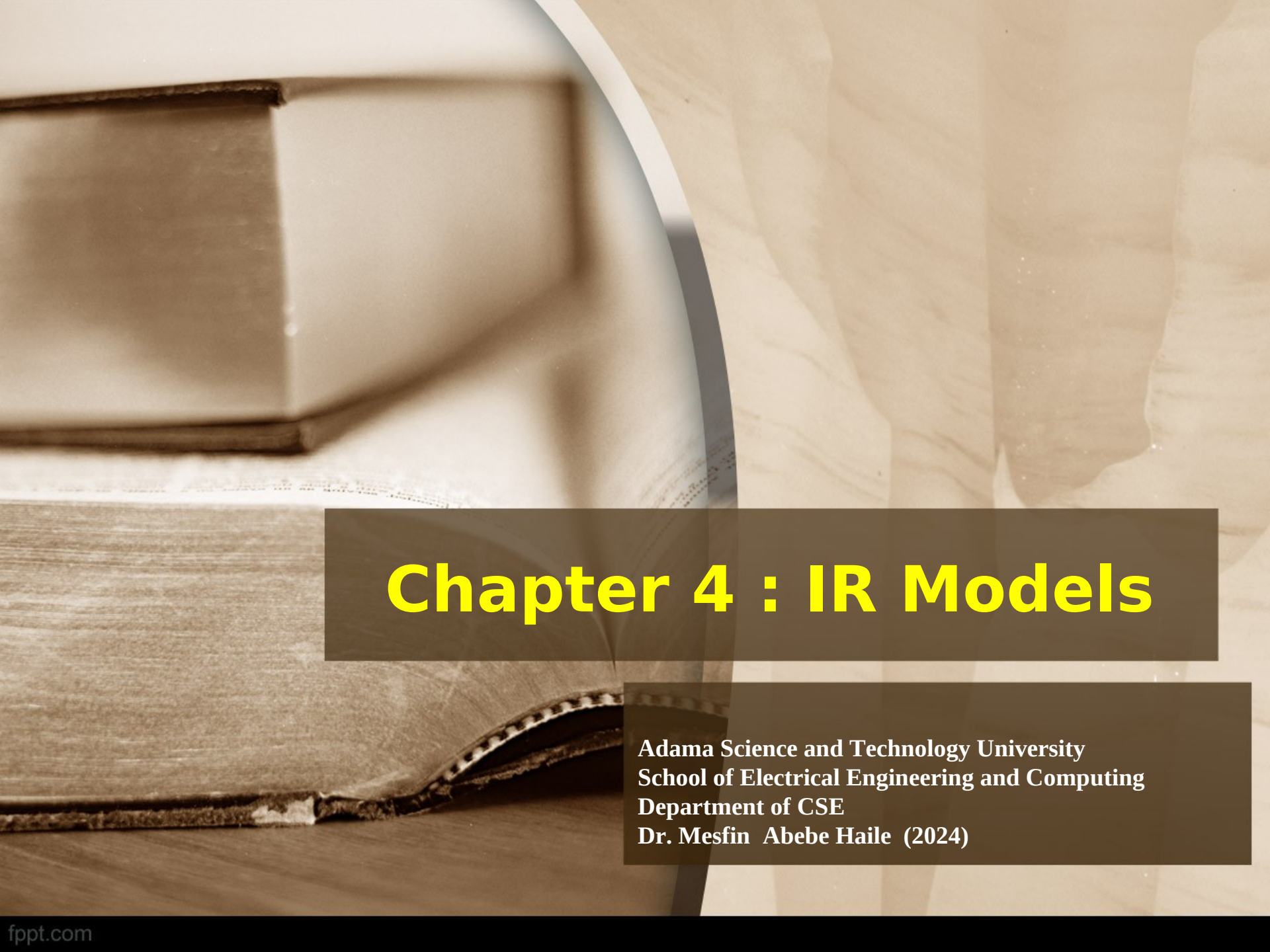




Chapter 4 : IR Models

**Adama Science and Technology University
School of Electrical Engineering and Computing
Department of CSE
Dr. Mesfin Abebe Haile (2024)**

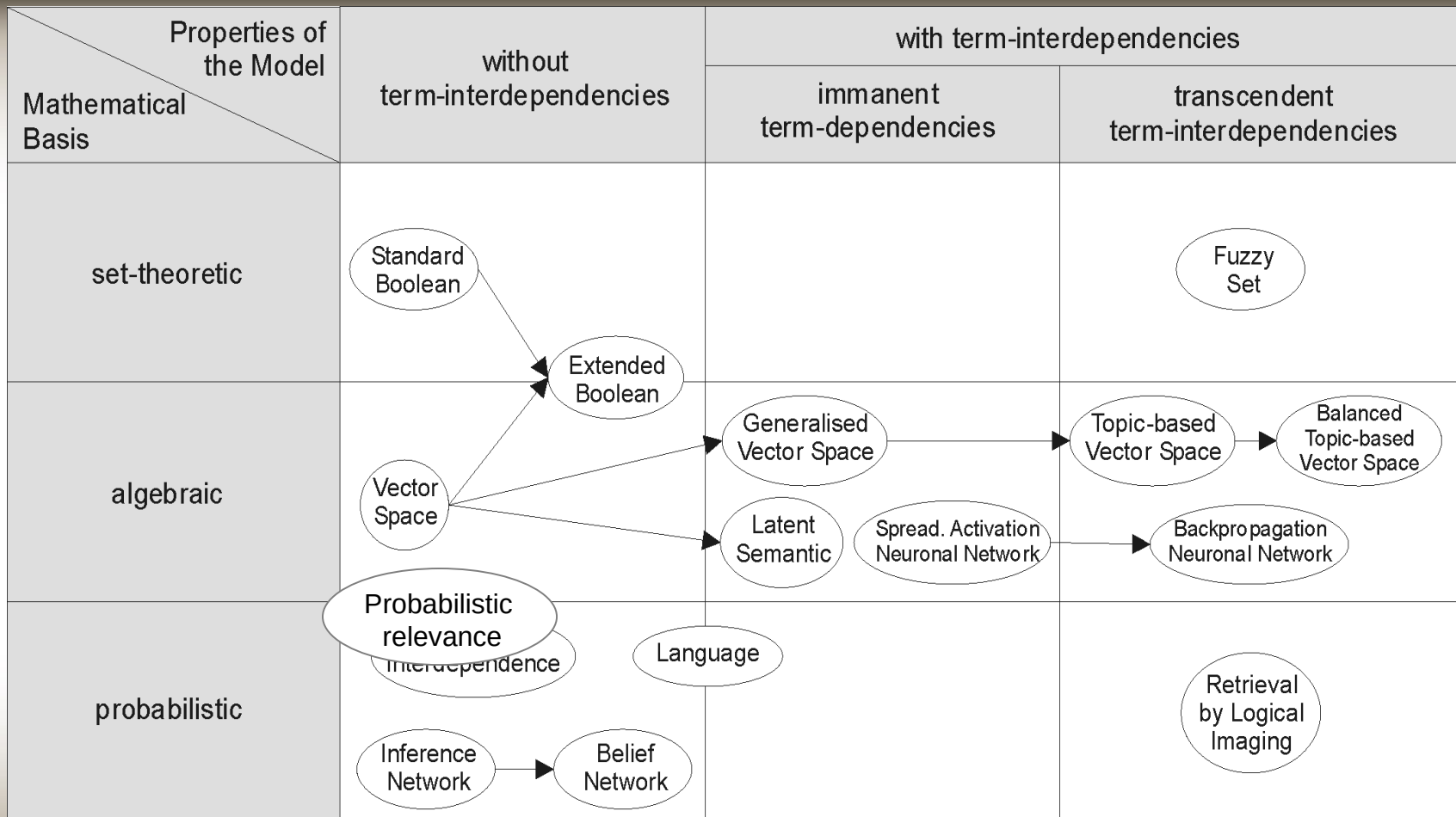
IR Models - Basic Concepts

- ❖ Word evidence:
- ❖ IR systems usually adopt *index terms to index* and *retrieve documents*.
- ❖ Each document is represented by a set of *representative keywords* or *index terms* (called Bag of Words).
- ❖ An *index term* is a document word useful for *remembering* the *document main themes*.
- ❖ Not all *terms are equally* useful for representing the *document contents*:
- ❖ *Less frequent terms* allow identifying a narrower set of documents.
- ❖ But *no ordering information* is attached to the *Bag of Words* identified from the document collection.

IR Models - Basic Concepts

- ❖ One central problem regarding IR systems is the issue of predicting *the degree of relevance of documents* for a *given query*.
 - ❖ Such a decision is usually *dependent on a ranking algorithm* which attempts to establish a *simple ordering of the documents retrieved*.
 - ❖ Documents *appearing at the top of this ordering* are considered to be more likely to be *relevant*.
- ❖ Thus *ranking algorithms* are at the *core of IR systems*.
 - ❖ The *IR models determine* the predictions of what is *relevant and what is not*, based on the notion of relevance implemented by the system.

Alternative IR models



IR Models - Basic Concepts

- ❖ After *preprocessing*, N distinct terms (Bag of words) remain which are unique terms that form the *VOCABULARY*. Let:
 - ❖ k_i be *an index term i* & d_j be a *document j* .
 - ❖ $K = (k_1, k_2, \dots, k_N)$ is the *set of all index terms*.
- ❖ Each term, i , in a document or query j , is given a *real-valued weight*, w_{ij} .
 - ❖ w_{ij} is a weight associated with (k_i, d_j) . If $w_{ij} = 0$, it indicates that term *does not belong to document d_j* .
- ❖ The weight w_{ij} *quantifies the importance of the index term* for describing the *document contents*.
 - ❖ $\text{vec}(d_j) = (w_{1j}, w_{2j}, \dots, w_{tj})$ is a *weighted vector associated* with the document d_j .

Mapping Documents & Queries

- ❖ Represent *both documents and queries as N-dimensional vectors* in a *term-document matrix*, which shows occurrence of terms in the *document collection* or *query*.

– E.g. $d_j = (t_{1,j}, t_{2,j}, \dots, t_{N,j})$; $q_k = (t_{1,k}, t_{2,k}, \dots, t_{N,k})$

- ❖ An entry in the matrix corresponds to the “*weight*” of a term in the *document*; zero means the *term doesn't exist in the document*.

	T_1	T_2		T_N
D_1	w_{11}	w_{12}	...	w_{1N}
D_2	w_{21}	w_{22}	...	w_{2N}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
D_M	w_{M1}	w_{M2}	...	w_{MN}
Q_i	w_{i1}	w_{i2}	...	w_{iN}

- ❖ Document collection is mapped to *term-by-document matrix*.
- ❖ View as vector in *multidimensional space*.
 - ❖ Nearby vectors are related.
- ❖ Normalize for *vector length* to avoid the effect of *document length*.

Weighting Terms in Vector Space

- ❖ The *importance of the index terms* is represented by *weights associated to them*.
- ❖ **Problem**: to show the importance of the index term for describing the document/query contents, *what weight we can assign?*
- ❖ **Solution 1**: Binary weights: $t=1$ if presence, 0 otherwise.
 - ❖ *Similarity*: number of terms in common.
- ❖ **Problem**: Not all terms *equally interesting*.
 - ❖ E.g. the vs. dog vs. cat
- ❖ **Solution**: Replace *binary weights* with *non-binary weights*.

$$d_j = (w_{1,j}, w_{2,j}, \dots, w_{N,j}); q_k = (w_{1,k}, w_{2,k}, \dots, w_{N,k})$$

How to evaluate Models?

- ❖ We need to investigate *what procedures they follow* and *what techniques* they used for:
 - ❖ Are they using *binary* or *non-binary weighting* for measuring importance of terms in documents.
 - ❖ Are they using *similarity measurements*?
 - ❖ Are they applying *partial matching*?
- ❖ Are they performing *Exact matching* or *Best matching* for *document retrieval*?
- ❖ Any *Ranking mechanism*?

The Boolean Model

- ❖ Boolean model is a *simple model* based on set theory.
 - ❖ The Boolean model imposes a *binary criterion* for *deciding relevance*.

- ❖ Terms are either *present* or *absent*. Thus,

$$w_{ij} \in \{0,1\}$$

- ❖ $\text{sim}(q,d_j) = 1$, if document satisfies the boolean query,
0 otherwise

- ❖ **Note** that, no weights assigned in-between 0 and 1, just only *values 0 or 1*.

	T_1	T_2	...	T_N
D_1	w_{11}	w_{12}	...	w_{1N}
D_2	w_{21}	w_{22}	...	w_{2N}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
D_M	w_{M1}	w_{M2}	...	w_{MN}

The Boolean Model: Example

❖ Given the following docs *determine the documents retrieved* by the *Boolean model* based IR system.

❖ Index Terms: $K_1, \dots, K_8$.

❖ Documents:

1. $D_1 = \{K_1, K_2, K_3, K_4, K_5\}$

2. $D_2 = \{K_1, K_2, K_3, K_4\}$

3. $D_3 = \{K_2, K_4, K_6, K_8\}$

4. $D_4 = \{K_1, K_3, K_5, K_7\}$

5. $D_5 = \{K_4, K_5, K_6, K_7, K_8\}$

6. $D_6 = \{K_1, K_2, K_3, K_4\}$

• Query: $K_1 \wedge (K_2 \vee \neg K_3)$

• **Answer:** $\{D_1, D_2, D_4, D_6\} \cap (\{D_1, D_2, D_3, D_6\} \cup \{D_3, D_5\})$
 $= \{D_1, D_2, D_6\}$

The Boolean Model: Further Example

- ❖ Given the following *three documents*, Construct *Term – document matrix* and find the relevant documents retrieved by the Boolean model for given query.
 - ❖ D1: “Shipment of gold damaged in a fire”
 - ❖ D2: “Delivery of silver arrived in a silver truck”
 - ❖ D3: “Shipment of gold arrived in a truck”
 - ❖ Query: “gold silver truck”

Table below shows document –term (t_i) matrix

	arrive	damage	deliver	fire	gold	silver	ship	truck
D1								
D2								
D3								
query								

- ❖ Find the *relevant documents* for the *queries*:
 - (a) gold delivery
 - (b) ship gold
 - (c) silver truck

Exercise

- ❖ Given the following *four documents* with the following *contents*:
 - ❖ $D_1 = \text{“computer information retrieval”}$
 - ❖ $D_2 = \text{“computer retrieval”}$
 - ❖ $D_3 = \text{“information”}$
 - ❖ $D_4 = \text{“computer information”}$
- ❖ What are the *relevant documents retrieved* for the *queries*:
 - ❖ $Q_1 = \text{“information} \wedge \text{retrieval”}$
 - ❖ $Q_2 = \text{“information} \wedge \neg \text{computer”}$

Drawbacks of the Boolean Model

- ❖ Retrieval based on binary decision criteria with *no notion of partial matching*,
- ❖ *No ranking* of the documents is provided (absence of a grading scale),
- ❖ *Information need* has to be translated into a *Boolean expression* which most users find *awkward*,
- ❖ The *Boolean queries formulated* by the users are most often *too simplistic*.
 - ❖ As a consequence, the Boolean model frequently returns either *too few or too many documents* in response to a user query.

Vector-Space Model

- ❖ This is the *most commonly used strategy* for measuring *relevance* of documents for a given query. This is because,
 - ❖ Use of binary weights is *too limiting*,
 - ❖ *Non-binary weights* provide consideration for *partial matches*.
- ❖ These term weights are used to *compute a degree of similarity between a query and each document*.
 - ❖ Ranked set of documents provides for *better matching*.
- ❖ The idea behind VSM is that:
 - ❖ The *meaning of a document* is conveyed by the *words used* in that document.

Vector-Space Model

To find *relevant documents* for a given query,

- ❖ First, map documents and queries into *term-document vector space*.

 - ❖ Note that queries are considered as *short document*.

- ❖ Second, in the vector space, queries and documents are *represented as weighted vectors*, w_{ij} .

 - ❖ There are *different weighting technique*; the most widely used one is computing *tf*idf* for each term.

- ❖ Third, *similarity measurement* is used to *rank documents* by the closeness of their vectors to the query.

 - ❖ Documents are ranked by *closeness to the query*.

 - ❖ Closeness is determined by a *similarity score calculation*.

Term-document Matrix.

- ❖ A collection of *n documents and query* can be represented in the vector space model by a *term-document matrix*.
- ❖ An entry in the matrix corresponds to the “*weight*” of a term in the document;
- ❖ Zero means the term has *no significance in the document* or it simply doesn't exist in the document. Otherwise, $w_{ij} > 0$ whenever $k_i \in d_j$.

$$\begin{pmatrix} & T_1 & T_2 & \dots & T_N \\ D_1 & w_{11} & w_{21} & \dots & w_{N1} \\ D_2 & w_{12} & w_{22} & \dots & w_{N2} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_M & w_{1M} & w_{2M} & \dots & w_{NM} \end{pmatrix}$$

Computing Weights

- ❖ How to compute weights for *term i in document j* and *query q*; w_{ij} and w_{iq} ?
- ❖ A good weight must take into account *two effects*:
 - ❖ Quantification of *intra-document* contents (*similarity*).
 - ❖ tf factor, the *term frequency* within a document.
 - ❖ Quantification of *inter-documents* separation (*dissimilarity*).
 - ❖ idf factor, the *inverse document frequency*.
- ❖ As a result of which most IR systems are using *tf*idf weighting technique*:

$$w_{ij} = tf(i,j) * idf(i)$$

Computing Weights

- ❖ Let: N be *the total number of documents* in the collection,
 - ❖ n_i be the *number of documents* which contain t_i .
 - ❖ $\text{freq}(i,j)$ *raw frequency* of t_i within d_j .
- ❖ A *normalized $\text{tf}(i,j)$* factor is given by
$$\text{tf}(i,j) = \text{freq}(i,j) / \max(\text{freq}(k,j))$$
 - ❖ Where the maximum is computed *over all terms* which occur within the document d_j .
- ❖ The *idf* factor is computed as
$$\text{idf}(i) = \log(N/n_i)$$
 - ❖ The log is used to *make the values* of *tf* and *idf* comparable. It can also be interpreted as the *amount of information associated* with the term t_i .
- ❖ A *normalized $\text{tf}*\text{idf}$ weight* is given by:
$$w_{ij} = \text{freq}(i,j) / \max(\text{freq}(k,j)) * \log(N/n_i)$$

Computing Weights

Query:

- ❖ Users *query* is typically treated *as a document* and also *tf-idf weighted*.
- ❖ For the *query term weights*, a suggestion is:

$$w_{iq} = (0.5 + [0.5 * \text{freq}(i,q) / \max(\text{freq}(k,q))]) * \log(N/n_i)$$

- ❖ The *vector space model* with *tf*idf weights* is a good ranking strategy with general collections,
- ❖ The *vector space model* is usually as good as the known ranking alternatives.
- ❖ It is also *simple and fast* to compute.

Example: Computing Weights

- ❖ A collection includes 10,000 documents:
 - ❖ The *term A* appears *20 times* in a *particular document*,
 - ❖ The *maximum appearance* of any term in this document is 50,
 - ❖ The *term A* appears in *2,000 of the collection* documents.
- ❖ Compute *TF*IDF weight*?
 - ❖ $tf(i,j) = freq(i,j) / \max(freq(k,j)) = 20/50 = 0.4$
 - ❖ $idf(i) = \log(N/n_i) = \log(10,000/2,000) = \log(5) = 2.32$
 - ❖ $w_{ij} = f(i,j) * \log(N/n_i) = 0.4 * 2.32 = 0.928$

Similarity Measure

- ❖ A *similarity measure* is a function that computes the *degree of similarity* between *two vectors*.
- ❖ Using a *similarity measure* between the *query* and each *document*:
 - ❖ It is possible to *rank the retrieved documents* in the order of *presumed relevance*.
 - ❖ It is possible to enforce a certain threshold so that we can *control the size of the retrieved set* of documents.

Similarity Measures

$$|Q \cap D|$$

❖ Dot Product (Simple matching)

$$2 \frac{|Q \cap D|}{|Q| + |D|}$$

❖ Dice's Coefficient

$$\frac{|Q \cap D|}{|Q \cup D|}$$

❖ Jaccard's Coefficient

$$\frac{|Q \cap D|}{|Q \cup D|}$$

$$\frac{|Q \cap D|}{|Q|^{\frac{1}{2}} \times |D|^{\frac{1}{2}}}$$

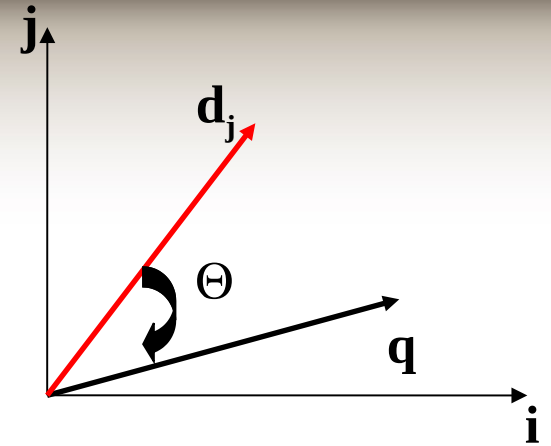
❖ Cosine Coefficient

$$\frac{|Q \cap D|}{\min(|Q|, |D|)}$$

❖ Overlap Coefficient

Similarity Measure

❖ $\text{Sim}(q, d_j) = \cos(\Theta)$



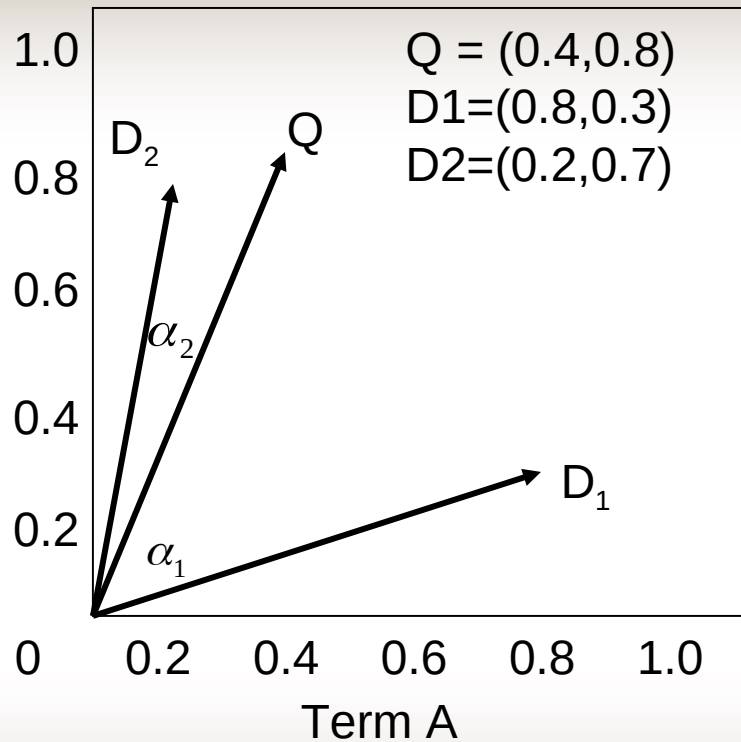
$$\text{sim}(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| |\vec{q}|} = \frac{\sum_{i=1}^n w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^n w_{i,j}^2} \sqrt{\sum_{i=1}^n w_{i,q}^2}}$$

❖ Since $w_{ij} > 0$ and $w_{iq} > 0$, $0 \leq \text{sim}(q, d_j) \leq 1$

❖ A document is *retrieved* even if it *matches the query terms* only partially.

Vector Space with Term Weights and Cosine Matching

Term B



$$D_i = (d_{i1}, w_{di1}; d_{i2}, w_{di2}; \dots; d_{it}, w_{dit})$$

$$Q = (q_{i1}, w_{qi1}; q_{i2}, w_{qi2}; \dots; q_{it}, w_{qit})$$

$$\text{sim}(Q, D_i) = \frac{\sum_{j=1}^t w_{q_j} w_{d_{ij}}}{\sqrt{\sum_{j=1}^t (w_{q_j})^2 \sum_{j=1}^t (w_{d_{ij}})^2}}$$

$$\begin{aligned} \text{sim}(Q, D2) &= \frac{(0.4 \cdot 0.2) + (0.8 \cdot 0.7)}{\sqrt{[(0.4)^2 + (0.8)^2] [(0.2)^2 + (0.7)^2]}} \\ &= \frac{0.64}{\sqrt{0.42}} = 0.98 \end{aligned}$$

$$\text{sim}(Q, D_1) = \frac{.56}{\sqrt{0.58}} = 0.74$$

Vector-Space Model: Example

- ❖ Suppose user query for: $Q = \text{“gold silver truck”}$. The database collection consists of *three documents* with the following content.
 - ❖ D1: “Shipment of gold damaged in a fire”
 - ❖ D2: “Delivery of silver arrived in a silver truck”
 - ❖ D3: “Shipment of gold arrived in a truck”
- ❖ Show retrieval results in *ranked order*?
 - ❖ Assume that *full text terms* are used *during indexing*, without removing common terms, *stop words*, & also no terms are *stemmed*.
 - ❖ Assume that *content-bearing terms* are selected during *indexing*.
 - ❖ Also compare your result *with or without normalizing term frequency*.

Vector-Space Model: Example

Terms	Q	Counts TF			DF	IDF	$W_i = TF \cdot IDF$			
		D1	D2	D3			Q	D1	D2	D3
a	0	1	1	1	3	0	0	0	0	0
arrived	0	0	1	1	2	0.176	0	0	0.176	0.176
damaged	0	1	0	0	1	0.477	0	0.477	0	0
delivery	0	0	1	0	1	0.477	0	0	0.477	0
fire	0	1	0	0	1	0.477	0	0.477	0	0
gold	1	1	0	1	2	0.176	0.176	0.176	0	0.176
in	0	1	1	1	3	0	0	0	0	0
of	0	1	1	1	3	0	0	0	0	0
silver	1	0	2	0	1	0.477	0.477	0	0.954	0
shipment	0	1	0	1	2	0.176	0	0.176	0	0.176
truck	1	0	1	1	2	0.176	0.176	0	0.176	0.176

Vector-Space Model

Terms	Q	D1	D2	D3
a	0	0	0	0
arrived	0	0	0.176	0.176
damaged	0	0.477	0	0
delivery	0	0	0.477	0
fire	0	0.477	0	0
gold	0.176	0.176	0	0.176
in	0	0	0	0
of	0	0	0	0
silver	0.477	0	0.954	0
shipment	0	0.176	0	0.176
truck	0.176	0	0.176	0.176

Vector-Space Model: Example

- ❖ Compute similarity using cosine $\text{Sim}(q, d_1)$.
- ❖ First, for *each document* and *query*, compute all *vector lengths* (zero terms ignored).

$$|d_1| = \sqrt{0.477^2 + 0.477^2 + 0.176^2 + 0.176^2} = \sqrt{0.517} = 0.719$$

$$|d_2| = \sqrt{0.176^2 + 0.477^2 + 0.176^2 + 0.176^2} = \sqrt{1.2001} = 1.095$$

$$|d_3| = \sqrt{0.176^2 + 0.176^2 + 0.176^2 + 0.176^2} = \sqrt{0.124} = 0.352$$

$$|q| = \sqrt{0.176^2 + 0.471^2 + 0.176^2} = \sqrt{0.23896} = 0.538$$

- ❖ Next, *compute dot products* (zero products ignored):

- ❖ $Q \cdot d_1 = 0.176 \cdot 0.176 = 0.0310$

- ❖ $Q \cdot d_2 = 0.954 \cdot 0.477 + 0.176 \cdot 0.176 = 0.4862$

- ❖ $Q \cdot d_3 = 0.176 \cdot 0.167 + 0.176 \cdot 0.167 = 0.0620$

Vector-Space Model: Example

- ❖ Now, *compute similarity score*:
 - ❖ $\text{Sim}(q, d_1) = (0.0310) / (0.538 * 0.719) = 0.0801$
 - ❖ $\text{Sim}(q, d_2) = (0.4862) / (0.538 * 1.095) = 0.8246$
 - ❖ $\text{Sim}(q, d_3) = (0.0620) / (0.538 * 0.352) = 0.3271$
- ❖ Finally, we *sort and rank documents* in descending order according to the *similarity scores*:
 - ❖ Rank 1: Doc 2 = 0.8246
 - ❖ Rank 2: Doc 3 = 0.3271
 - ❖ Rank 3: Doc 1 = 0.0801
 - ❖ **Exercise**: using *normalized TF*, rank documents using *cosine similarity measure*?
 - ❖ **Hint**: Normalize TF of term i in doc j using *max frequency of a term* k in *document* j .

Vector-Space Model

❖ Advantages:

- ❖ *Term-weighting* improves quality of the answer set since it displays in *ranked order*,
- ❖ *Partial matching* allows retrieval of documents that approximate the *query conditions*,
- ❖ *Cosine ranking formula sorts documents* according to *degree of similarity* to the query.

❖ Disadvantages:

- ❖ Assumes *independence of index terms* (??)

Exercise 1

- ❖ Consider these documents:
 - ❖ *Doc 1*: breakthrough drug for schizophrenia
 - ❖ *Doc 2*: new schizophrenia drug
 - ❖ *Doc 3*: new approach for treatment of schizophrenia
 - ❖ *Doc 4*: new hopes for schizophrenia patients
 - ❖ Draw the *term-document incidence matrix* for this document collection.
 - ❖ Draw the *inverted index representation* for this collection.
- ❖ For the document collection shown above, what are the *returned results for the queries*:
 - ❖ schizophrenia AND drug
 - ❖ for AND NOT(drug OR approach)

Question & Answer



Thank You !!!