

IS 531 Document Storage & Retrieval

Chapter 3-Information Retrieval

Dr. Bassam Hammo



What is a model?



- A model is a construct designed to help us understand a complex system
 - A particular way of “looking at things”
- Models certainly make simplifying assumptions
 - What are the limitations of the model?
- Different types of models:
 - Conceptual models
 - Physical analog models
 - Mathematical models
 - ...

The Central Problem in IR



Information Seeker



Concepts



Query Terms

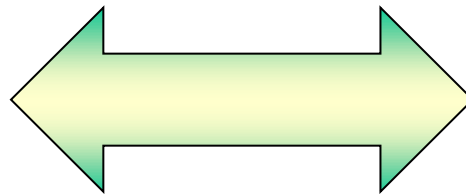
Authors



Concepts



Document Terms



Do these represent the same concepts?

What is Information Retrieval?



- Finding relevant information in large collections of data

- In such a collection you may want to find:

“Give me information on the history of Saudi Arabia”

An article about Saudi Arabia (text retrieval)

“What does a brain tumor look like on a CT-scan”

A picture of a brain tumor (image retrieval)

“It goes like this: dnn dnn daah. . .”

A certain song (audio retrieval)

The IR Problem



The standard information retrieval (IR) scenario

- The user has an **information need**
- The user types a **query** that describes the information need
- The IR system retrieves a set of documents from a **document collection** that it believes to be relevant
- The documents are **ranked** according to their likelihood of being relevant

The IR Problem



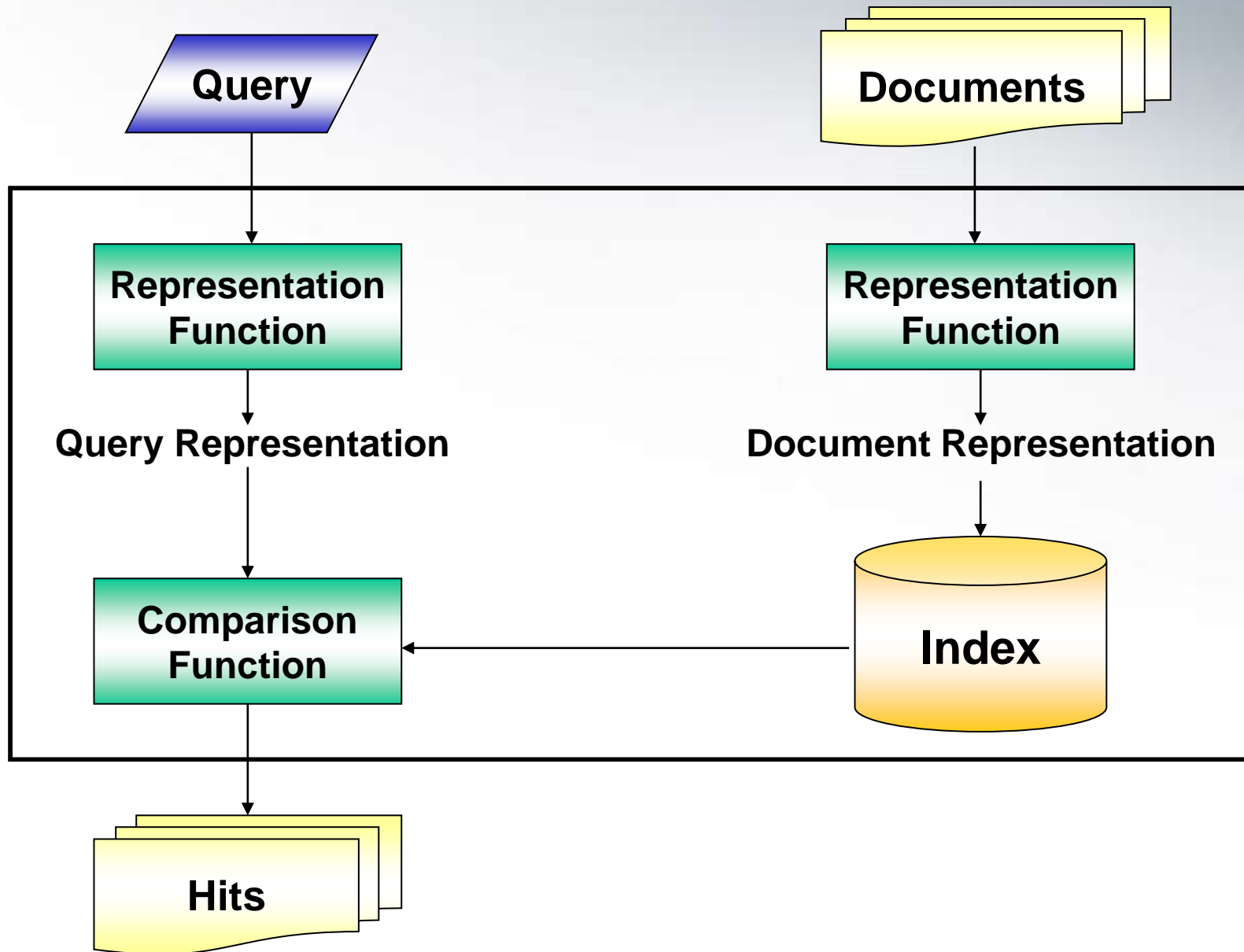
■ Input:

- a set/collection of documents (**corpus**)
(plural **corpora**)
- a user query

■ Output:

- a (**ranked**) list of relevant documents

The IR Black Box



Information Retrieval

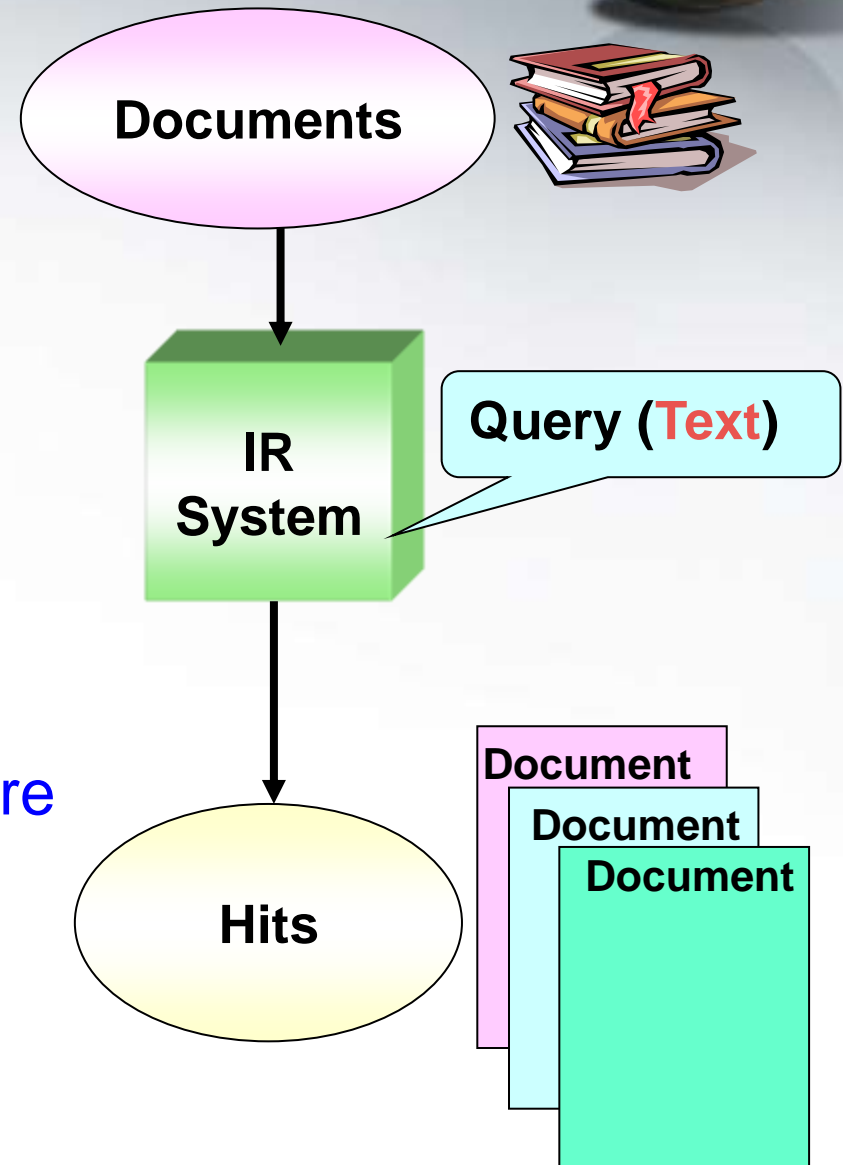


■ Given:

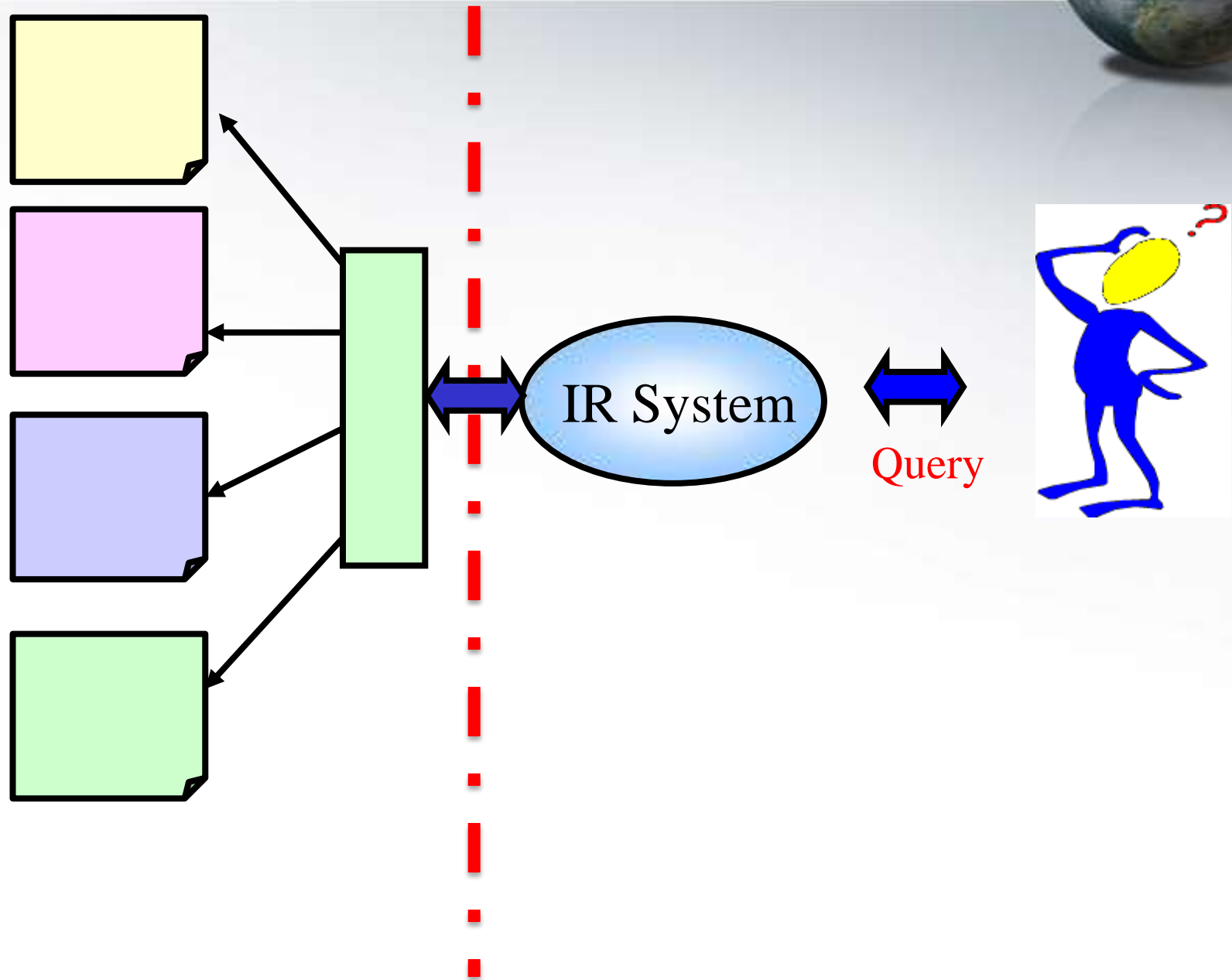
- A source of textual documents
- A user query (text based)

■ Find:

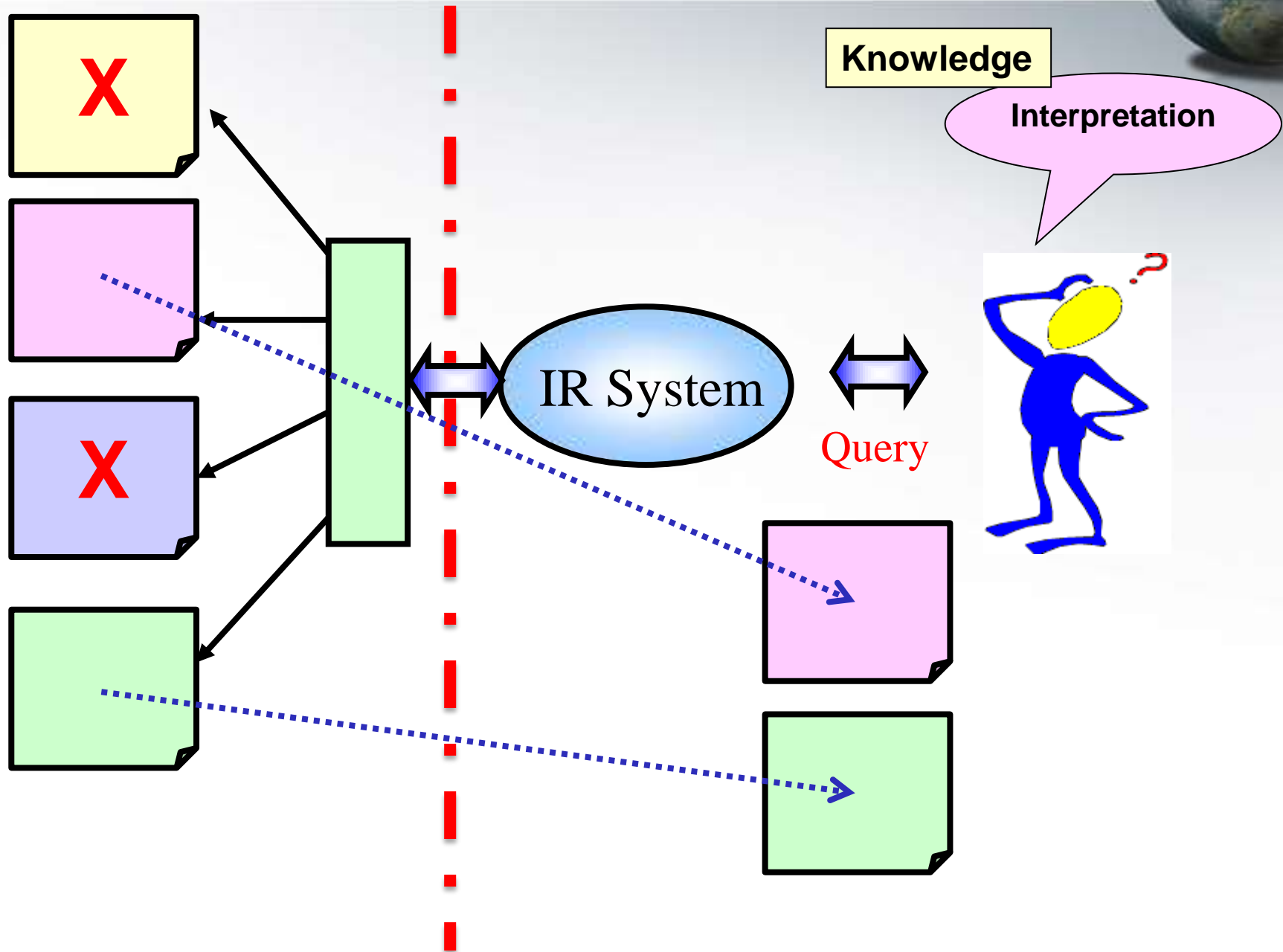
- A set of (ranked) documents that are relevant to the query



IR System



IR System



Retrieval Models



- A retrieval model is an idealization or abstraction of an actual retrieval process
- Conclusions derived from a model depend on whether the model is a good approximation of the retrieval situation
- Note that a retrieval model is not the same thing as a retrieval implementation

Retrieval Models



■ Boolean model

- Based on the notion of sets
- Documents are retrieved *only* if they satisfy Boolean conditions specified in the query
- Does not impose a ranking on retrieved documents
- Exact match

■ Vector space model

- Based on geometry, the notion of vectors in high dimensional space
- Documents are ranked based on their similarity to the query (ranked retrieval)
- Best/partial match

Components of a Retrieval Model



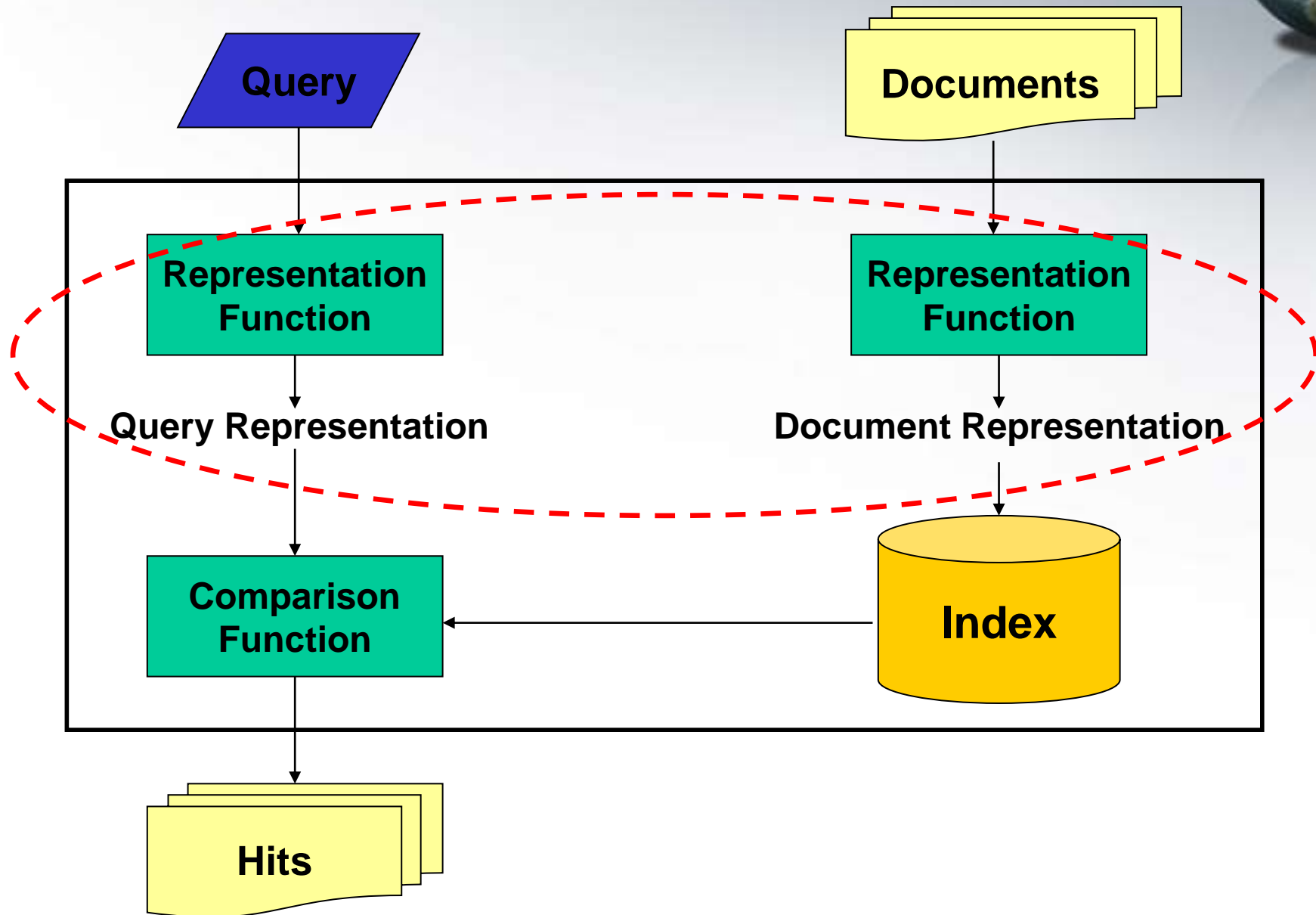
■ The user:

- Search expert (e.g., librarian) vs. non-expert
- Background of the user (knowledge of the topic)
- In-depth searching vs. 'just-want-to-get-an-idea' searching

■ The documents:

- Different languages
- Semi-structured (e.g. HTML or XML) vs. plain

Representing Text



Automatic Content Representation



- Using natural language understanding?
 - Computationally too expensive in real-world settings
 - Coverage
 - Language dependence
 - The resulting representations may be too explicit to deal with the vagueness of a user's information need

How do we represent text?



- How do we represent the complexities of language?
 - Keeping in mind that computers don't "understand" documents or queries
- Simple, yet effective approach: "bag of words"
 - Treat all the words in a document as index terms for that document
 - Assign a "weight" to each term based on its "importance"
 - Disregard order, structure, meaning, etc. of the words

What's the point?



- Retrieving relevant information is hard!
 - Evolving, ambiguous user needs, context, etc.
 - Complexities of language
- Bag-of-words approach:
 - Information retrieval is *all* (and *only*) about matching words in documents with words in queries

Bag-of-Words Approach



- A document is an unordered list of words
Grammatical information is lost
- Tokenization: What is a word?
Is 'Saudi Arabia' one or two words?
- Stemming or lemmatization
Morphological information is thrown away
 - 'agreements' becomes 'agreement' (lemmatization)
 - or even 'agree' (stemming)


Two Major Issues!



■ Indexing

- representing the document collection using words/terms
- for fast access to documents

■ Retrieval methods

- matching a user query to indexed documents
- Three major models are used:
 1. Boolean model
 2. Vector-Space model (VS) 
 3. Probabilistic model

Vector Representation



- “Bags of words” can be represented as vectors
 - Why? Computational efficiency, ease of manipulation
 - Geometric metaphor: “arrows”
- A vector is a set of values recorded in any consistent order
 - “The quick brown fox jumped over the lazy dog’s back”
 - [1 1 1 1 1 1 1 1 2]
 - 1st position corresponds to “back”
 - 2nd position corresponds to “brown”
 - 3rd position corresponds to “dog”
 - 4th position corresponds to “fox”
 - 5th position corresponds to “jump”
 - 6th position corresponds to “lazy”
 - 7th position corresponds to “over”
 - 8th position corresponds to “quick”
 - 9th position corresponds to “the”

Representing Documents



Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

Stopword List

for
is
of
the
to

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Boolean Retrieval

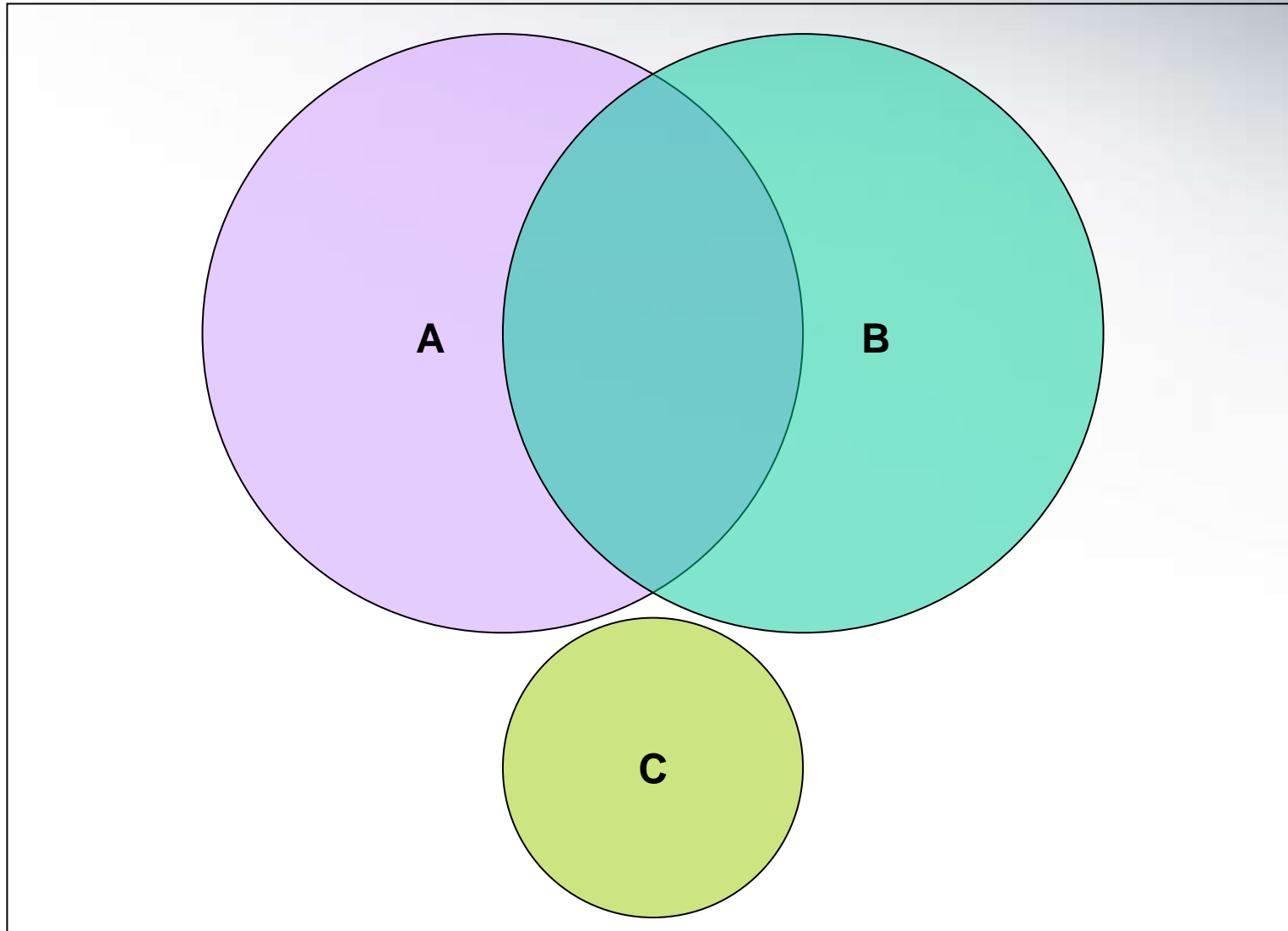


- Weights assigned to terms are either “0” or “1”
 - “0” represents “absence”: term **isn't** in the document
 - “1” represents “presence”: term **is** in the document
- Build queries by combining terms with Boolean operators
 - **AND, OR, NOT**
- The system returns all documents that satisfy the query

AND/OR/NOT



All documents



Logic Tables



A \ B	0	1
0	0	1
1	1	1

A OR B

B	0	1
1	1	0

NOT B

A \ B	0	1
0	0	0
1	0	1

A AND B

A \ B	0	1
0	0	0
1	1	0

A NOT B
(= A AND NOT B)

Boolean View of a Collection



Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8
aid	0	0	0	1	0	0	0	1
all	0	1	0	1	0	1	0	0
back	1	0	1	0	0	0	1	0
brown	1	0	1	0	1	0	1	0
come	0	1	0	1	0	1	0	1
dog	0	0	1	0	1	0	0	0
fox	0	0	1	0	1	0	1	0
good	0	1	0	1	0	1	0	1
jump	0	0	1	0	0	0	0	0
lazy	1	0	1	0	1	0	1	0
men	0	1	0	1	0	0	0	1
now	0	1	0	0	0	1	0	1
over	1	0	1	0	1	0	1	1
party	0	0	0	0	0	1	0	1
quick	1	0	1	0	0	0	0	0
their	1	0	0	0	1	0	1	0
time	0	1	0	1	0	1	0	0

Each column represents the view of a particular document: What terms are contained in this document?

Each row represents the view of a particular term: What documents contain this term?

To execute a query, pick out rows corresponding to query terms and then apply logic table of corresponding Boolean operator

Sample Queries



Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8
dog	0	0	1	0	1	0	0	0
fox	0	0	1	0	1	0	1	0

$\text{dog} \wedge \text{fox}$	0	0	1	0	1	0	0	0
--------------------------------	---	---	---	---	---	---	---	---

dog AND fox → Doc 3, Doc 5

$\text{dog} \vee \text{fox}$	0	0	1	0	1	0	1	0
------------------------------	---	---	---	---	---	---	---	---

dog OR fox → Doc 3, Doc 5, Doc 7

$\text{dog} \neg \text{fox}$	0	0	0	0	0	0	0	0
------------------------------	---	---	---	---	---	---	---	---

dog NOT fox → empty

$\text{fox} \neg \text{dog}$	0	0	0	0	0	0	1	0
------------------------------	---	---	---	---	---	---	---	---

fox NOT dog → Doc 7

Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8
good	0	1	0	1	0	1	0	1
party	0	0	0	0	0	1	0	1

$\text{g} \wedge \text{p}$	0	0	0	0	0	1	0	1
over	1	0	1	0	1	0	1	1

good AND party → Doc 6, Doc 8

$\text{g} \wedge \text{p} \neg \text{o}$	0	0	0	0	0	1	0	0
------------------------------------------	---	---	---	---	---	---	---	---

good AND party NOT over → Doc 6

Proximity Operators



- More “precise” versions of AND
 - “NEAR n ” allows at most $n-1$ intervening terms
 - “WITH” requires terms to be adjacent and in order
 - Other extensions: within n sentences, within n paragraphs, etc.
- Relatively easy to implement, but less efficient
 - Store position information for each word in the document vectors
 - Perform normal Boolean computations, but treat WITH and NEAR as extra constraints

Other Extensions



- Ability to search on fields
 - Leverage document structure: title, headings, etc.
- Wildcards
 - comp^* = compute, computing, computer, computation, computerization, computational, etc.
- Special treatment of dates, names, companies, etc.

Why Boolean Retrieval Works



- Boolean operators *approximate* natural language
- AND can discover relationships between concepts
- OR can discover alternate terminology
- NOT can discover alternate meanings

Strengths and Weaknesses



■ Strengths

- Precise, if you know the right strategies
- Precise, if you have an idea of what you're looking for
- Efficient for the computer

■ Weaknesses

- Users must learn Boolean logic
- Boolean logic insufficient to capture the richness of language
- No control over size of result set: either too many documents or none
- When do you stop reading? All documents in the result set are considered "equally good"
- What about partial matches? Documents that "don't quite match" the query may be useful also

Ranked Retrieval



- Order documents by how likely they are to be relevant to the information need
 - Present hits one screen at a time
 - At any point, users can continue browsing through ranked list or reformulate query
- Attempts to retrieve relevant documents directly, not merely provide tools for doing so

Why Ranked Retrieval?



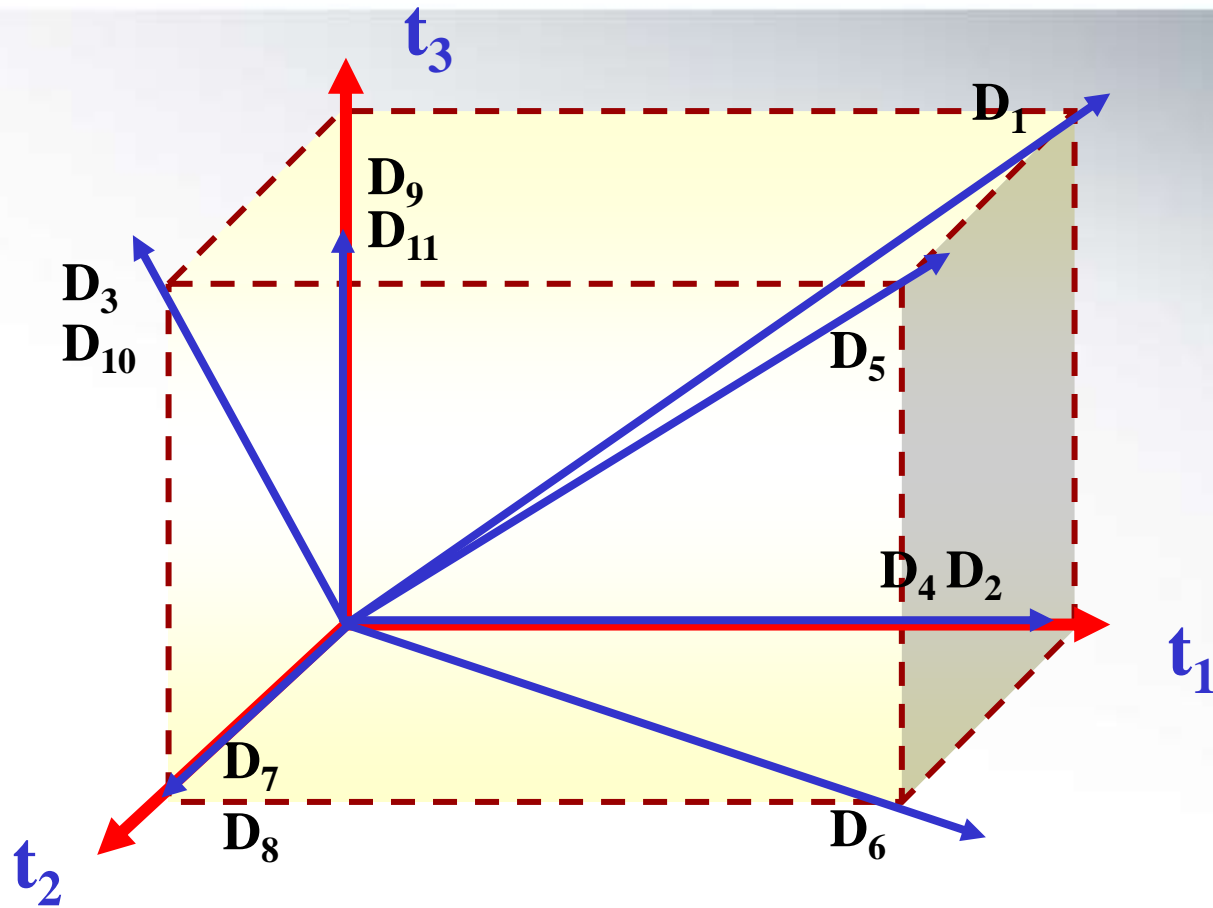
- Arranging documents by relevance is
 - Closer to how humans think: some documents are “better” than others
 - Closer to user behavior: users can decide when to stop reading
- Best (partial) match: documents need not have all query terms
 - Although documents with more query terms should be “better”

Similarity-Based Queries



- Let's replace relevance with “similarity”
 - Rank documents by their similarity with the query
- Treat the query as if it were a document
 - Create a query bag-of-words
- Find its similarity to each document
- Rank order the documents by similarity
- Surprisingly, this works pretty well!

Documents in Vector Space



Postulate: Documents that are “close together” in vector space “talk about” the same things

Therefore, retrieve documents based on how close the document is to the query (i.e., similarity \sim “closeness”)

Indexing



- Most IR systems use an **inverted file** to represent the texts in the collection
- Inverted file = a table of terms with a list of texts that contain these terms

<i>information</i>	{d ₁ , d ₄ , d ₉₅ , d ₅ , d ₉₀ ...}
<i>retrieval</i>	{d ₃ , d ₇ , d ₉₅ ...}
<i>system</i>	{d ₂₄ , d ₇ , d ₄₄ ...}
<i>inverted</i>	{d ₃ , d ₅₅ , d ₉₀ , d ₉₈ ...}

Example of an inverted file



INVERTED DICTIONARY

Token	DocCnt	FreqCnt	Head
LANGAUGE	28	51	•
NATURAL	32	37	•
PROCESSING	135	185	...
SYSTEM	7	10	...

POSTING

DocNo	Freq	Word Position	
67	2	279 283	•
424	1	24	•
1376	7	137 189 481...	..
206	1	170	•
4819	2	4 26 32	..



Term-Value Vector Representation



- **term**: all possible terms that occur in the query/document
- **value**: presence or absence of term in query/document

T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	...	T_n
V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	...	V_n

- value can be
 - Binary (0, if term is absent ; 1, if term is present)
 - Term frequency (raw frequency), or
 - Term weight, $tf \times idf$

Term-Weight Vector Representation



- Binary values (1,0) do not tell if a term is more important than others
- We should weight the terms by importance
- weight of terms (for document & query) can be their raw frequency or other measure

T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	...	T_n
W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	...	W_n

Term-by-document matrix



- The collection of documents is represented by a matrix of weights called a **term-by-document matrix**

	term ₁	term ₂	term ₃	...	term _n
D ₁	w ₁₁	w ₁₂	w ₁₃	...	w _{1n}
D ₂	w ₂₁	w ₂₂	w ₂₃	...	w _{2n}
D ₃	w ₃₁	w ₃₂	w ₃₃	...	w _{3n}
...
D _n	w _{n1}	w _{n2}	w _{n3}		w _{nn}

- 1 column = representation of 1 term across all documents
- 1 row = representation of one document
- cell w_{ij} = weight of term i in document j
- note: the matrix is **sparse** !!!

Term-Frequency Vector Space Example



- The document collection:
 - D_1 = “introduction knowledge, speech and language processing, language understanding and the state of the art, future some brief history summary”
 - D_2 = “HMM and speech recognition, speech recognition architecture overview of the HM model and the viterbi algorithm in processing of speech computing probabilities and training a speech recognizer for speech synthesis and human speech recognition summary”
 - D_3 = “language and complexity, how to tell if a language is regular, are English and other languages regular languages? is natural language context-free complexity and human processing summary”
- The query:
 - Q = “speech language processing”

Term-Frequency Vector Space Example

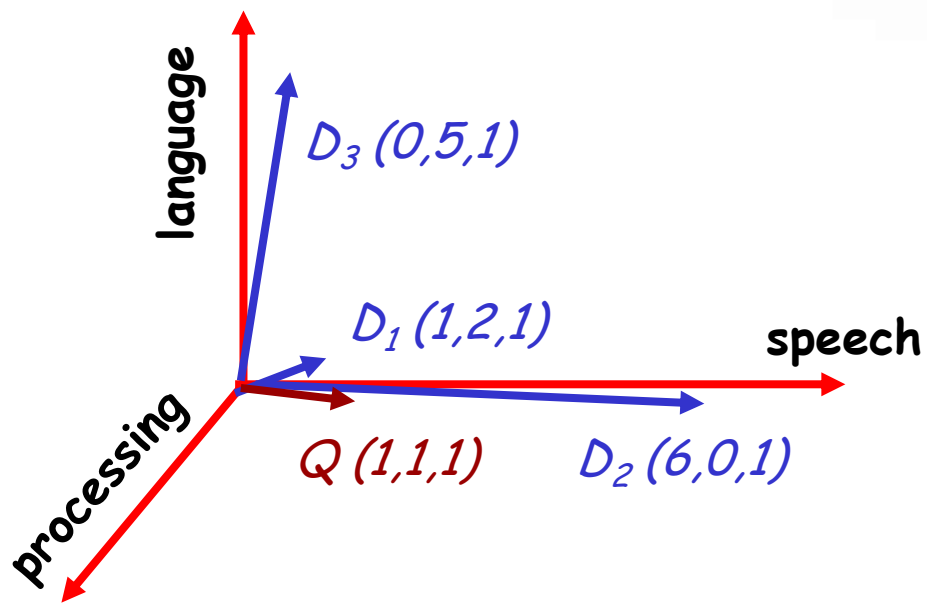


- The document collection:
 - D_1 = “introduction knowledge, **speech** and **language processing**, **language** understanding and the state of the art, future some brief history summary”
 - D_2 = “HMM and **speech** recognition, **speech** recognition architecture overview of the HM model and the viterbi algorithm in **processing** of **speech** computing probabilities and training a **speech** recognizer for **speech** synthesis and human **speech** recognition summary”
 - D_3 = “**language** and complexity, how to tell if a **language** is regular, English and other **language** regular **language**? is natural **language** context-free complexity and human **processing** summary”
- The query:
 - Q = “**speech language processing**”

Term-Frequency Vector Space Example



- using raw term frequencies
- vectors for the documents and the query can be seen as a point in a multi-dimensional space



	...	speech	language	processing	...
D ₁	...	1	2	1	..
D ₂	...	6	0	1	..
D ₃	...	0	5	1	..
Q	...	1	1	1	..

Similarity between Two Vectors (2-Documents)



$$\text{sim}(D_i, Q) = \sum_{i=1}^n (w_{di} \times w_{qi})$$

Similarity Example



	speech	language	processing
D ₁	1	2	1
D ₂	6	0	1
D ₃	0	5	1
Q	1	1	1

$$\text{sim}(D_i, Q) = \sum_{i=1}^n (d_i \times q_i)$$

Q = {speech language processing}

RANK

$$\text{sim}(D_1, Q) = (1 \times 1) + (2 \times 1) + (1 \times 1) = 1 + 2 + 1 = 4$$

3

$$\text{sim}(D_2, Q) = (6 \times 1) + (0 \times 1) + (1 \times 1) = 6 + 0 + 1 = 7$$

1

$$\text{sim}(D_3, Q) = (0 \times 1) + (5 \times 1) + (1 \times 1) = 0 + 5 + 1 = 6$$

2

Vector Normalization



- The longer the document, the more chances it will be retrieved:
 - Because it may contain many of the query's terms
 - It may also contain lots of non-pertinent terms...
- we can normalize raw term frequencies to convert all vectors to a standard length

The cosine measure



- The cosine of 2 vectors (in N dimensions)

inner product

$$\cos(\phi) = \frac{\vec{D} \cdot \vec{Q}}{|\vec{D}| |\vec{Q}|} = \frac{\sum_{i=1}^N d_i q_i}{\sqrt{\sum_{i=1}^N d_i^2} \sqrt{\sum_{i=1}^N q_i^2}}$$

Lengths of The vectors

- Also known as the *normalized inner product*

Similarity between two vectors (2-D)



- In the general case of N-dimensions (N-terms) & normalized vectors

$$\text{sim}(Q, D) = \frac{\sum_{i=1}^N (w_{iq} \times w_{id})}{\sqrt{\sum_{i=1}^N w_{iq}^2} \times \sqrt{\sum_{i=1}^N w_{id}^2}}$$

- which is the cosine of the angle between the vector D and vector Q in N-dimensions

Back to our example



	speech	language	processing
D ₁	1	2	1
D ₂	6	0	1
D ₃	0	5	1
Q	1	1	1

$$\text{sim}(\mathbf{D}, \mathbf{Q}) = \frac{\sum_{i=1}^N d_i q_i}{\sqrt{\sum_{i=1}^N d_i^2} \sqrt{\sum_{i=1}^N q_i^2}}$$

Q = {speech language processing}

$$\text{sim}(\mathbf{D}_1, \mathbf{Q}) = \frac{(1 \times 1) + (2 \times 1) + (1 \times 1)}{\sqrt{(1^2 + 2^2 + 1^2)} \times \sqrt{(1^2 + 1^2 + 1^2)}} = \frac{1 + 2 + 1}{\sqrt{6} \times \sqrt{3}} = 0.943$$

$$\text{sim}(\mathbf{D}_2, \mathbf{Q}) = \frac{(6 \times 1) + (0 \times 1) + (1 \times 1)}{\sqrt{(6^2 + 0^2 + 1^2)} \times \sqrt{(1^2 + 1^2 + 1^2)}} = \frac{6 + 0 + 1}{\sqrt{37} \times \sqrt{3}} = 0.664$$

$$\text{sim}(\mathbf{D}_3, \mathbf{Q}) = \frac{(0 \times 1) + (5 \times 1) + (1 \times 1)}{\sqrt{(0^2 + 5^2 + 1^2)} \times \sqrt{(1^2 + 1^2 + 1^2)}} = \frac{0 + 5 + 1}{\sqrt{26} \times \sqrt{3}} = 0.680$$

RANK

1

3

2

Components of Similarity



- The “inner product” is the key to the similarity function

$$\vec{d}_j \cdot \vec{d}_k = \sum_{i=1}^n w_{i,j} w_{i,k}$$

Example:

$$\begin{aligned} & [1 \ 2 \ 3 \ 0 \ 2] \cdot [2 \ 0 \ 1 \ 0 \ 2] \\ & = 1 \times 2 + 2 \times 0 + 3 \times 1 + 0 \times 0 + 2 \times 2 = 9 \end{aligned}$$

- The denominator handles document length normalization

$$|\vec{d}_j| = \sqrt{\sum_{i=1}^n w_{i,j}^2}$$

Example:

$$\begin{aligned} & |[1 \ 2 \ 3 \ 0 \ 2]| \\ & = \sqrt{1 + 4 + 9 + 0 + 4} = \sqrt{18} \approx 4.24 \end{aligned}$$

Components of Similarity



Query Vector

$$\text{sim}(d_j, d_k) = \frac{\vec{d}_j \cdot \vec{d}_k}{\|\vec{d}_j\| \|\vec{d}_k\|} = \frac{\sum_{i=1}^n w_{i,j} w_{i,k}}{\sqrt{\sum_{i=1}^n w_{i,j}^2} \sqrt{\sum_{i=1}^n w_{i,k}^2}}$$

Document Vector

Inner Product
Length Normalization

The diagram illustrates the components of the similarity formula. Red dashed ovals highlight the inner product $\vec{d}_j \cdot \vec{d}_k$ and the length normalization terms $\sqrt{\sum_{i=1}^n w_{i,j}^2}$ and $\sqrt{\sum_{i=1}^n w_{i,k}^2}$. Red arrows point from the labels 'Query Vector' and 'Document Vector' to the corresponding vectors in the formula.

How do we weight doc terms?



■ Here's the intuition:

■ Terms that appear often in a document should get high weights

- The more often a document contains the term “dog”, the more likely that the document is “about” dogs.

■ Terms that appear in many documents should get low weights

- Words like “the”, “a”, “of” appear in (nearly) all documents.

■ How do we capture this mathematically?

■ Term frequency

■ Inverse document frequency

Term weights



- We have used term frequency as the weights
- But the core of most weighting functions are **tf** & **df**:
 - **tf_{ij}** **term frequency**
 - frequency of a term *i* in document *j*
 - if a term appears often in a document, then it describes well the document contents
 - **df_i** **document frequency**
 - number of documents in the collection containing the term *i*
 - if a term appears in many documents, then it is not useful for distinguishing a document
 - used to compute **idf**

Weighting Function



- The most widely used family of weighting functions **tf.idf**
 - let: n = number of documents in the collection
 - Inverse Document Frequency for term i (**idf**) (*measures weight of term i for the query*)

$$\text{idf}_i = \log\left(\frac{n}{\text{df}_i}\right)$$

If $n = 1000$	df_i	$\log(n / \text{df}_i)$	comments
	1000	$\text{Log}(1) = 0$	term i is ignored! (<i>it appears in all docs</i>)
	10	$\text{Log}(100) = 2$	term i has weight of 2 <u>in the query</u>
	1	$\text{Log}(1000) = 3$	term i has weight of 3 <u>in the query</u>

- weight of term i in document d is:
 $w_{id} = \text{tf}_{id} \times \text{idf}_i$

TF.IDF Term Weighting



- Simple, yet effective!

$$w_{i,j} = \text{tf}_{i,j} \cdot \log \frac{N}{n_i}$$

$w_{i,j}$ weight assigned to term i in document j

$\text{tf}_{i,j}$ number of occurrence of term i in document j

N number of documents in entire collection

n_i number of documents with term i

TF.IDF Example



	<i>tf</i>					W_{ij}			
	1	2	3	4	<i>idf</i>	1	2	3	4
complicated			5	2	0.301			1.51	0.60
contaminated	4	1	3		0.125	0.50	0.13	0.38	
fallout	5		4	3	0.125	0.63		0.50	0.38
information	6	3	3	2	0.000				
interesting		1			0.602		0.60		
nuclear	3		7		0.301	0.90		2.11	
retrieval		6	1	4	0.125		0.75	0.13	0.50
siberia	2				0.602	1.20			

Normalizing Document Vectors



- Recall our similarity function:

$$\text{sim}(d_j, d_k) = \frac{\vec{d}_j \cdot \vec{d}_k}{\|\vec{d}_j\| \|\vec{d}_k\|} = \frac{\sum_{i=1}^n w_{i,j} w_{i,k}}{\sqrt{\sum_{i=1}^n w_{i,j}^2} \sqrt{\sum_{i=1}^n w_{i,k}^2}}$$

- Normalize document vectors in advance
 - Use the “cosine normalization” method: divide each term weight through by length of vector

TF.IDF Example

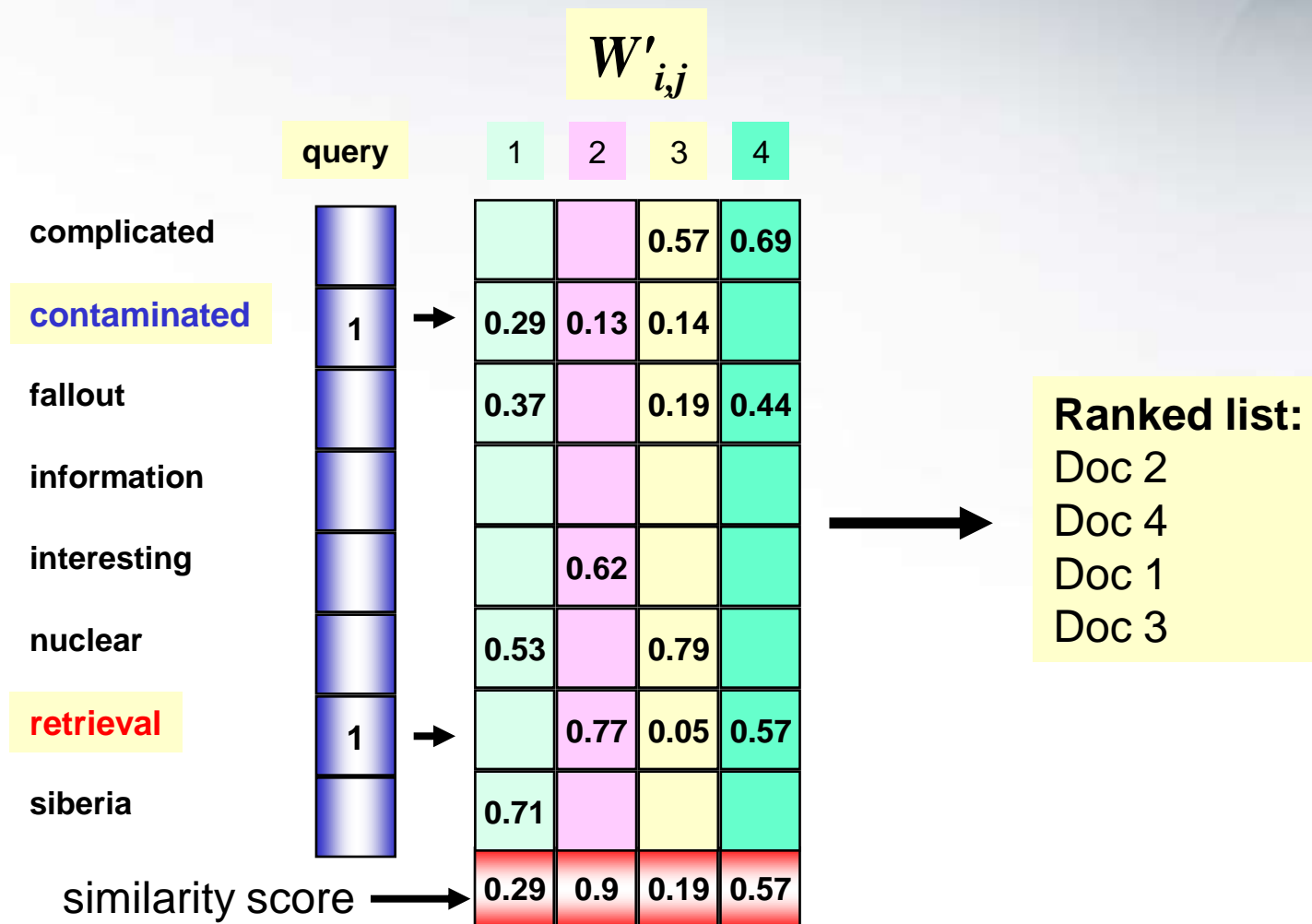


	<i>tf</i>					W_{ij}					W'_{ij}			
	1	2	3	4	<i>idf</i>	1	2	3	4		1	2	3	4
complicated			5	2	0.301			1.51	0.60				0.57	0.69
contaminated	4	1	3		0.125	0.50	0.13	0.38			0.29	0.13	0.14	
fallout	5		4	3	0.125	0.63		0.50	0.38		0.37		0.19	0.44
information	6	3	3	2	0.000									
interesting		1			0.602		0.60					0.62		
nuclear	3		7		0.301	0.90		2.11			0.53		0.79	
retrieval		6	1	4	0.125		0.75	0.13	0.50			0.77	0.05	0.57
siberia	2				0.602	1.20					0.71			
						Length	1.70	0.97	2.67	0.87				

Retrieval Example



Query: contaminated retrieval



Do we need to normalize the query vector?

Summary



- Boolean retrieval is powerful in the hands of a trained searcher
- Ranked retrieval is preferred in other circumstances
- Key ideas in the vector space model
 - Goal: find documents most similar to the query
 - Geometric interpretation: measure similarity in terms of angles between vectors in high dimensional space
 - Documents weights are some combinations of TF, DF, and Length
 - Length normalization is critical
 - Similarity is calculated via the inner product