Retrieval Models: Vector Space

Intelligent Information Retrieval

The Notion of Relevance

Relevance

Relevance constraints
(Fang et al. 04)

$\Delta(Rep(q), Rep(d))$
Similarity

Different rep & similarity

Vector space model
(Salton et al., 75)

. . .

Prob. distr. model
(Wong & Yao, 89)

$P(r = 1|q, d) \quad r \quad \{0, 1\}$
Probability of Relevance

Regression Model
(Fox 83)

Generative Model

Doc generation

Classical prob. Model
(Robertson & Sparck Jones, 76)

Query generation

LM approach
(Ponte & Croft, 98)
(Lafferty & Zhai, 01a)

$P(d \quad q)$ or $P(q \quad d)$
Probabilistic inference

Different inference system

Prob. concept space model
(Wong & Yao, 95)

Inference network model
(Turtle & Croft, 91)

The Basic Question

- Given a query, how do we know if document A is more relevant than B?

One Possible Answer

- If document A uses more query words than document B
- (Word usage in document A is more similar to that in query)

Relevance = Similarity

- Assumptions
  - Query and document are represented similarly
  - A query can be regarded as a "document"
  - $Relevance(d,q) \propto similarity(d,q)$
- $R(q) = \{d \in C | f(d,q) > \theta\}, f(q,d) = \Delta(\boldsymbol{Rep(q)}, \boldsymbol{Rep(d)})$
- Key issues
  - How to represent query/document?
  - How to define the similarity measure Δ?

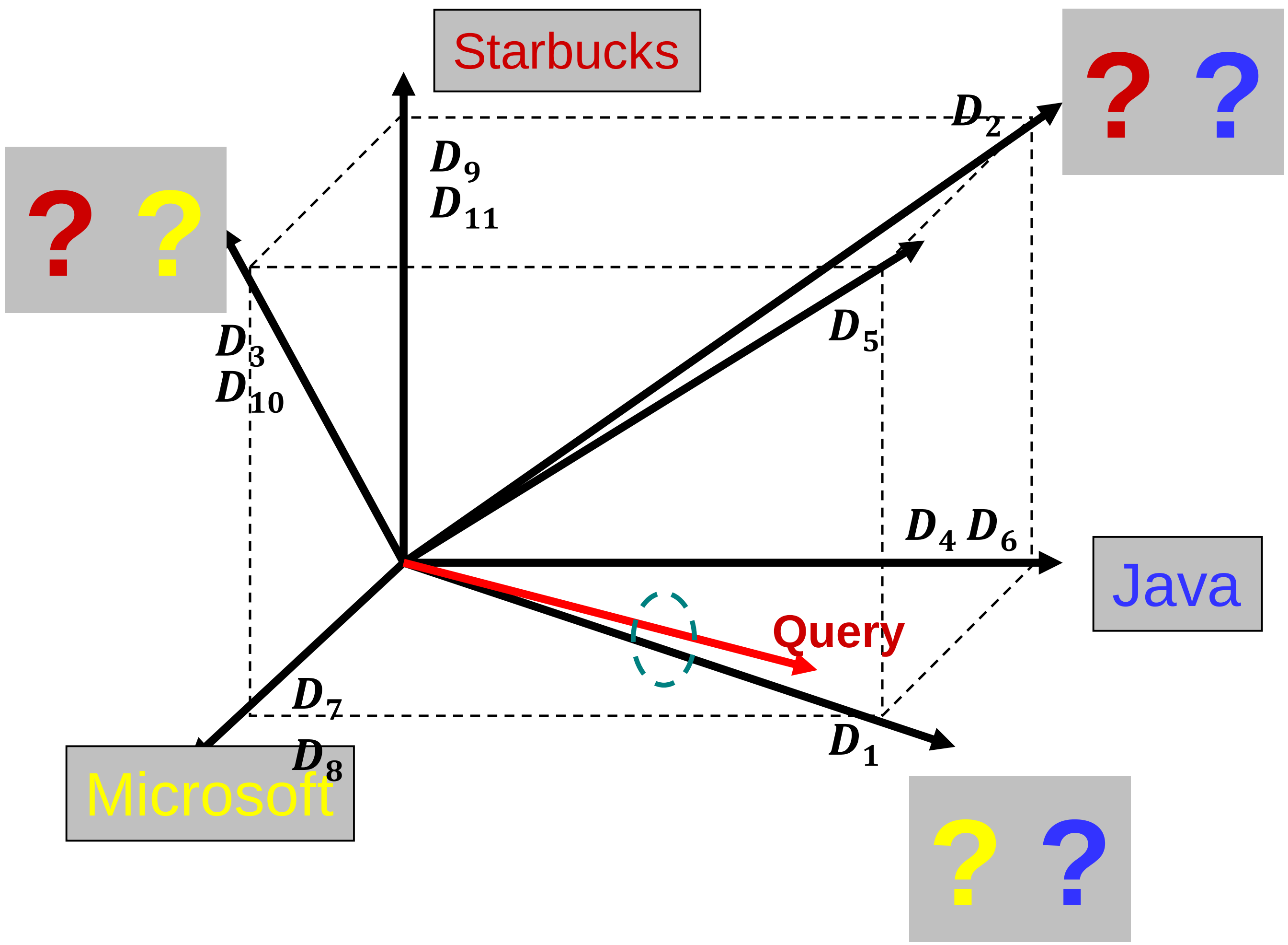
Vector Space Model

- Represent a doc/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

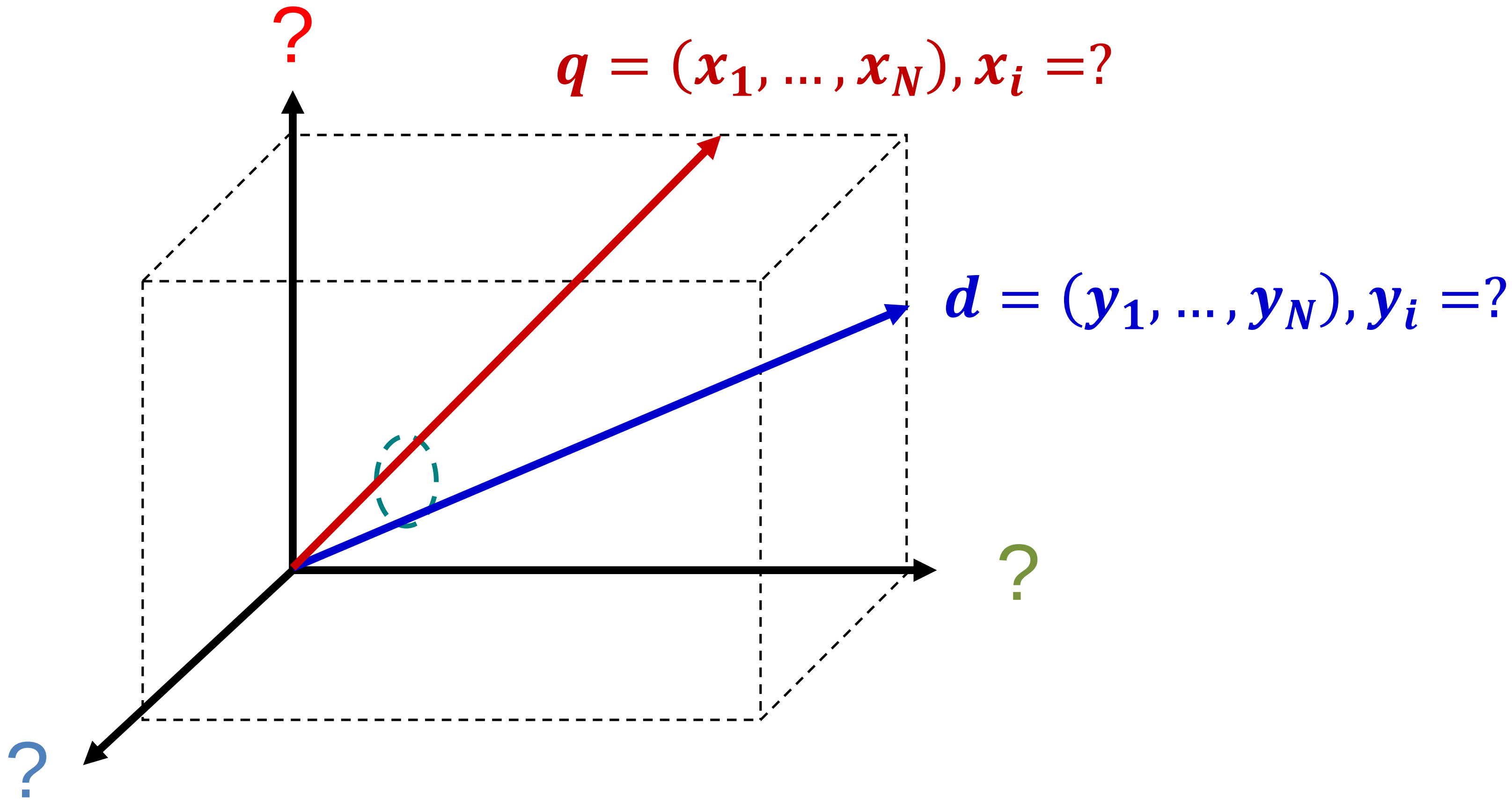
What’s a good “basic concept”?

- Orthogonal
  - Linearly independent basis vectors
  - “Non-overlapping” in meaning
- No ambiguity
- Many possibilities: Words, stemmed words, phrases, “latent concepts”, …
- Single words + short statistical phrases are generally “good enough”

VS Model: illustration



What the VS model doesn't say

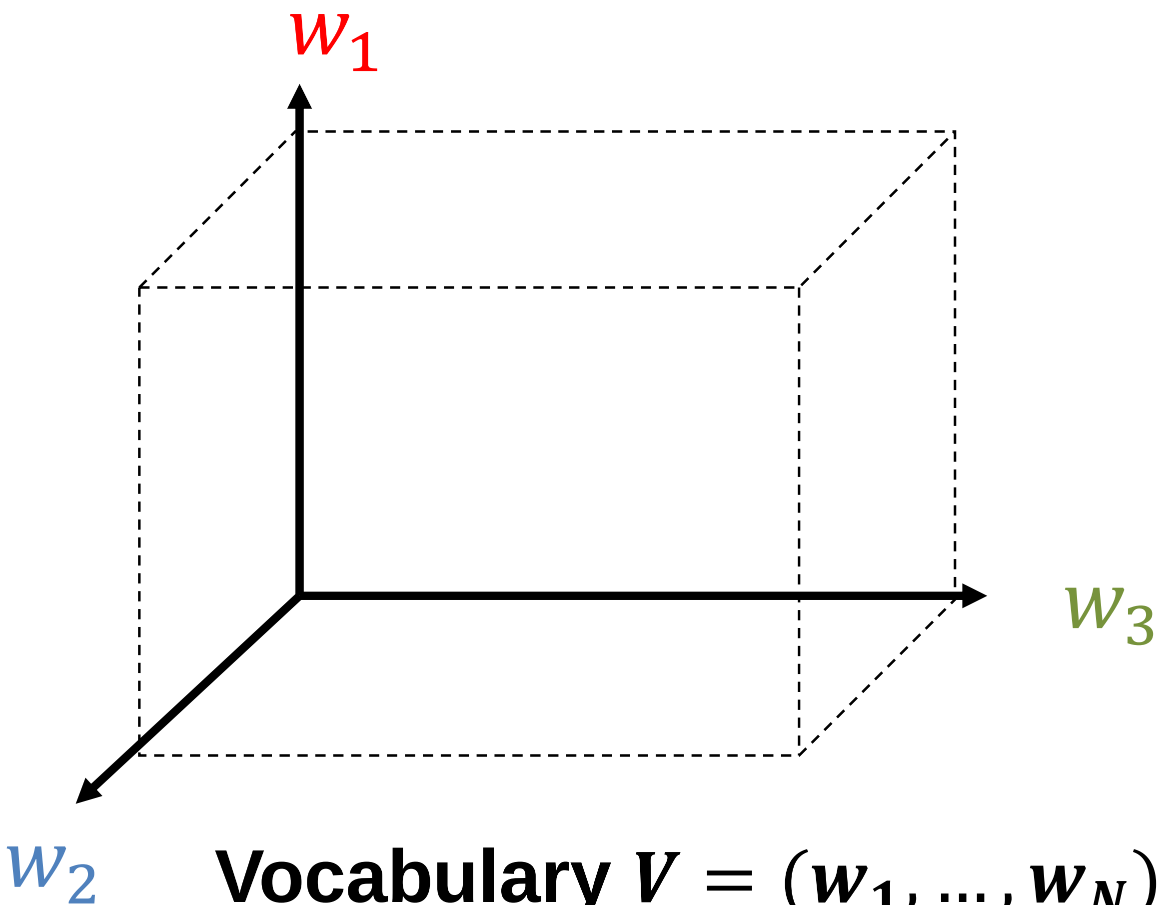


What the VS model doesn't say

- How to define the dimensions (define "basic concepts" or terms)
  - Concepts are assumed to be orthogonal
- How to place queries and documents in the vector space (how to assign 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

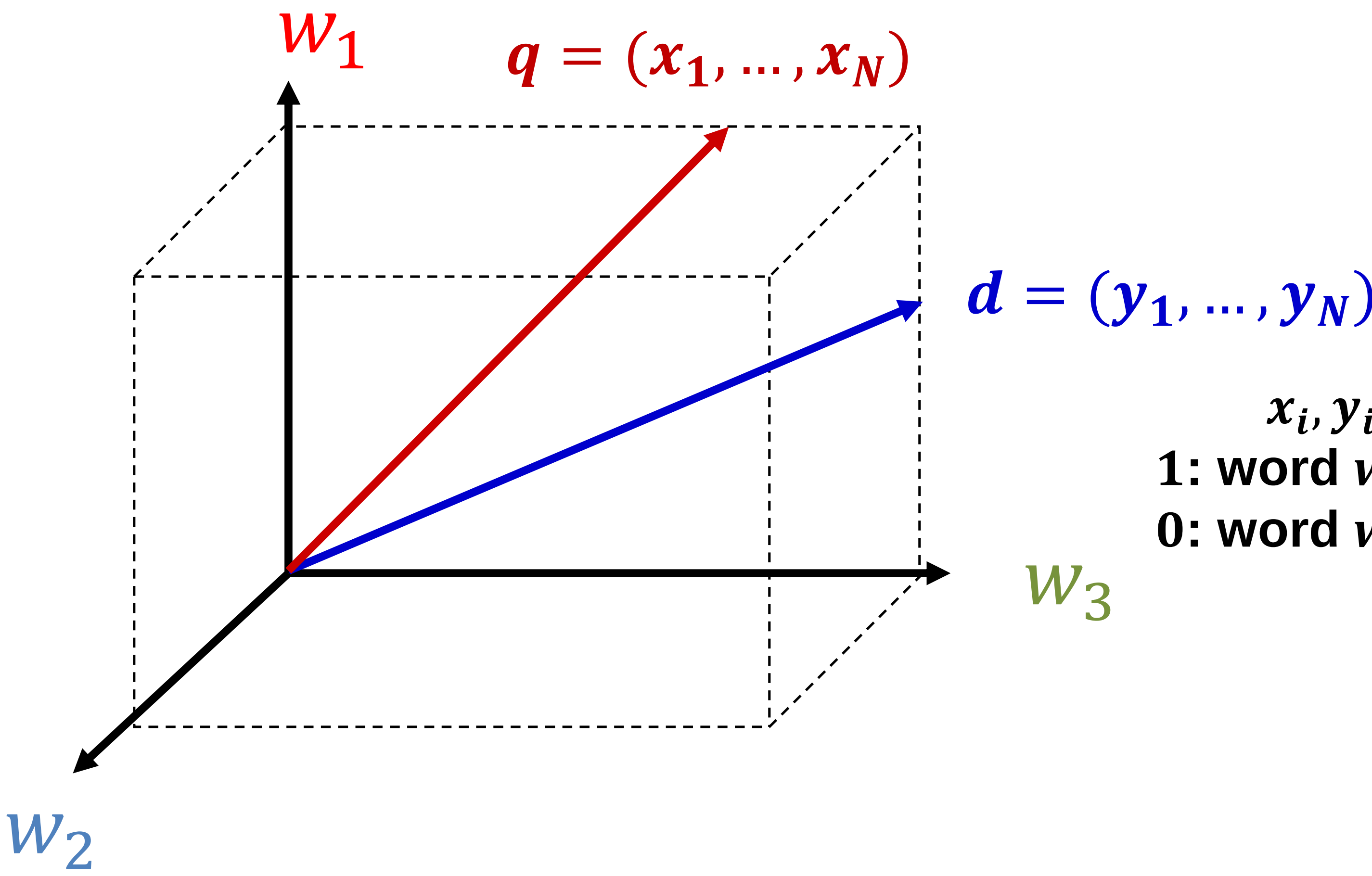
Most research work in VS model tried to address these questions

Dimension Instantiation: Bag of Words (BOW)



Vocabulary $\boldsymbol{V = (w_1, \ldots, w_N)}$

Vector Placement: Bit Vector



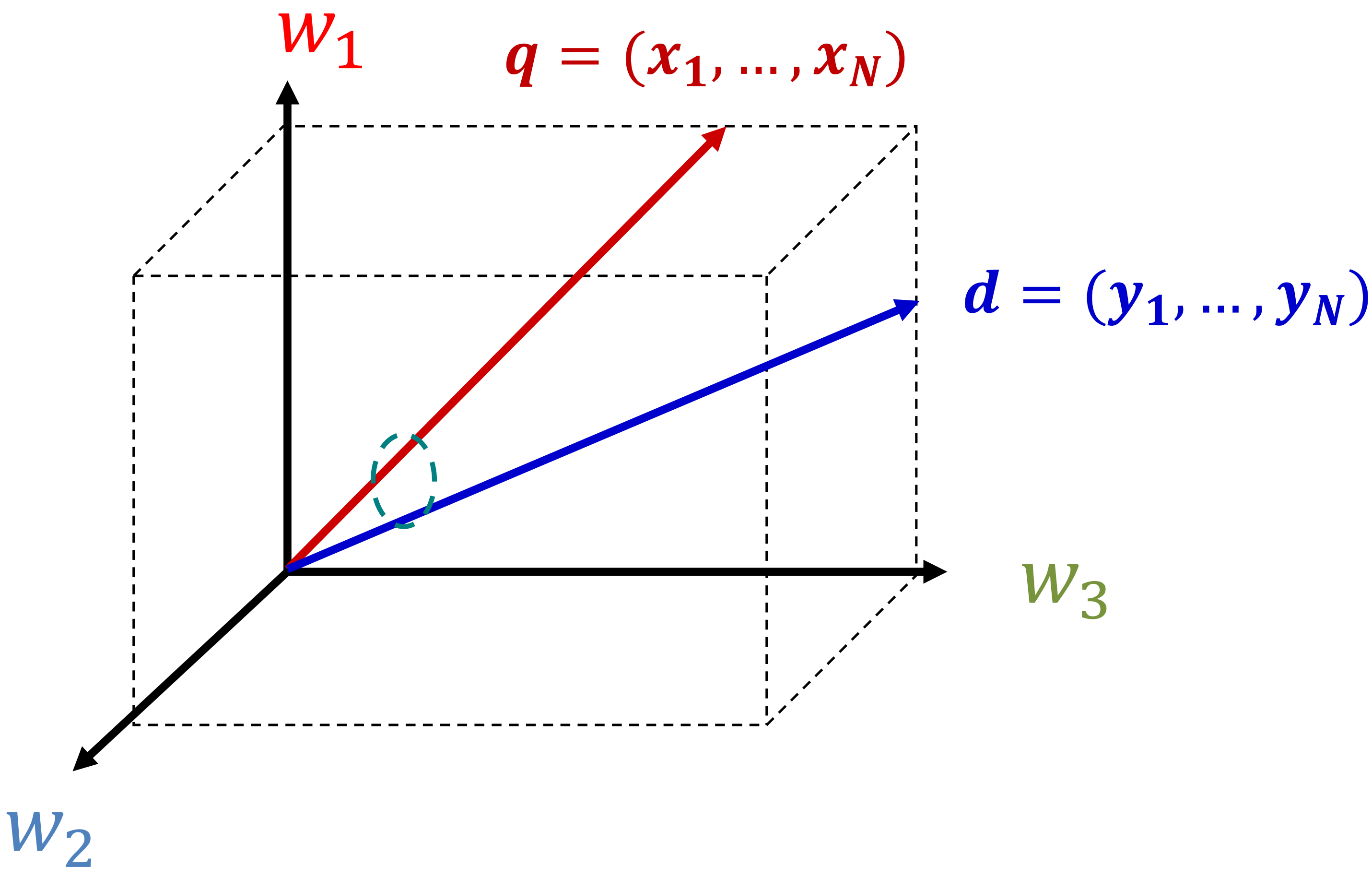
$\boldsymbol{x_i}, \boldsymbol{y_i} \in \{\mathbf{0}, \mathbf{1}\}$

1: word $\boldsymbol{w_i}$ is present

0: word $\boldsymbol{w_i}$ is absent

Similarity Instantiation: Dot Product

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



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

$$q = (x_1, \dots, x_N)$$

$$d = (y_1, \dots, y_N)$$

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

1: word w_i is present

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: How Would You Rank These Documents?

Query = "news about presidential campaign"

Ideal Ranking?

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 …

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 …

$$V = \{news, about, presidential, campaign, food, \ldots\}$$

$$q \;=\; (\,1, \quad 1, \quad 1, \quad 1, \quad 0, \ldots\,)$$

$$d_1 = (\,1, \quad 1, \quad 0, \quad 0, \quad 0, \ldots\,)$$

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

$$d_3 = (\,1, \quad 0, \quad 1, \quad 1, \quad 0, \ldots\,)$$

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

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 …	$f(q, d_5) = 2$

Two Problems of the Simplest VSM

Query = "news about presidential campaign"

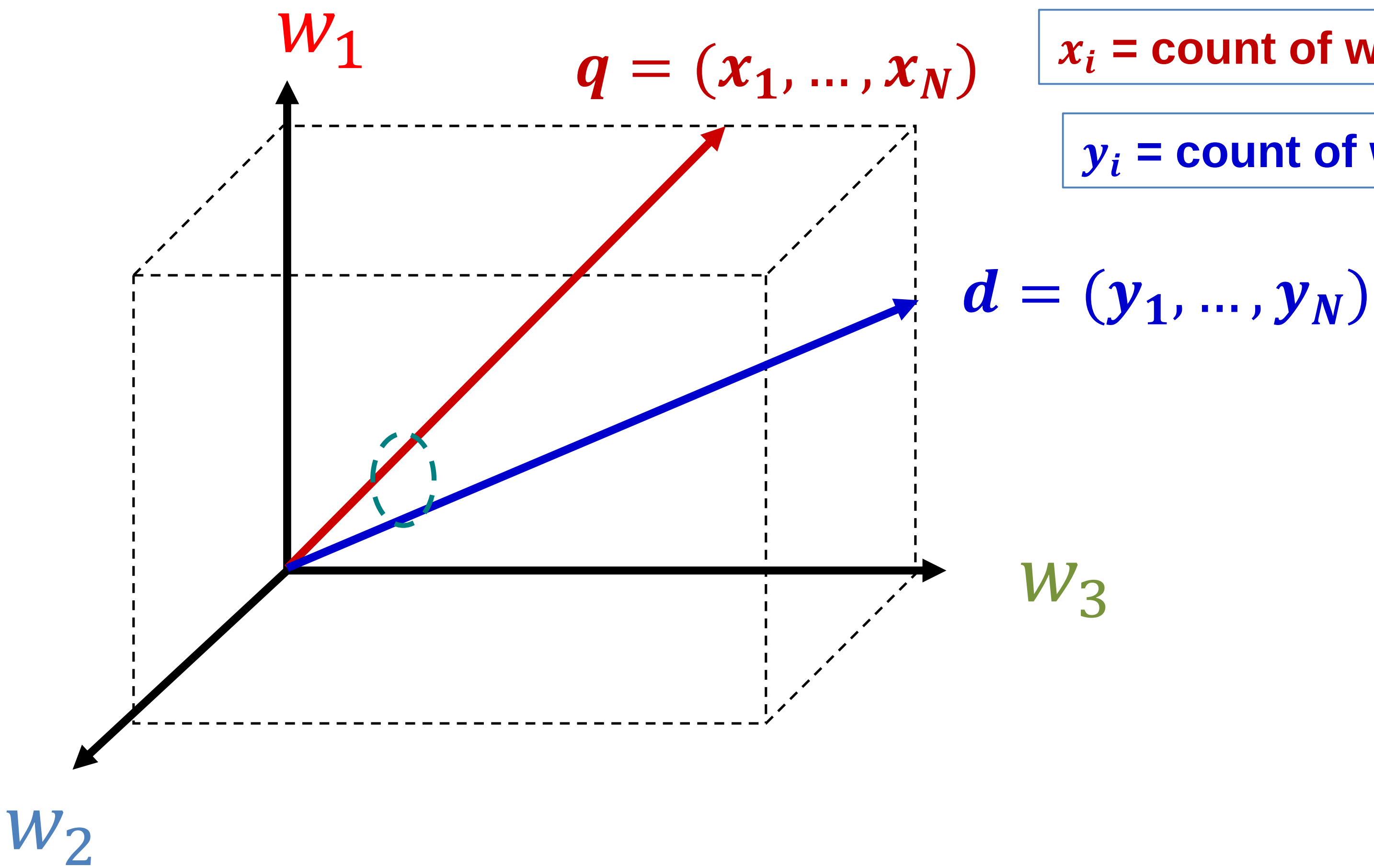
d_2 … news about organic food campaign … $f(q, d_2) = 3$

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

d_4 … news of presidential campaign …
… presidential candidate… $f(q, d_4) = 3$

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

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

$\boldsymbol{d_2}$ **… news about organic food campaign …**

$$q \;= (1, \quad 1, \quad 1, \quad 1, \quad 0, \quad ...)$$
$$d_2 = (1, \quad 1, \quad 0, \quad 1, \quad 1, \quad ...)$$

$$\boldsymbol{f(q, d_2) = 3}$$

$\boldsymbol{d_3}$ **… news of presidential campaign …**

$$q \;= (1, \quad 1, \quad 1, \quad 1, \quad 0, \quad ...)$$
$$d_3 = (1, \quad 0, \quad 1, \quad 1, \quad 0, \quad ...)$$

$$\boldsymbol{f(q, d_3) = 3}$$

$\boldsymbol{d_4}$ **… news of presidential campaign …**
… presidential candidate…

$$q = (1, \quad 1, \quad 1, \quad 1, \quad 0, \quad ...)$$
$$d_4 = (1, \quad 0, \quad 2, \quad 1, \quad 0, \quad ...)$$

$$\boldsymbol{f(q, d_4) = 4!}$$

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

$\boldsymbol{d_2}$ **… news about organic food campaign …**

$\boldsymbol{d_3}$ **… news of presidential campaign …**

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

$$q = (1, \quad 1, \quad 1, \quad 1, \quad 0, \quad \ldots)$$
$$d_2 = (1, \quad 1, \quad 0, \quad 1, \quad 1, \quad \ldots)$$

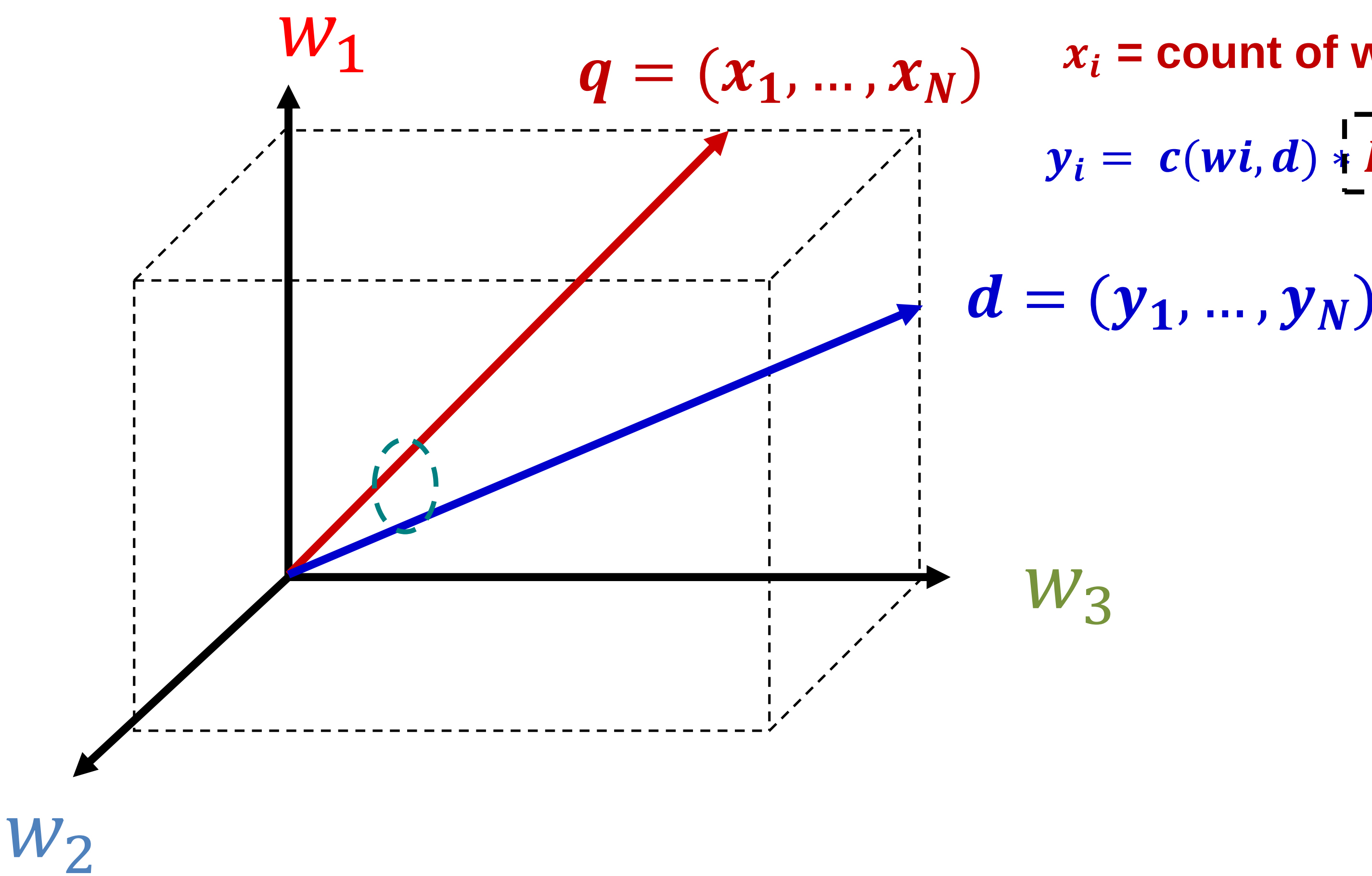
$$q = (1, \quad 1, \quad 1, \quad 1, \quad 0, \quad \ldots)$$
$$d_3 = (1, \quad 0, \quad 1, \quad 1, \quad 0, \quad \ldots)$$

$$\boldsymbol{f(q, d_2) < 3}$$
$$\boldsymbol{f(q, d_3) > 3}$$

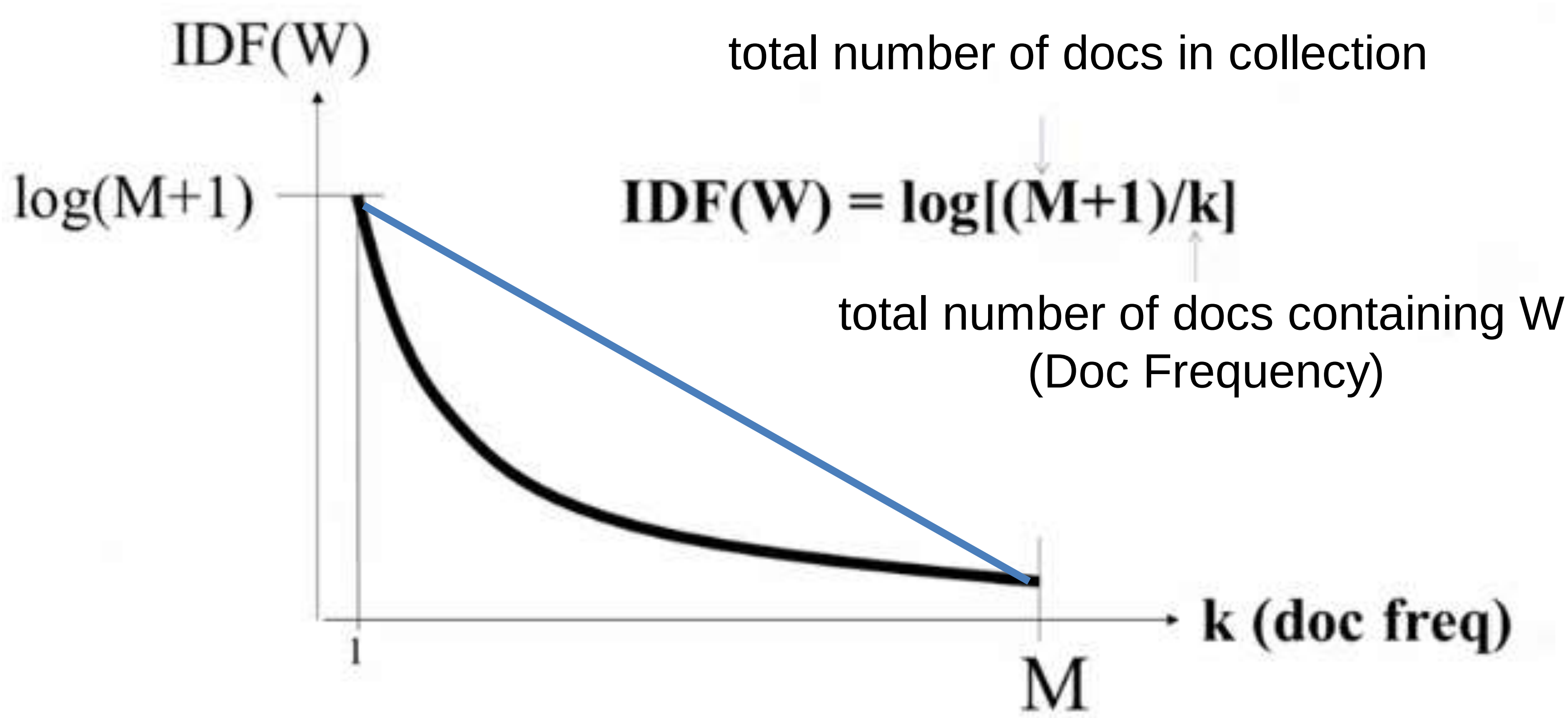
Further Improvement of Vector Placement: Adding Inverse Document Frequency (IDF)

x_i = count of word w_i in query

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



IDF Weighting: Penalizing Popular Terms



Solving Problem 2 ("presidential" vs. "about")?

$\boldsymbol{d_2}$ … news about organic food campaign …

$\boldsymbol{d_3}$ … news of presidential campaign …

$$V = \{news, about, presidential, campaign, food, \ldots\}$$
$$IDF(W) = 1.5 \quad 1.0 \quad 2.5 \quad 3.1 \quad 1.8$$

$$q = (1, \quad 1, \quad 1, \quad 1, \quad 0, \quad \ldots)$$
$$d_2 = (1 * 1.5, 1 * 1.0, 0, \quad 1 * 3.1, \quad 1 * 1.8, \ldots)$$

$$q = (1, \quad 1, \quad 1, \quad 1, \quad 0, \quad \ldots)$$
$$d_3 = (1 * 1.5, 0, \quad 1 * 2.5, \quad 1 * 3.1 \quad 0, \quad \ldots)$$

$$\boldsymbol{f(q, d_2) = 5.6 < f(q, d_3) = 7.1}$$

How Effective is VSM with 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 …**	$\boldsymbol{f(q, d_5) = 13.9}$!

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

d_5

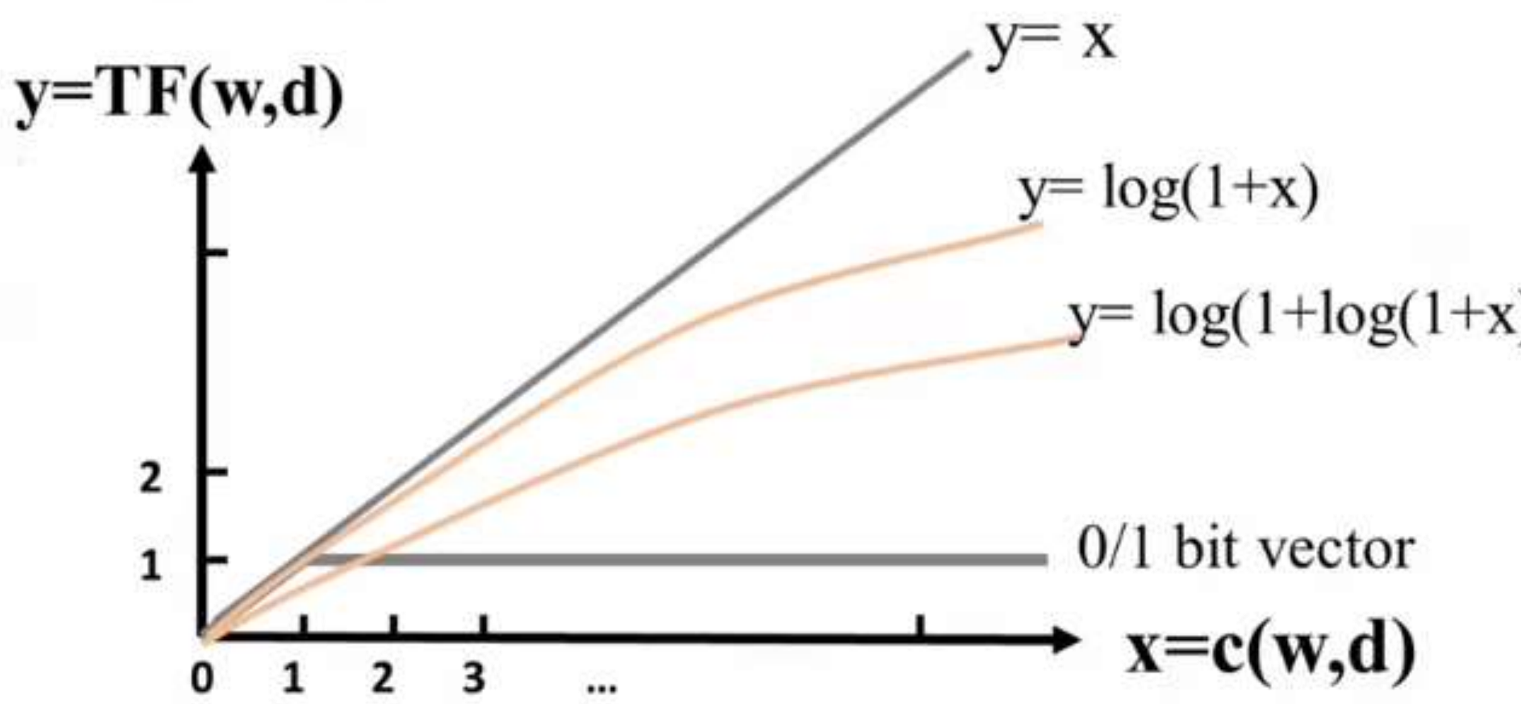
… **news** of organic food **campaign** …
… **campaign** …. **campaign** … **campaign** …

$c(\text{"}campaign\text{"}, d_5) = 4$

$\Rightarrow f(q, d_5) = 13.9?$

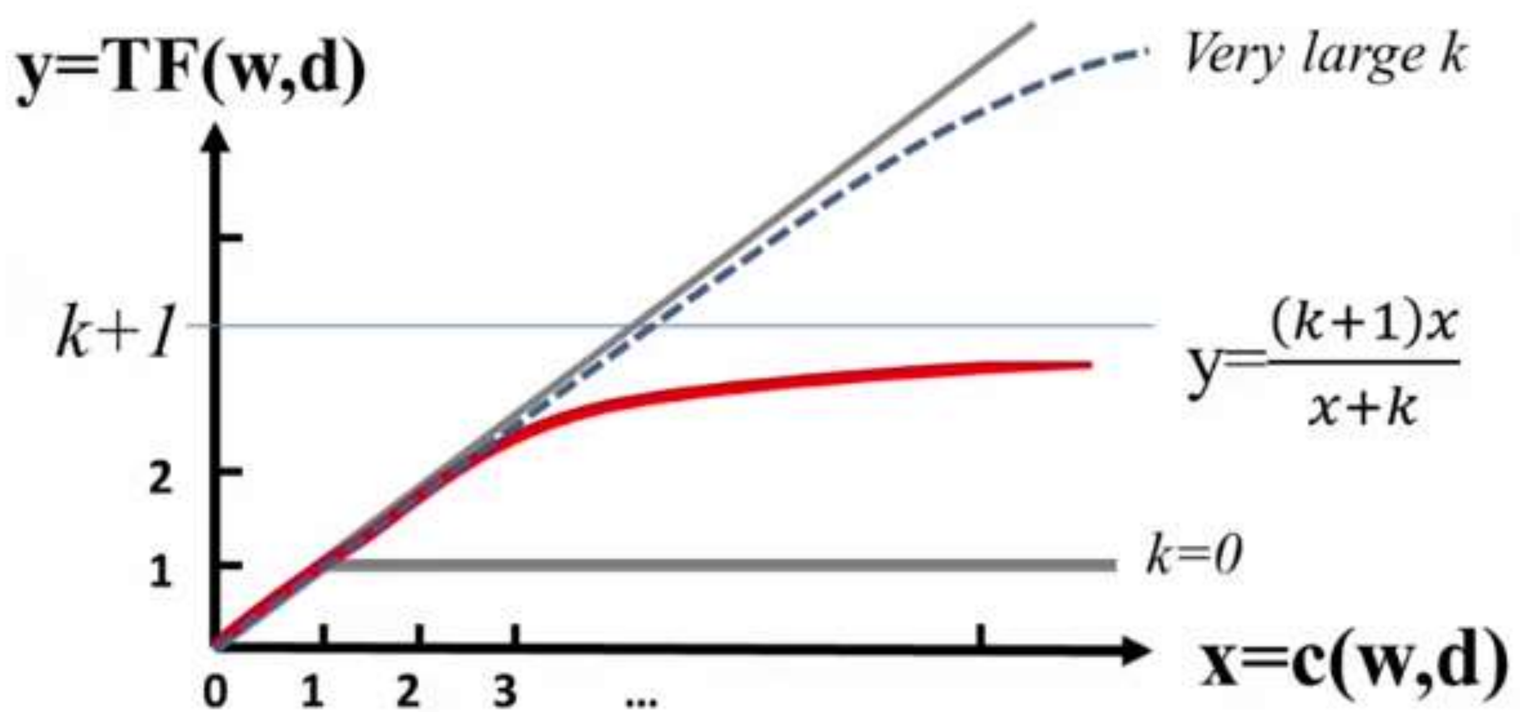
TF Transformation: $c(w,d) \rightarrow TF(w,d)$

Term Frequency Weight



TF Transformation: BM25 Transformation

Term Frequency Weight



What about Document Length?

Query = “news about presidential campaign”

d_4

… news of presidential campaign …
… presidential candidate…

100 words

$d_6 > d_4$?

d_6

… campaign ……………………………………………

5000 words

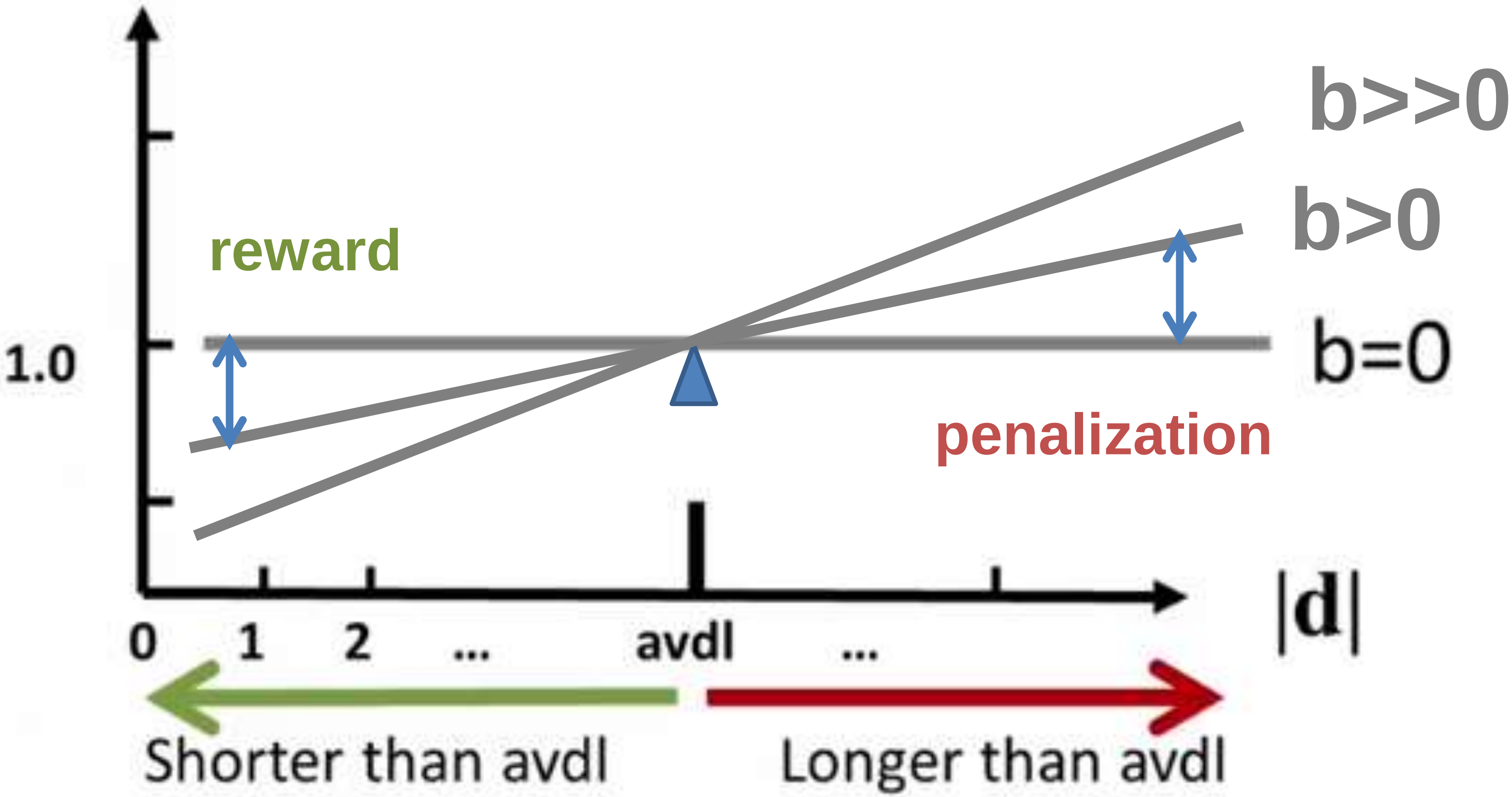
………………………………………………………………….
……………news……………………………………………
………………………………………………………………….
……………………………………………………news……………….
………………………………………………………………….
… presidential …. presidential …

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 doc is long because
  - it uses more words $\rightarrow$ more penalization
  - it has more contents $\rightarrow$ less penalization
- Pivoted length normalizer: average doc length as "pivot"
  - Normalizer = 1 if $|d|$ = average doc length ($avdl$)

Pivoted Length Normalization

$$normalizer = 1 - b + b\frac{|d|}{avdl} \qquad b \in [0,1]$$



State of the Art VSM Ranking Functions

- Pivoted Length Normalization VSM [Singhal et al 96]

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

- BM25/Okapi [Robertson & Walker 94]

$b \in [0,1]$

$k_1, k_3 \in [0,+\infty)$

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

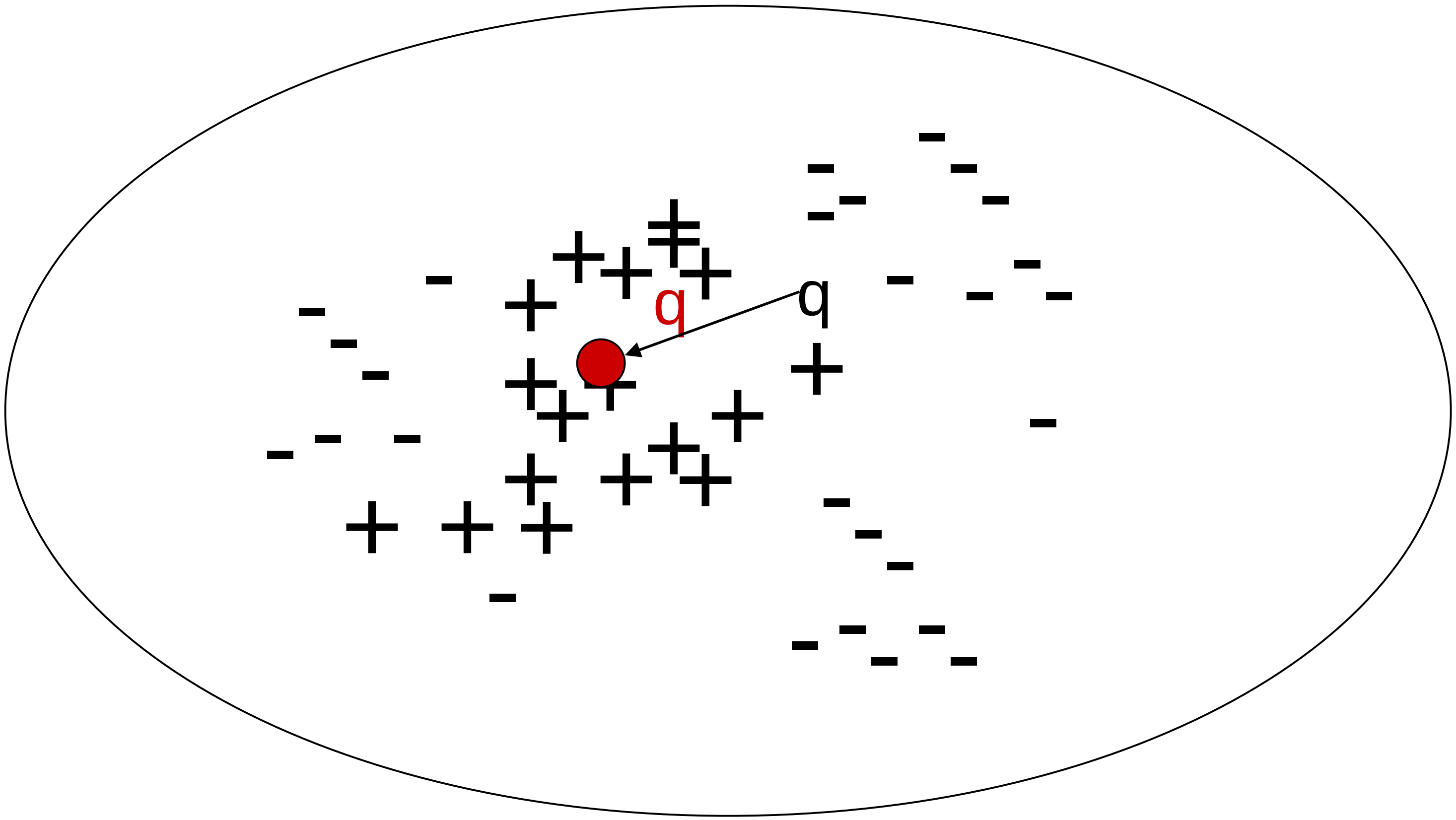
Further Improvement of VSM?

- 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)

Relevance Feedback in VS

- Basic setting: Learn from examples
  - Positive examples: docs known to be relevant
  - Negative examples: docs known to be non-relevant
  - How do you learn from this to improve performance?
- General method: Query modification
  - Adding new (weighted) terms
  - Adjusting weights of old terms
  - Doing both
- The most well-known and effective approach is Rocchio **[Rocchio 1971]**

Rocchio Feedback: Illustration



Rocchio Feedback: Formula

Parameters

New query

$$\vec{q}_m = \alpha \, \vec{q} + \frac{\beta}{|D_r|} \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{|D_n|} \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

Origial query

Rel docs

Non-rel docs

Rocchio in Practice

- Negative (non-relevant) examples are not very important (why?)
- Often project the vector onto a lower dimension (i.e., consider only a small number of words that have high weights in the centroid vector)
- Avoid "training bias" (keep relatively high weight on the original query weights)
- Can be used for relevance feedback and pseudo feedback
- Usually robust and effective

Advantages of VS Model

- Empirically effective! (Top TREC performance)
- Intuitive
- Easy to implement
- Well-studied/Most evaluated
- Warning: Many variants of TF-IDF!

Disadvantages of VS Model

- Assume term independence
- Assume query and document to be the same
- Lots of parameter tuning!

Questions?