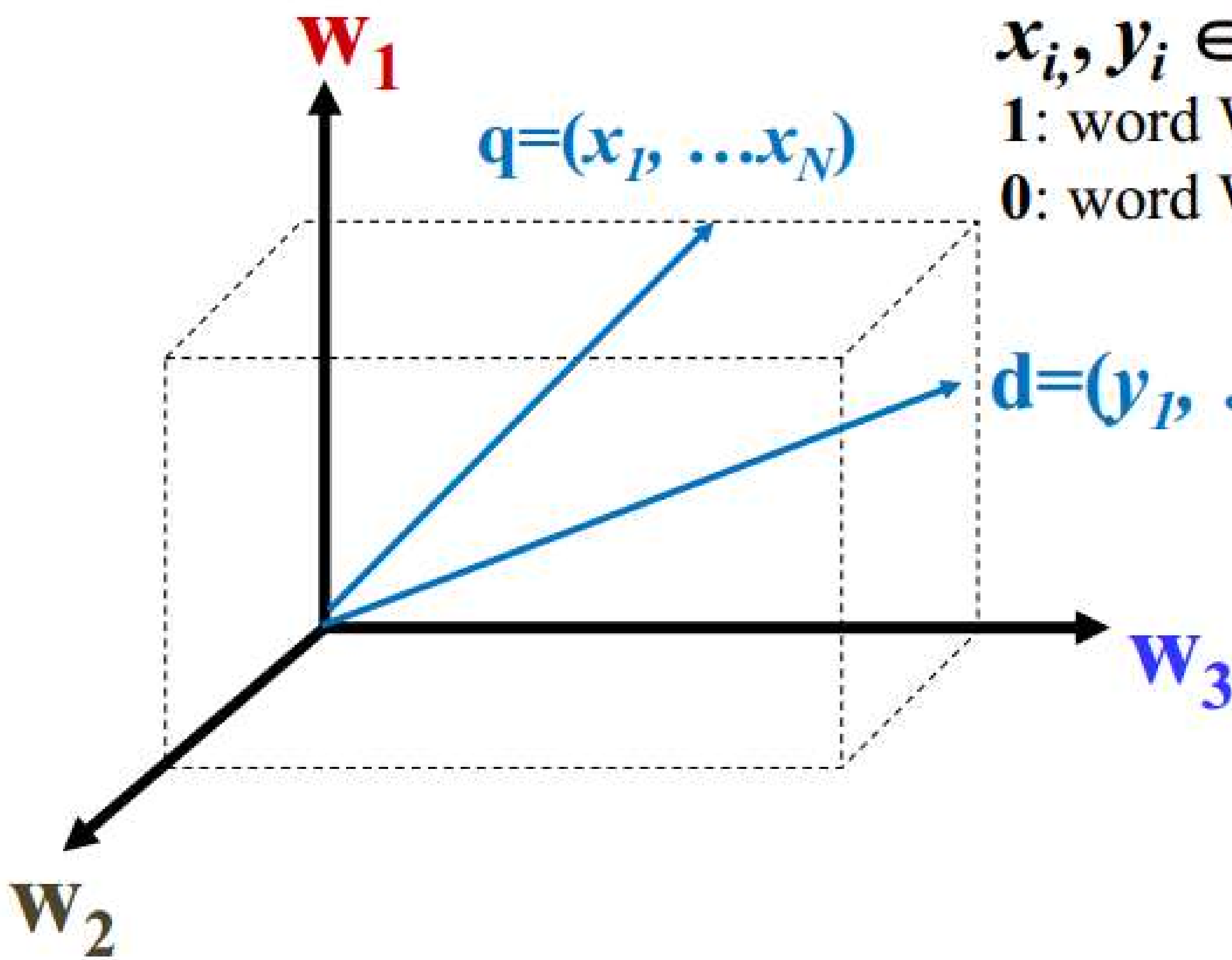
- Dimension = word
- Term weight = 0-1 bit vector (word presence/absence)
- Similarity = dot product
- f(q,d) = number of **distinct** query words matched in d

Dimension Instantiation: Bag of Words (BOW)



Vocabulary $V=(w_1, \ldots w_N)$

Term Weight Instantiation: Bit Vector



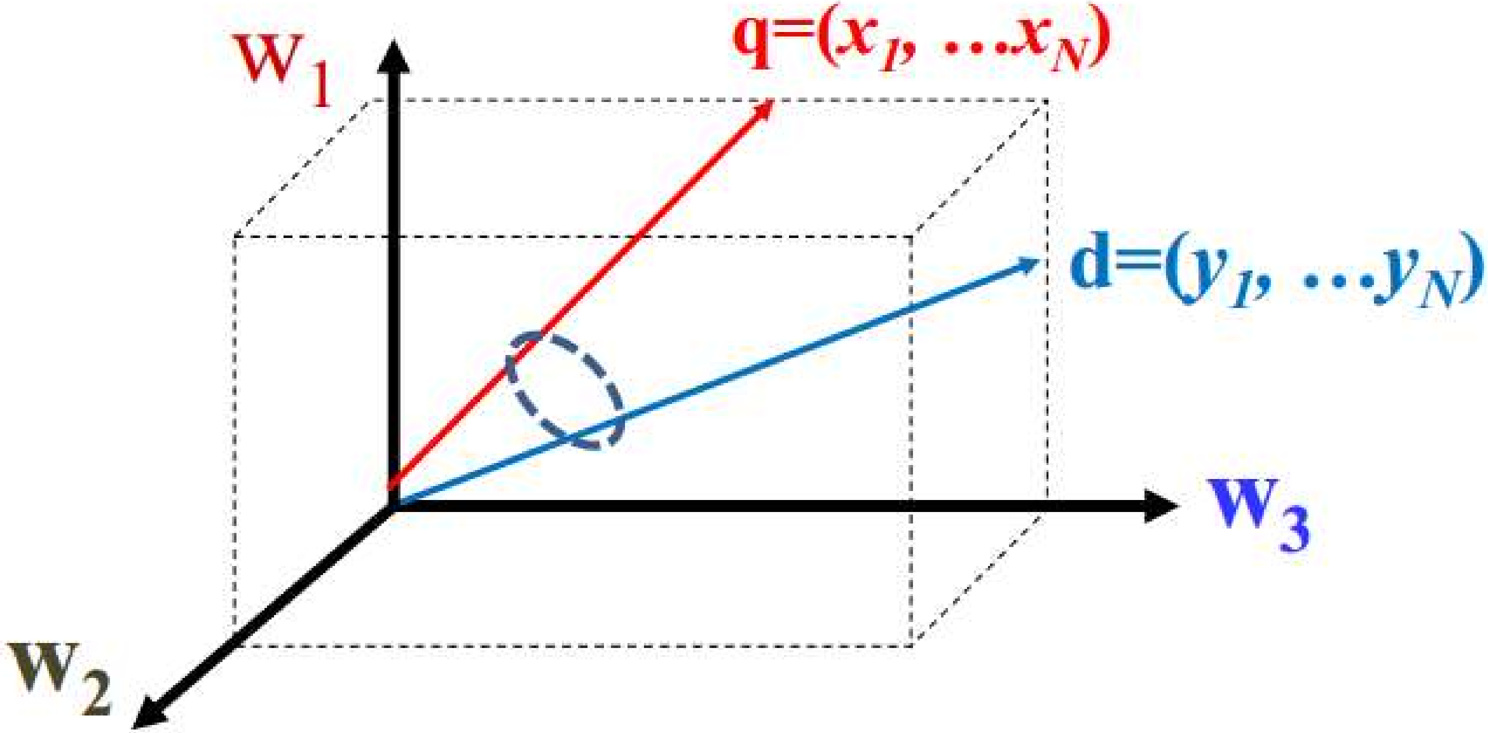
$x_i, y_i \in \{0,1\}$

1: word W_i is **present**

0: word W_i is **absent**

Similarity Instantiation: Dot Product

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



Simplest VSM = BOW + Bit-Vector + Dot-Product

$q = (x_1, \ldots, x_N)$ $\quad x_i, y_i \in \{0, 1\}$

1: word W_i is **present**

$d = (y_1, \ldots, y_N)$ $\quad$ **0**: word W_i is **absent**

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

What does this ranking function intuitively capture?
Is this a good ranking function?

An Example

Query = "news about presidential campaign"

d_1	... news about ...	
d_2	... news about organic food campaign ...	
d_3	... news of presidential campaign ...	
d_4	... news of presidential campaign ... ... presidential candidate ...	
d_5	... news of organic food campaign ... campaign ... campaign ... campaign ...	

Ideal ranking?

d_4 +
d_3 +

d_1 −
d_2 −
d_5 −

Ranking Using the Simplest VSM

Query = "news about presidential campaign"

d_1 ... news about ...

d_3 ... news of presidential campaign ...

$$
\begin{array}{rllllll}
V = & \{\text{news}, & \text{about}, & \text{presidential}, & \text{campaign}, & \text{food} & \ldots\} \\
q = & (1, & 1, & 1, & 1, & 0, & \ldots) \\
d_1 = & (1, & 1, & 0, & 0, & 0, & \ldots)
\end{array}
$$

$$f(q, d_1) = 1 * 1 + 1 * 1 + 1 * 0 + 1 * 0 + 0 * 0 + \ldots = 2$$

$$
\begin{array}{rllllll}
d_3 = & (1, & 0, & 1, & 1, & 0, & \ldots)
\end{array}
$$

$$f(q, d_3) = 1 * 1 + 1 * 0 + 1 * 1 + 1 * 1 + 0 * 0 + \ldots = 3$$

Is the Simplest VSM Effective?

Query = "news about presidential campaign"

d_1	... news about ...	$f(q, d_1) = 2$
d_2	... news about organic food campaign ...	$f(q, d_2) = 3$
d_3	... news of presidential campaign ...	$f(q, d_3) = 3$
d_4	... news of presidential campaign ... ... presidential candidate ...	$f(q, d_4) = 3$
d_5	... news of organic food campaign ... campaign ... campaign ... campaign ...	$f(q, d_5) = 2$

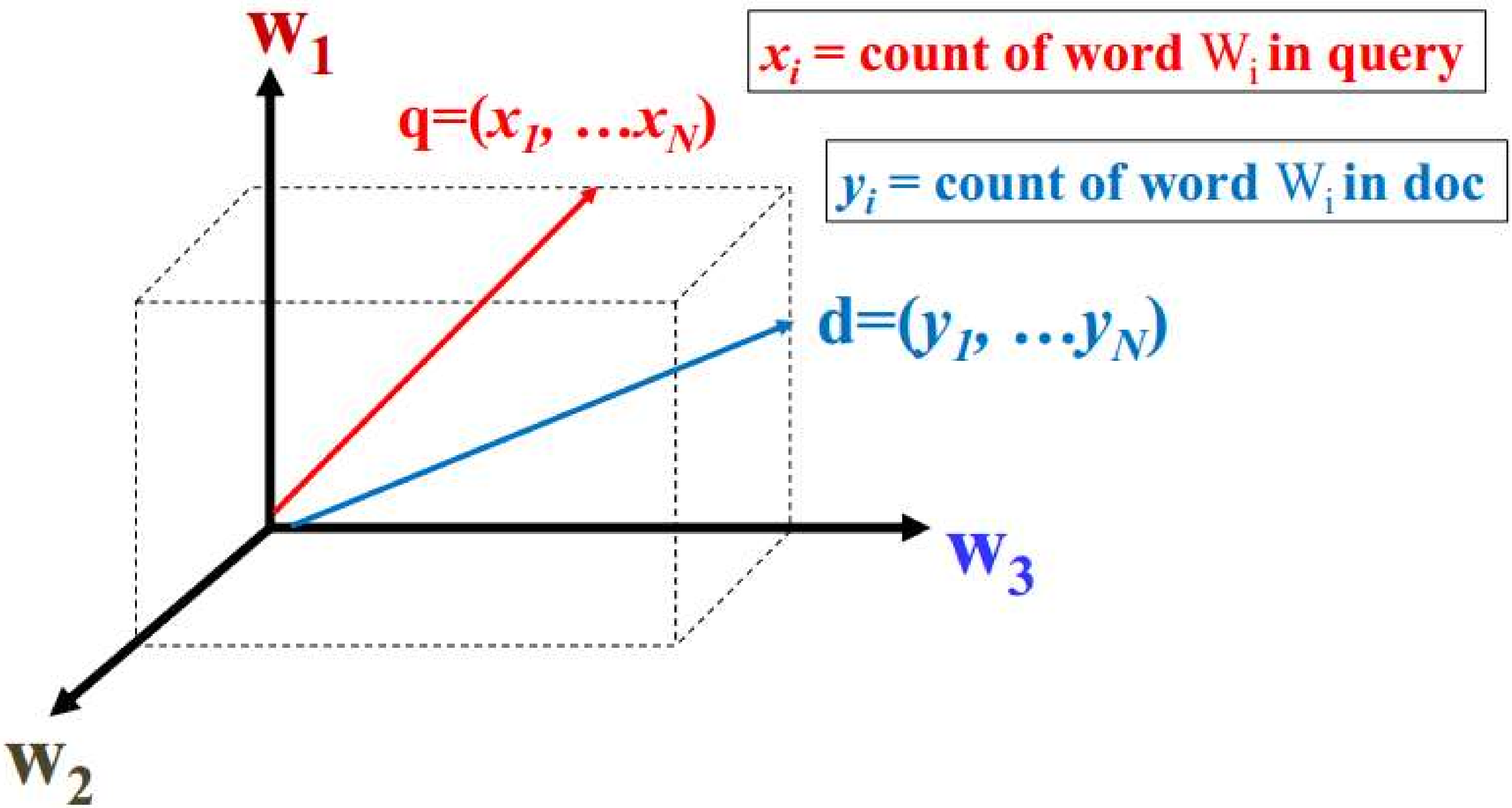
Problems of the Simplest VSM

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 Term Weighting: Term Frequency



Improved VSM with Term Frequency Weighting

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

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

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

- What does this ranking function intuitively capture?
- Does it fix the problems of the simplest VSM?

Ranking Using Term Frequency (TF) Weighting

d_2 ... news about organic food campaign ... $f(q, d_2) = 3$

$q = (\boxed{1,}\ \boxed{1,}\ 1,\ \boxed{1,}\ 0,\ \ldots)$

$d_2 = (\boxed{1,}\ \boxed{1,}\ 0,\ \boxed{1,}\ 1,\ \ldots)$

d_3 ... news of presidential campaign ... $f(q, d_3) = 3$

$q = (\boxed{1,}\ 1,\ \boxed{1,}\ \boxed{1,}\ 0,\ \ldots)$

$d_3 = (\boxed{1,}\ 0,\ \boxed{1,}\ \boxed{1,}\ 0,\ \ldots)$

d_4 ... news of presidential campaign ... ... presidential candidate ... $f(q, d_4) = 4!$

$q = (\boxed{1,}\ 1,\ \boxed{1,}\ \boxed{1,}\ 0,\ \ldots)$

$d_4 = (\boxed{1,}\ 0,\ \boxed{2,}\ \boxed{1,}\ 0,\ \ldots)$

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

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

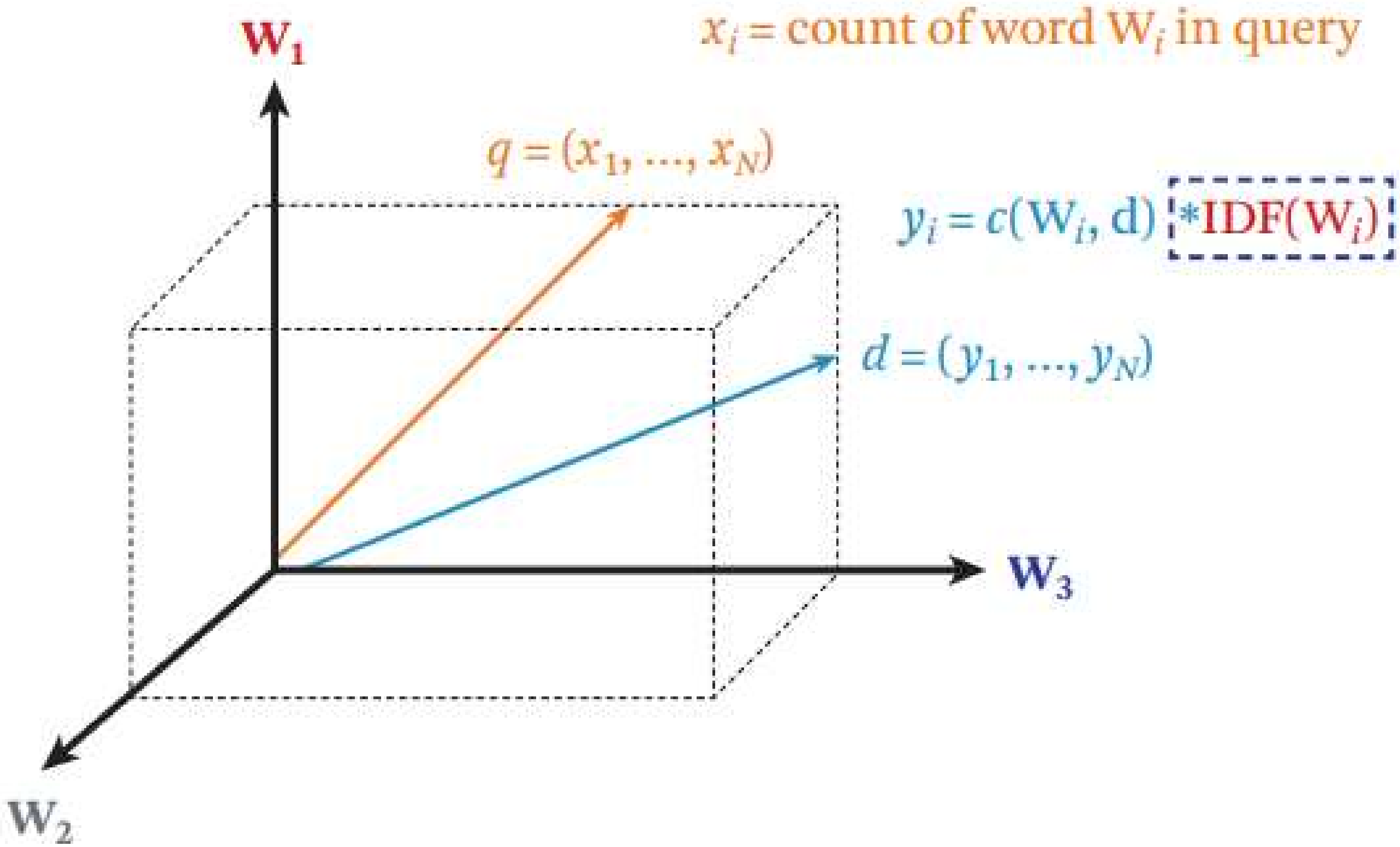
d3 … **news** of **presidential campaign** …

V=	{news,	about,	presidential,	campaign,	food …. }	
q=	**(1,**	**1,**	**1,**	**1,**	**0,**	**…)**
d2=	**(1,**	**1,**	**0,**	**1,**	**1,**	**…)**
q=	**(1,**	**1,**	**1,**	**1,**	**0,**	**…)**
d3=	**(1,**	**0,**	**1,**	**1,**	**0,**	**…)**

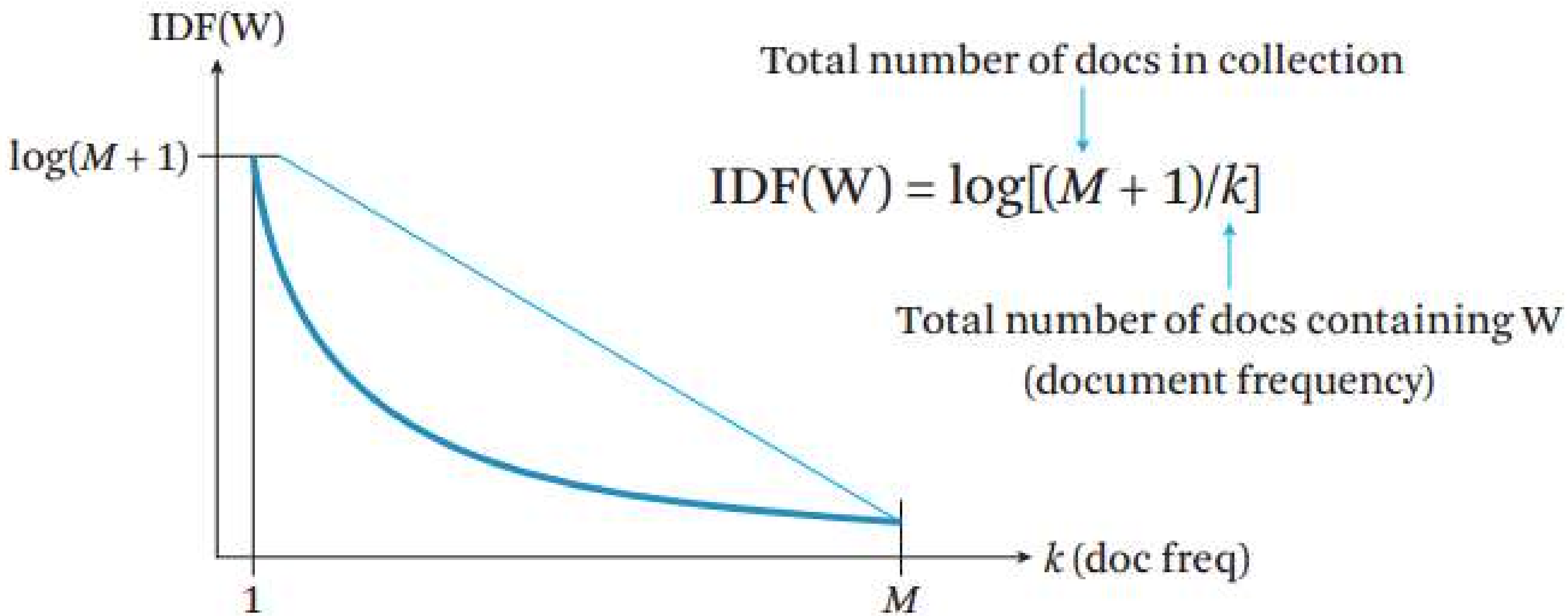
f(q,d2)<3

f(q,d3)>3

Further Improvement of Term Weighting: Adding Inverse Document Frequency (IDF)



IDF Weighting: Penalizing Popular Terms



The impact of IDF weighting on document ranking

d_2 ... news about organic food campaign ...

d_3 ... 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,	...)
$d_2 =$	(1 * 1.5,	1 * 1.0,	0,	1 * 3.1,	0,	...)
$q =$	(1,	1,	1,	1,	0,	...)
$d_3 =$	(1 * 1.5,	0,	1 * 2.5,	1 * 3.1,	0,	...)

$$f(q, d_2) = 5.6 \quad < \quad f(q, d_3) = 7.1$$

Scores of all documents using TF-IDF weighting

Query = "news about presidential campaign"

d_1	... news about ...	$f(q, d_1) = 2.5$
d_2	... news about organic food campaign ...	$f(q, d_2) = 5.6$
d_3	... news of presidential campaign ...	$f(q, d_3) = 7.1$
d_4	... news of presidential campaign ... ... presidential candidate ...	$f(q, d_4) = 9.6$
d_5	... news of organic food campaign ... campaign ... campaign ... campaign ...	$\boldsymbol{f(q, d_5) = 13.9}$**!**

Improved VSM

- Dimension = word
- Term weighting = TF-IDF
- Similarity = dot product
- Working better than the simplest VSM
- Still having problems

Ranking function using a TF-IDF weighting scheme

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

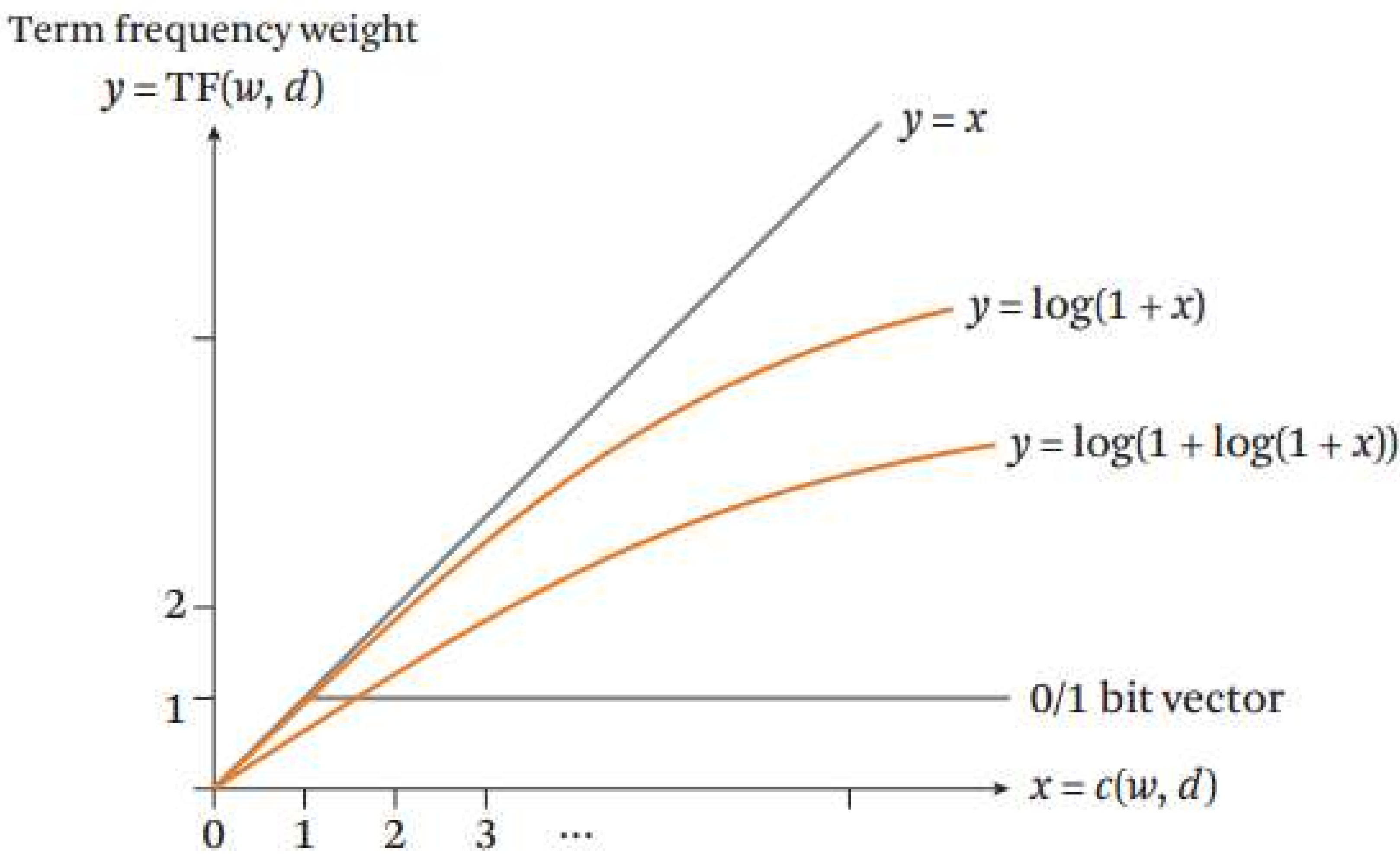
Doc Frequency

d5

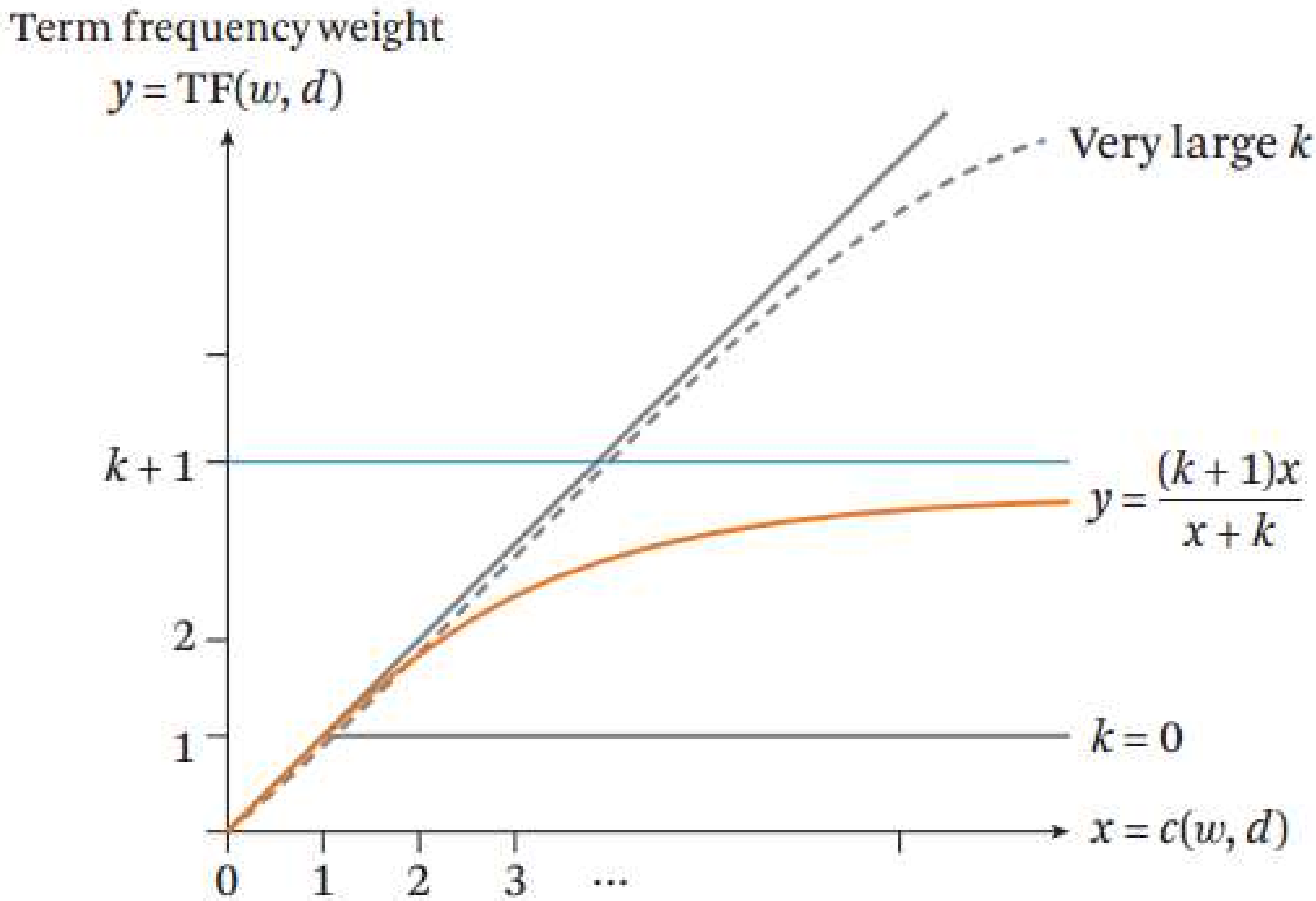
… **news** of organic food **campaign**…
campaign…**campaign**…**campaign**…

c("campaign",d5)=4
➔f(q,d5)=13.9?

TF Transformation: c(w,d) → TF(w,d)



TF Transformation: BM25 Transformation

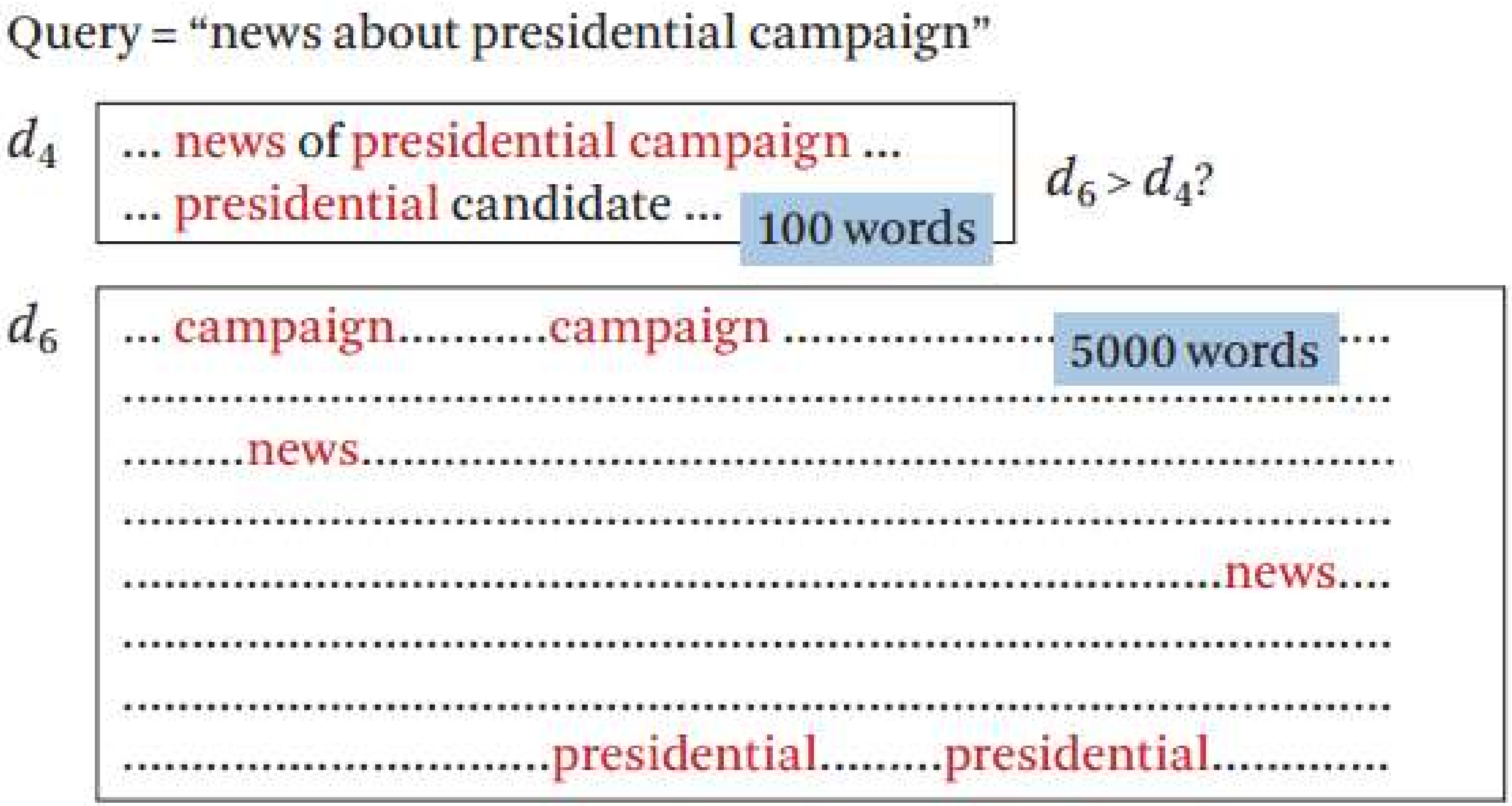


Ranking function with BM25 TF

- BM25 Transformation
  - has an upper bound
  - is robust and effective

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

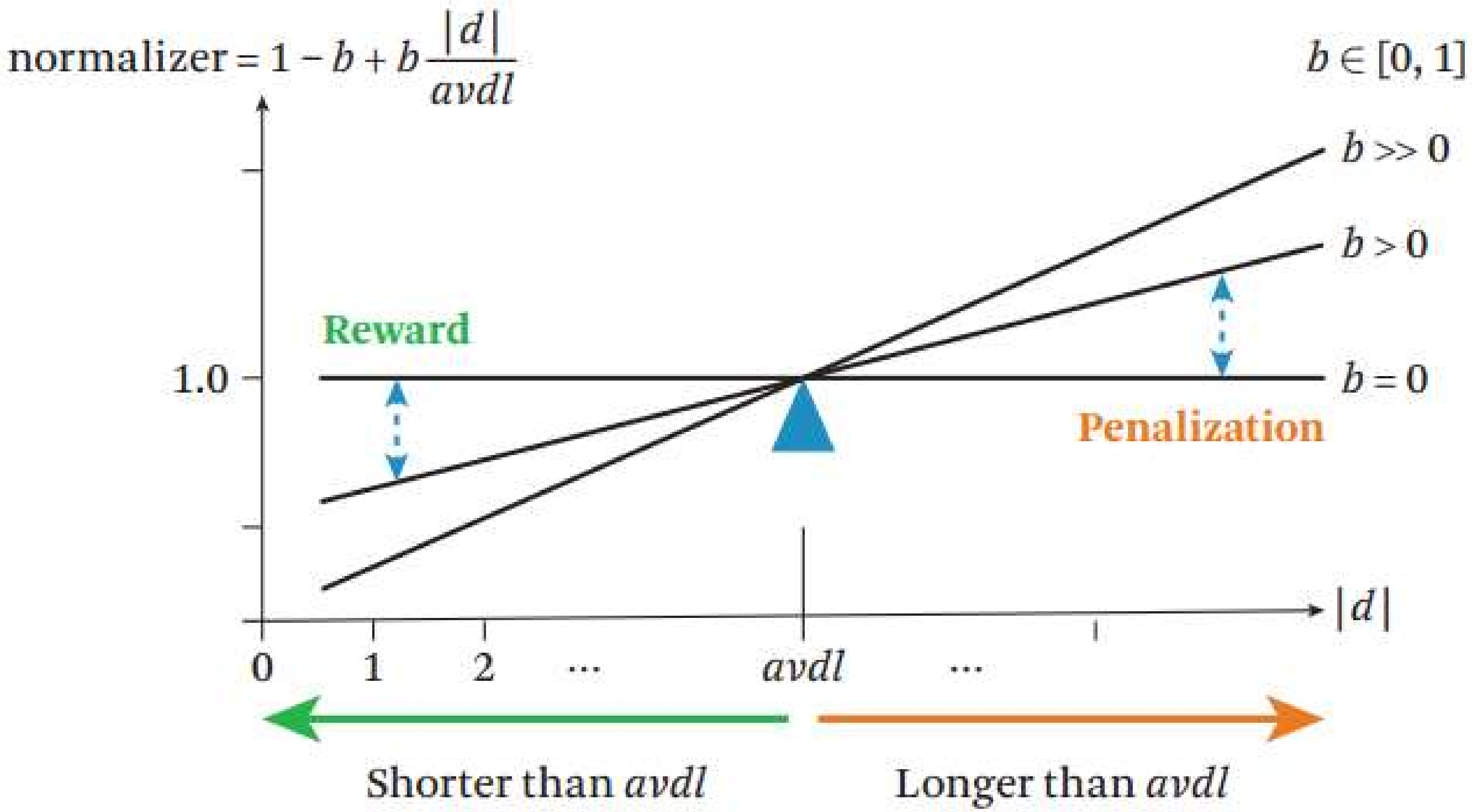
Document Length Normalization



Document Length Normalization

- Penalize a long doc with a doc length normalizer
  - Long doc has a better chance to match any query
  - Need to avoid over-penalization
- A document is long because
  - it uses more words → more penalization
  - it has more contents → less penalization
- Pivoted length normalizer: average doc length as "pivot"
  - Normalizer = 1 if |d| = average doc length (avdl)

Illustration of pivoted document length normalization



State of the Art VSM Ranking Functions

Pivoted length normalization VSM

$$f(q,d) = \sum_{w \in q \cap d} c(w,q) \frac{\ln(1+\ln(1+c(w,d)))}{1-b+b\frac{|d|}{avdl}} \log \frac{M+1}{\mathrm{df}(w)} \quad b \in [0,1]$$

BM25/Okapi

$$f(q,d) = \sum_{w \in q \cap d} c(w,q) \frac{(k+1)c(w,d)}{c(w,d)+k(1-b+b\frac{|d|}{avdl})} \log \frac{M+1}{\mathrm{df}(w)} \quad b \in [0,1],\ k \in [0,+\infty)$$

Further Improvement of Basic VSM

- Improved instantiation of **dimension?**
  - Stemmed words, stop word removal, phrases, latent semantic indexing (word clusters), character n-grams, …
  - Bag-of-words with phrases is often sufficient in practice
  - Language-specific and domain-specific tokenization is important to ensure "normalization of terms"
- Improved instantiation of **similarity function?**
  - cosine of angle between two vectors?
  - Euclidean?
  - Dot product seems still the best (sufficiently general especially with appropriate term weighting)

Further Improvement of BM25

- BM25F [Robertson & Zaragoza 09]
  - Use BM25 for documents with structures ("F"=fields)
  - Key idea: combine the frequency counts of terms in all fields and then apply BM25 (instead of the other way)
- BM25+ [Lv & Zhai 11]
  - Address the problem of over penalization of long documents by BM25 by adding a small constant to TF
  - Empirically and **analytically** shown to be better than BM25

Summary of Vector Space Model

- Relevance(q, d) = similarity(q, d)
- Query and documents are represented as vectors
- Heuristic design of ranking function
- Major term weighting heuristics
  – TF weighting and transformation
  – IDF weighting
  – Document length normalization
- BM25 and Pivoted normalization seem to be most effective

References

- ChengXiang Zhai and Sean Massung, *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*, ACM Books, 2016.
  - Chapter 6, Section 6.1-6.3 (Vector space model)

Questions