

# Week 1

06 July 2018 22:20

## Lecture 1.1 (NL Content Analysis):

What is NLP?

Why is it difficult for computers?

Sematic Analysis

Part of speech tagging

Entity extraction - partial understanding of sentence

Word sense disintegration - how words have different meanings in different contexts

Sentiment analysis - whether sentence is positive or negative - useful in review analysis

Inference

Speech Act analysis

NLP for Text Retrieval - General, robust & efficient - shallow NLP techniques are used

Bag of words representation - used by most search engines

---

## Lecture 1.2 (Text Access):

Text-Information systems

Modes of text access:

- Push (recommender)

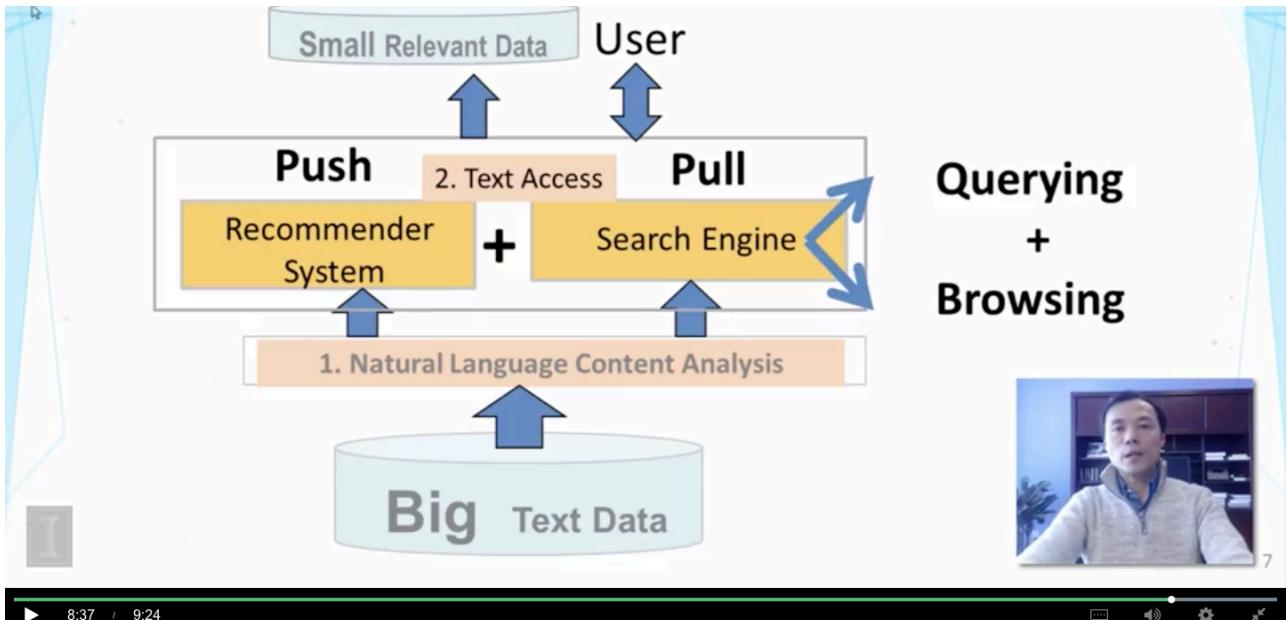
vs.

- Pull (search engines)

0. Querying vs. Browsing

Summary text access (picture):





### Lecture 1.3 (Text Retrieval problem):

Formal Formulation of text retrieval:

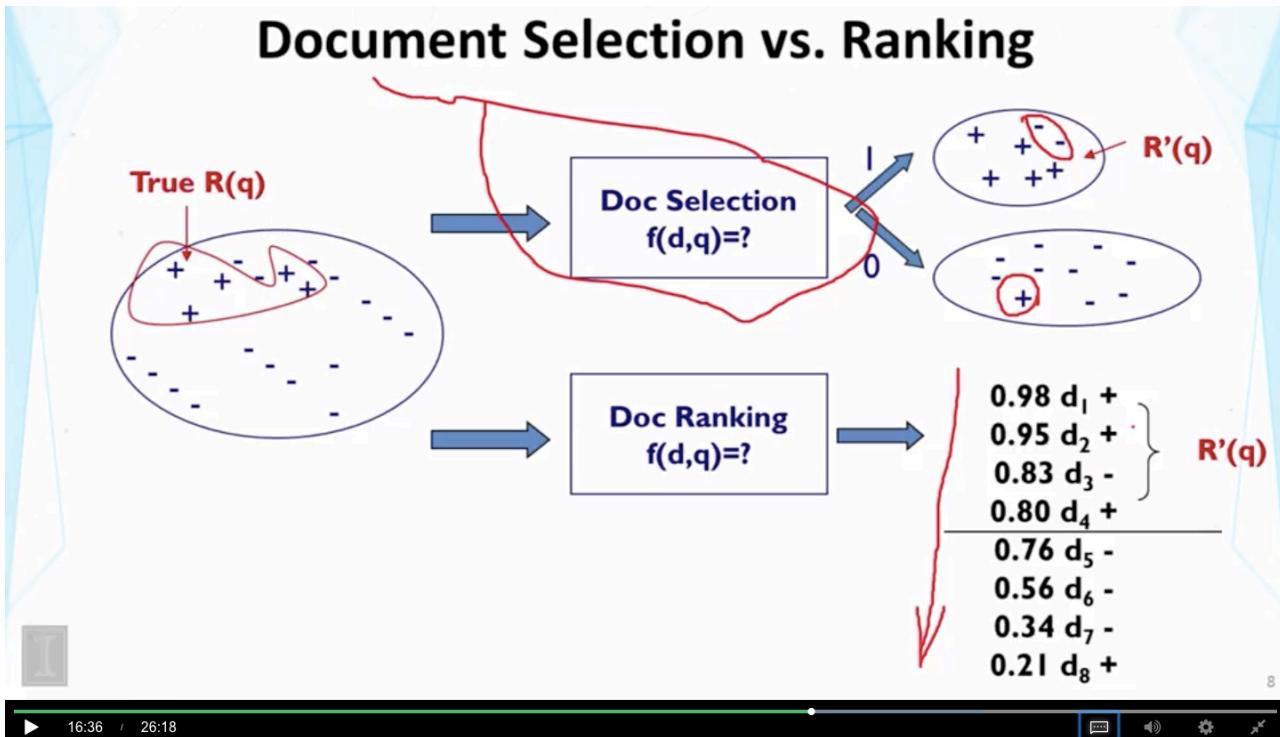
## Formal Formulation of TR

- **Vocabulary:**  $V = \{w_1, w_2, \dots, w_N\}$  of language
- **Query:**  $q = q_1, \dots, q_m$ , where  $q_i \in V$
- **Document:**  $d_i = d_{i1}, \dots, d_{im_i}$ , where  $d_{ij} \in V$
- **Collection:**  $C = \{d_1, \dots, d_M\}$
- **Set of relevant documents:**  $R(q) \subseteq C$ 
  - Generally unknown and user-dependent
  - Query is a “hint” on which doc is in  $R(q)$
- **Task** = compute  $R'(q)$ , an approximation of  $R(q)$



Methods of TR:

1. Document selection - binary classifier (document shown or not)
2. Document ranking - assign probability of relevance of the document



Document ranking is better suited to user queries. Why?

Problems with document selection:

## Problems of Document Selection

- The classifier is unlikely accurate
  - “Over-constrained” query → no relevant documents to return
  - “Under-constrained” query → over delivery
  - Hard to find the right position between these two extremes
- Even if it is accurate, all relevant documents are not equally relevant (relevance is a matter of degree!)
  - Prioritization is needed
- Thus, ranking is generally preferred



Why Ranking? Theoretically justify ranking method using probability ranking principle.

## Theoretical Justification for Ranking

- **Probability Ranking Principle** [Robertson 77]: Returning a ranked list of documents in descending order of probability that a document is relevant to the query is the optimal strategy under the following two assumptions:
  - The utility of a document (to a user) is **independent** of the utility of any other document
  - A user would browse the results **sequentially**
- Do these two assumptions hold?



However, none of the assumptions are actually true.

- a. In first assumption, there is no use showing those documents which are identical or similar to the ones the user has read, meaning that utility of documents depend on other documents.
- b. In second assumption, it might be the case that one document might not be helpful to the user but, when three documents are put together, they make sense to the user and satiate user's query.

These assumptions aren't necessarily true, still a lot of research work is based on these assumptions. We can address this problem by post-processing rank list to remove not useful / redundant documents.

The main challenge is to design the ranking function, what value is to be assigned to the query-document pair.

#### Lecture 1.4 (Text Retrieval Methods):

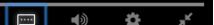
How to design a Ranking function:

## How to Design a Ranking Function

- **Query:**  $q = q_1, \dots, q_m$ , where  $q_i \in V$
- **Document:**  $d = d_1, \dots, d_n$ , where  $d_i \in V$

- **Ranking function:**  $f(q, d) \in \mathbb{R}$
- 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)

1:30 / 10:10



3

Ranking function - measure the likelihood that document 'd' is relevant to query 'q'  
Retrieval model - provides a computational definition of relevance

Types of retrieval models:

## Many Different Retrieval Models

- **Similarity-based models:**  $f(q, d) = \text{similarity}(q, d)$ 
  - 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:**  $f(q, d) = p(d \rightarrow q)$
- **Axiomatic model:**  $f(q, d)$  must satisfy a set of constraints
- These different models tend to result in similar ranking functions involving similar variables

4:22 / 10:10



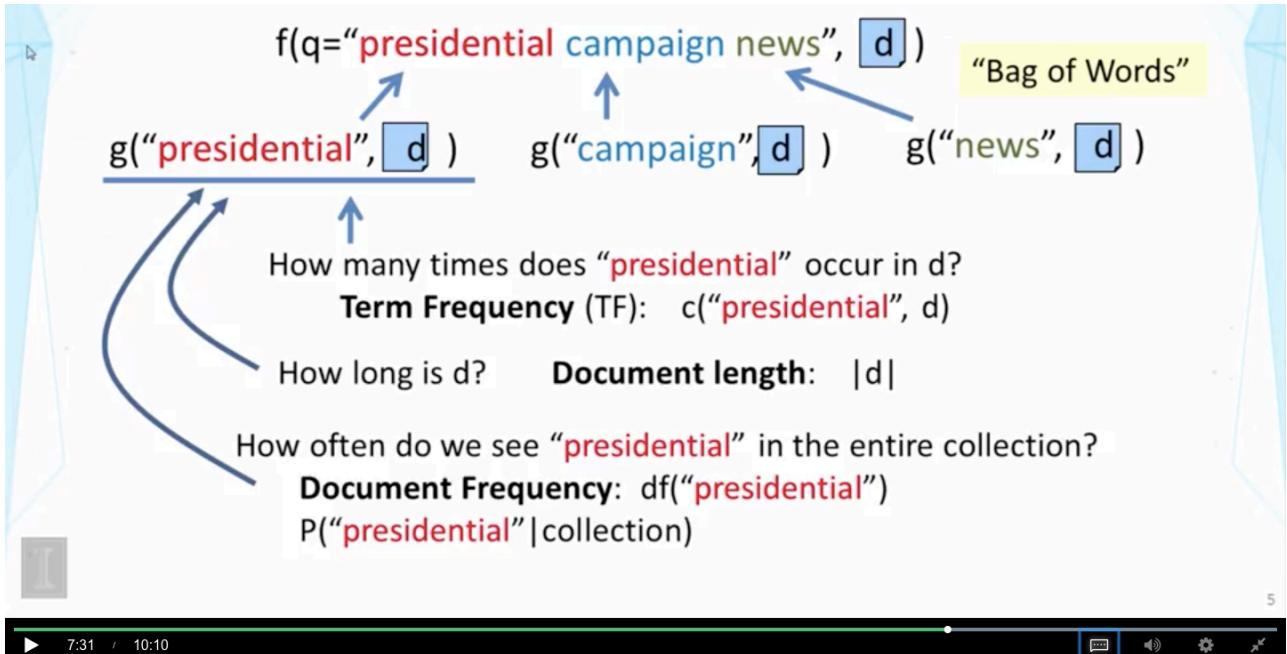
4

State-of-the-Art Text Retrieval models:

Rely on:

1. Bag of words representation
2. Term Frequency (TF) & Document Frequency (DF) of words
3. Document length

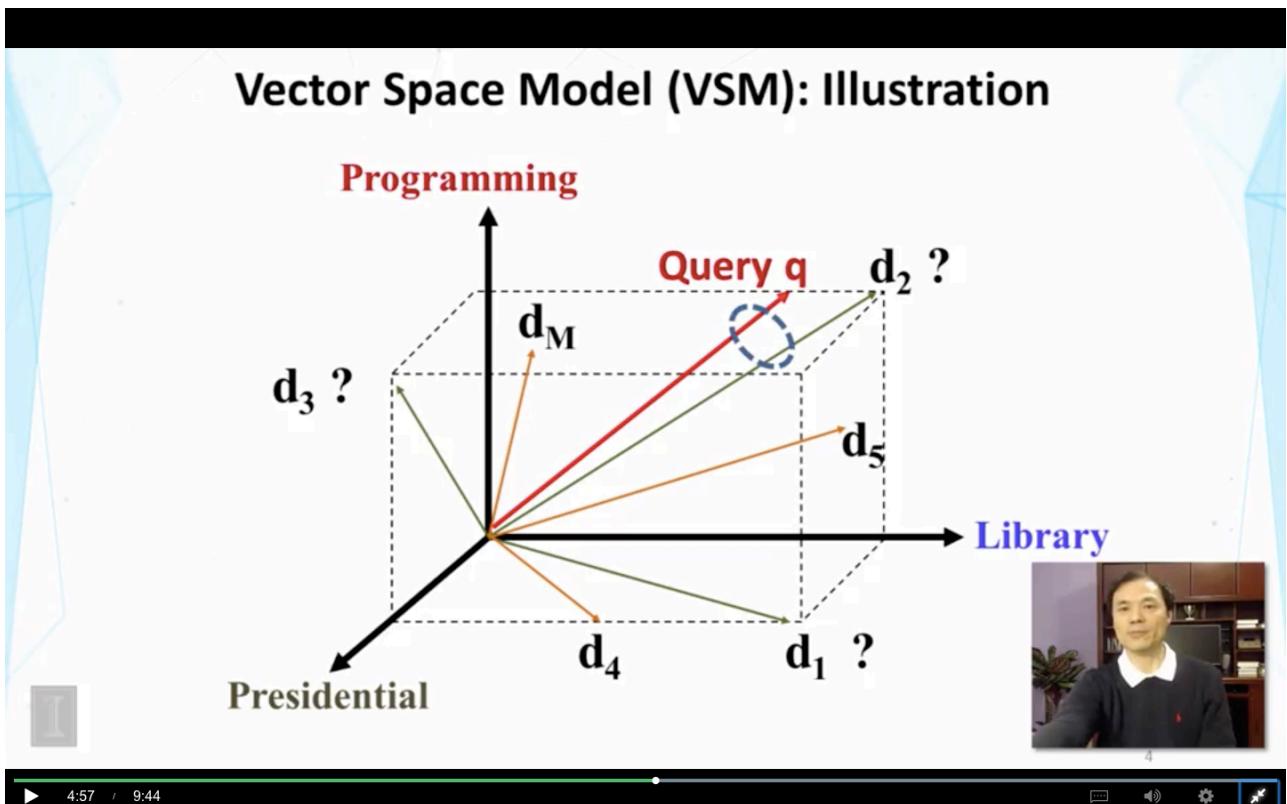
## Common Ideas in State of the Art Retrieval Models



### Lecture 1.5 (Vector Space Model):

VSM is a framework

What is VSM ?



This basic idea of VSM leaves out lots of important mentions due to which this idea cannot be programmed yet.

## What VSM Doesn't Say

- How to define/select the “basic concept”
  - Concepts are assumed to be orthogonal
- How to place docs and query in the space (= how to assign term weights)
  - Term weight in query indicates importance of term
  - Term weight in doc indicates how well the term characterizes the doc
- How to define the similarity measure

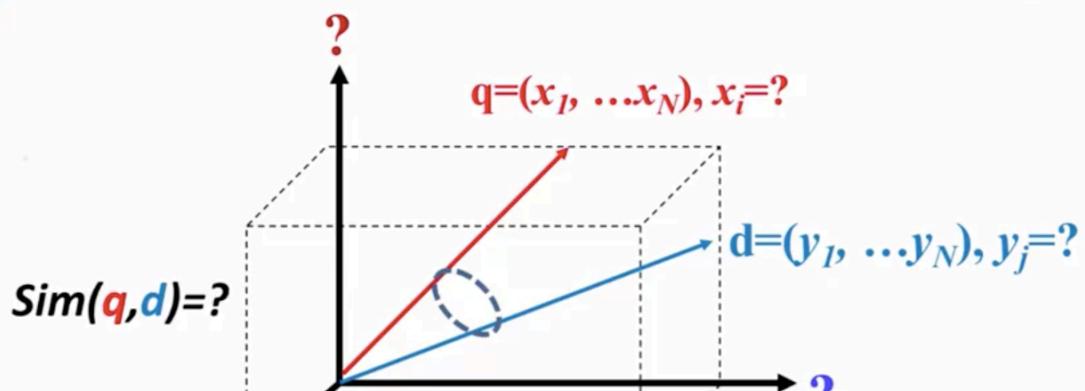


### Lecture 1.6 (VSM - simplest instantiation):

What VSM doesn't say:

1. How to represent dimensions of the model (Dimension instantiation)
2. How to represent the vectors in the space, what weights do we assign to the terms
3. How do we measure similarity between query vector 'q' and document vector 'd'

## What VSM Doesn't Say

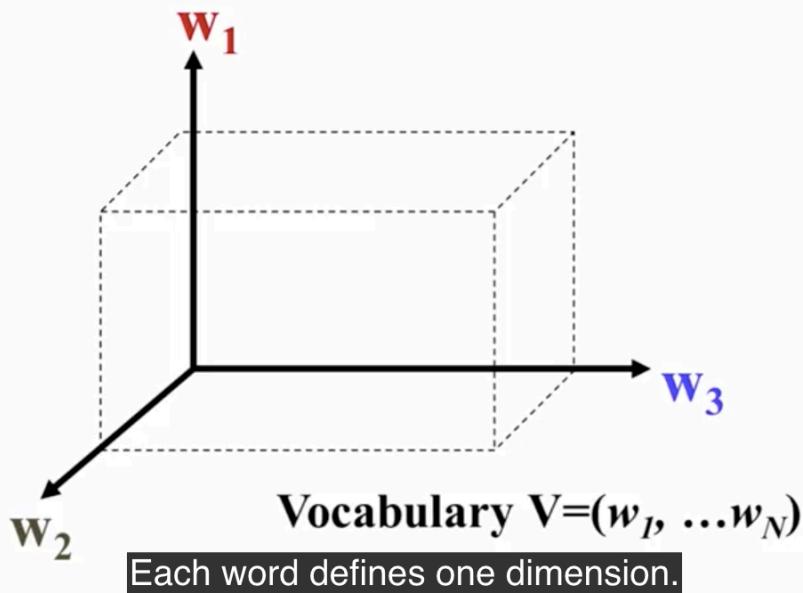




So, let's define these terms:

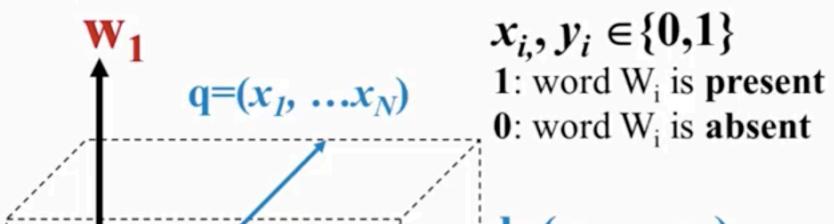
1. Dimension instantiation

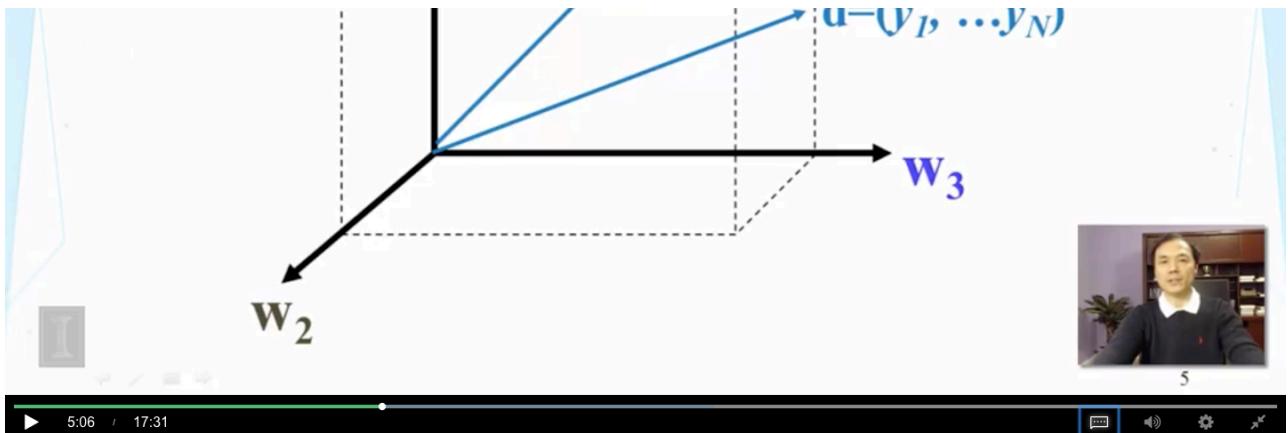
## Dimension Instantiation: Bag of Words (BOW)



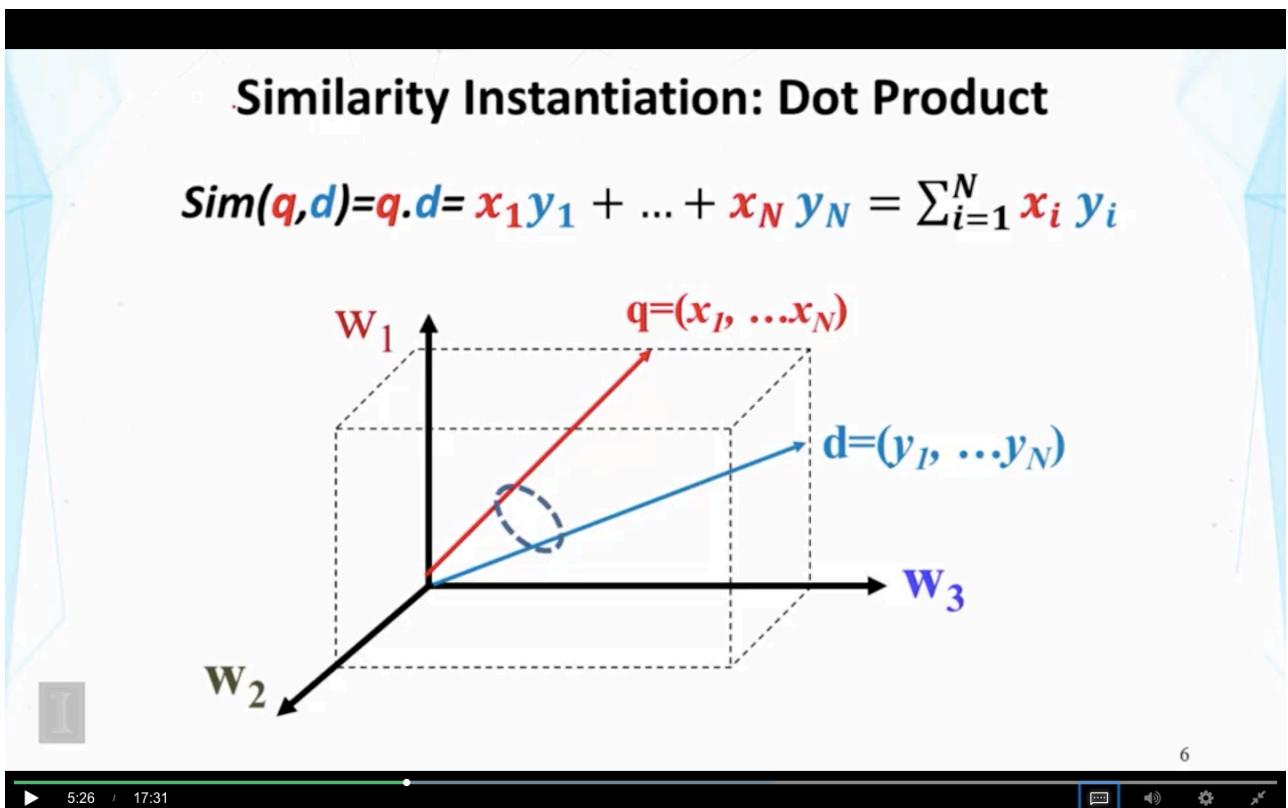
2. Vector placement: Bit vector used here; We can also use the term **frequency** (no. of occurrences) or **document frequency** (no. of documents in which these words occur) as a weight to assign these vectors. These might provide more information as to how the query and document contain these dimensions (words in vocabulary).

## Vector Placement: Bit Vector

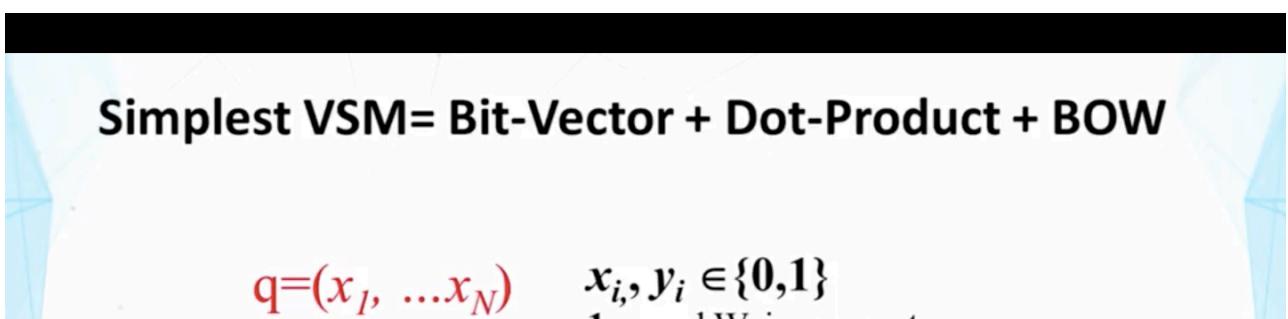




3. Similarity instantiation: Dot product used here - we can also use distance measures (Euclidean, etc.) to measure distance b/w 2 vectors. More distance, less similar are query and document vectors.



Simplest VSM instantiation:



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

1: word  $w_i$  is **present**

0: word  $W_i$  is **absent**

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

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

7



How do we say whether this ranking function is good or not? What does the value of this ranking function represent?

Example:

### An Example: How Would You Rank These Documents?

Query = “news about presidential campaign”

Ideal Ranking?

d1

... news about ...

d4 +  
d3 +

d2

... news about organic food campaign...

d3

... news of presidential campaign ...

d4

... news of presidential campaign ...  
... presidential candidate ...

d1 -  
d2 -

d5

... news of organic food campaign...  
campaign...campaign...campaign...

d5



8



Let's see how our simplest instantiation of VSM does on matching this query with relevant documents.

### Ranking Using the Simplest VSM

Query = “news about presidential campaign”

d1 news about



... news about ...

d3 ... news of presidential campaign ...

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

$$q = (1, 1, 1, 1, 0, \dots)$$

$$d1 = (1, 1, 0, 0, 0, \dots)$$

$$f(q, d1) = 1 * 1 + 1 * 1 + 1 * 0 + 1 * 0 + 0 * 0 + \dots = 2$$

$$d3 = (1, 0, 1, 1, 0, \dots)$$

$$f(q, d3) = 1 * 1 + 1 * 0 + 1 * 1 + 1 * 1 + 0 * 0 + \dots = 3$$



13:18 / 17:31

Observation: Whenever one of the words is not present in query or document, its contribution to dot product is 0, meaning we are counting no. of 1-1 pairs. So, the ranking function effectively tells how many unique words in query match with those in document.

With this knowledge, let's see if the simplest VSM is effective at ranking documents.

## 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...

$$f(q, d4) = 3$$

... presidential candidate ...

d5 ... news of organic food campaign...

$$f(q, d5) = 2$$

... campaign...campaign...campaign...



10

15:36 / 17:31

The **problem** is that this model ranks d2 above d3, because both have 3 matched words. But, we know that matching presidential is more important than matching about. So, there are still unresolved problems in this model.

Summary of simplest VSM:

## Summary

- VSM instantiation: dimension, vector placement, similarity
- Simplest VSM
  - Dimension = word
  - Vector = 0-1 bit vector (word presence/absence)
  - Similarity = dot product
  - $f(q,d)$  = number of **distinct** query words matched in d

11

15:48 / 17:31

Video player controls: play/pause, volume, settings, full screen.

-----X-----