Bilgiye Erişim Sistemleri

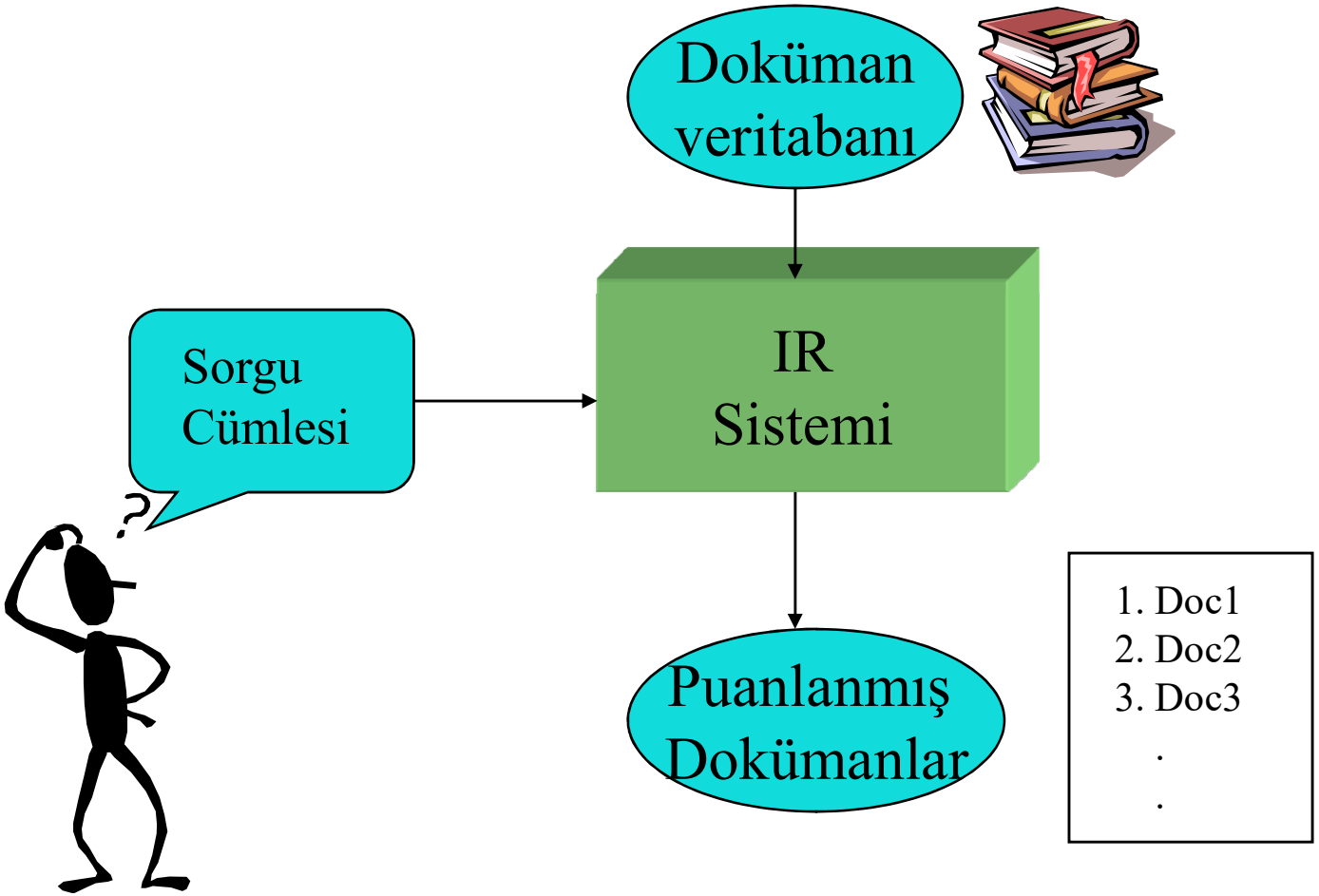
(Information Retrieval Systems-IE)

Prof.Dr.Banu Diri

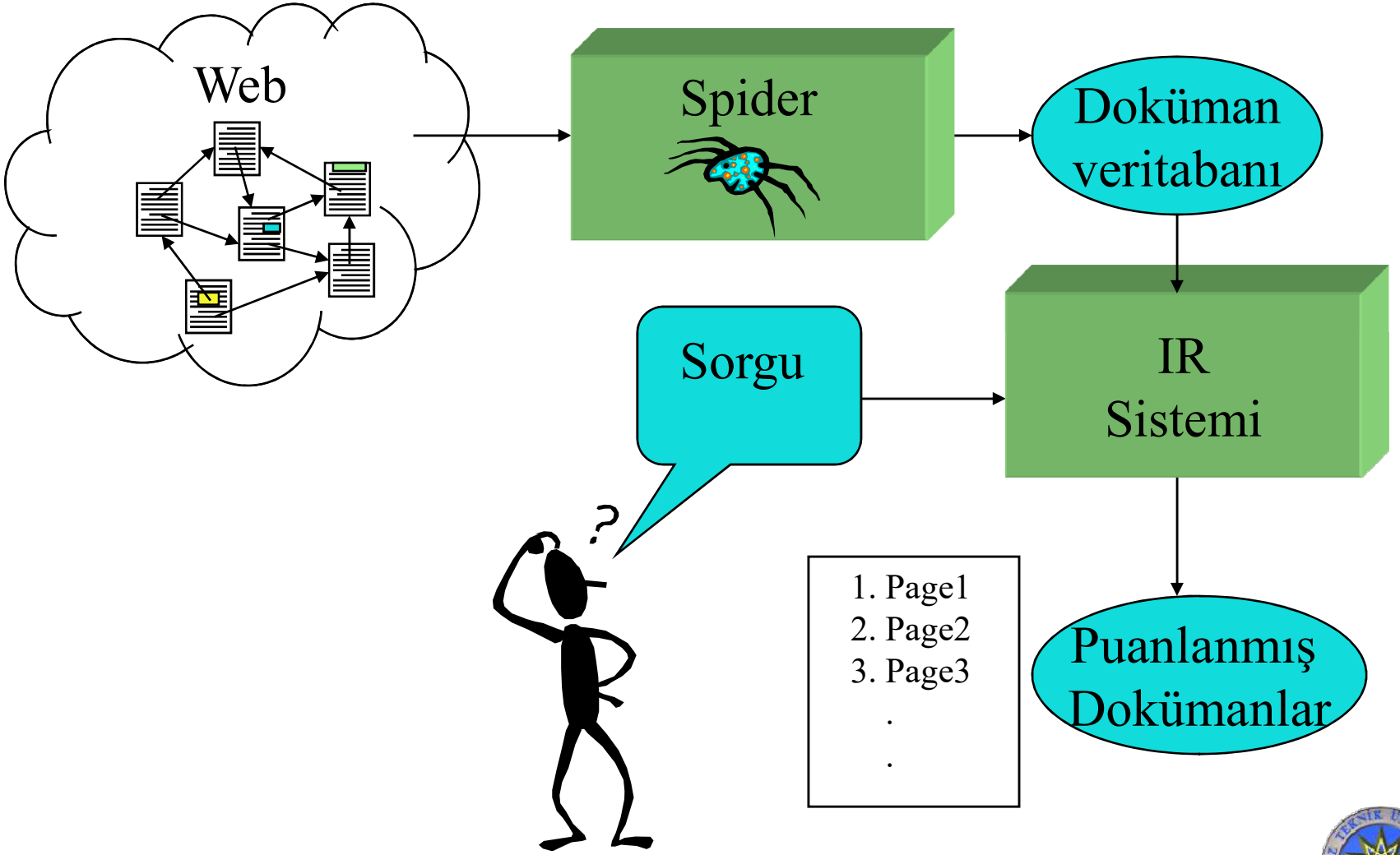
Örnek IR Sistemleri

- Kütüphane veritabanları
  anahtar kelime, başlık, yazar, konu vs. ile büyük veritabanlarında arama (www.library.unt.edu)
- Metin Tabanlı Arama Motorları (Google, Yahoo, Altavista vs.)
  Anahtar kelimelerle arama
- Multimedya Arama (QBIC-IBM's Query By Image Content, WebSeek-**A Content-Based Image and Video Catalog and Search Tool for the Web**)
  Görsel öğelerle arama (şekil, renk vs.)
- Soru Cevaplama Sistemleri (AskJeeves, Answerbus)
  Doğal dille arama
- Hakia (http://www.hakia.com/)

IR Sistem Mimarisi



Arama Motoru Mimarisi



IR –Modelleri

Etkili bir IR yapmak için, dokümanlar uygun bir gösterim formuna dönüştürülmelidir. Modeller iki boyutlu olarak ele alınır.

❖ **Matemetiksel Modeller** (*Set-theoretic models, Algebraic models, Probabilistic models)*

❖ Modelin özellikleri (*without term-interdependencies, with term-interdependencies)*

Matemetiksel Modeller

Küme-Kuramsal Modeller (Set-theoretic models)

Dokümanlar kelimeler veya ifadelerin kümesi olarak gösterilir.

En çok bilinen modeller:

- **Standard Boolean Model**
- Extended Boolean Model
- Fuzzy Retrieval

Standard Boolean Model

Boolean Information Retrieval (BIR) , Boolean Logic ve Klasik Küme teorisine dayanır.

Retrieval is based on whether or not the documents contain the query terms.

- T = {t1, t2, ..., tj, ..., tm} of elements called index terms (e.g. words or expressions)
- D = {D1, ..., Di, ..., Dn}, where Di is an element of the powerset of T of elements called documents.
- Given a Boolean expression - in a normal form - Q called a query as follows:
- Q = (Wi OR Wk OR ...) AND ... AND (Wj OR Ws OR ...), with Wi=ti, Wk=tk, Wj=tj, Ws=ts, or Wi=NON ti, Wk=NON tk, Wj=NON tj, Ws=NONts where ti means that the term ti is present in document Di, whereas NON ti means that it is not.

Retrieval, consisting of two steps, is defined as follows:

1. The sets Sj of documents are obtained that contain or not term tj (depending on whether Wj=tj or Wj=NON tj) : Sj = {Di|Wj element of Di}
2. Those documents are retrieved in response to Q which are the result of the corresponding sets operations, i.e. the answer to Q is as follows:

UNION (INTERSECTION Sj)

Example

Let the set of original (real) documents be, for example D = {D1, D2, D3} where

D1 = Bayes' Principle: The principle that, in estimating a parameter, one should initially assume that each possible value has equal probability (a uniform prior distribution).

D2 = Bayesian Decision Theory: A mathematical theory of decision-making which presumes utility and probability functions, and according to which the act to be chosen is the Bayes act, i.e. the one with highest Subjective Expected Utility. If one had unlimited time and calculating power with which to make every decision, this procedure would be the best way to make any decision.

D3 = Bayesian Epistemology: A philosophical theory which holds that the epistemic status of a proposition (i.e. how well proven or well established it is) is best measured by a probability and that the proper way to revise this probability is given by Bayesian conditionalisation or similar procedures. A Bayesian epistemologist would use probability to define, and explore the relationship between, concepts such as epistemic status, support or explanatory power.

Let the set T of terms be:

T = {t1 = Bayes' Principle, t2 = probability, t3 = decision-making, t4 = Bayesian Epistemology}

Then, the set D of documents is as follows:

D = {D1, D2, D3} where
D1 = {Bayes' Principle, probability}
D2 = {probability, decision-making}
D3 = {probability, Bayesian Epistemology}

Let the query Q be: **Q =** probability ***AND*** decision-making

1. Firstly, the following sets S1 and S2 of documents Di are obtained (retrieved):

S1 = {D1, D2, D3} →*probability*
S2 = {D2} →decision-making

2. Finally, the following documents Di are retrieved in response to Q:

{D1, D2, D3} INTERSECTION {D2} = {D2}
This means that the original document D is the answer to Q.

Avantajları

✓Formüle edilmesi ve geliştirilmesi kolaydır.

Dezavantajları

✓Birden fazla doküman sonuç olarak döndürülebilir
✓Sonuçların önemine göre sıralanması güçtür
✓Sorgu cümlesinde yer alan boolean ifadenin yorumlanması güç olabilir
✓Bütün terimler eşit ağırlıktadır
✓*information retrieval* dan ziyade *data retrieval* olarak çalışır

Matemetiksel Modeller

Küme-Kuramsal Modeller (Set-theoretic models)

Dokümanlar kelimeler veya ifadelerin kümesi olarak gösterilir.

En çok bilinen modeller:

Standard Boolean Model
Extended Boolean Model
Fuzzy Retrieval

Extended Boolean Model

❖ The goal of the Extended Boolean Model is to overcome the drawbacks of the Boolean model.

❖ The Boolean model doesn't consider term weights in queries.

❖ The result set of a Boolean query is often either too small or too big.

❖ Extended Boolean Model combines the characteristics of the **Vector Space Model** with the properties of Boolean algebra .

❖ Ranks the similarity between queries and documents.

Definition

In the Extended Boolean model, a document is represented as a vector.
Each *i* dimension corresponds to a separate term associated with the document.
The weight of term K_x associated with document d_j is measured by its normalized Term frequency and can be defined as:

$$w_{x,j} = f_{x,j} * \frac{Idf_x}{max_i Idf_x}$$

where Idf_x is inverse document frequency.
The weight vector associated with document d_j can be represented as:

$$\mathbf{v}_{d_j} = [w_{1,j}, w_{2,j}, \ldots, w_{i,j}]$$

The 2 Dimensions Example

Considering the space composed of two terms K_x and K_y only, the corresponding term weights are w_1 and w_2.
Thus, for query $q_{or} = (K_x \vee K_y)$, we can calculate the similarity with the following formula:

$$sim(q_{or}, d) = \sqrt{\frac{w_1^2 + w_2^2}{2}}$$

For query $q_{and} = (K_x \wedge K_y)$, we can use:

$$sim(q_{and}, d) = 1 - \sqrt{\frac{(1 - w_1)^2 + (1 - w_2)^2}{2}}$$

Generalizing the idea and P-norms

We can generalize the 2D extended Boolean model example to higher t-dimensional space using Euclidean distances.
This can be done using P-norms which extends the notion of distance to include p-distances, where $1 \leq p \leq \infty$ is a new parameter.

- A generalized conjunctive query is given by:

$$q_{or} = k_1 \vee^p k_2 \vee^p .... \vee^p k_t$$

- The similarity of q_{or} and d_j can be defined as:

$$sim(q_{or}, d_j) = \sqrt[p]{\frac{w_1^p + w_2^p + ... + w_t^p}{t}}$$

- A generalized disjunctive query is given by:

$$q_{and} = k_1 \wedge^p k_2 \wedge^p .... \wedge^p k_t$$

- The similarity of q_{and} and d_j can be defined as:

$$sim(q_{and}, d_j) = 1 - \sqrt[p]{\frac{(1-w_1)^p + (1-w_2)^p + ... + (1-w_t)^p}{t}}$$

Example

Consider the query q = (K_1 and K_2) or K_3. The similarity between query q and document d can be computed using the formula:

$$sim(q, d) = \sqrt[p]{\frac{(1 - \sqrt[p]{(\frac{(1-w_1)^p + (1-w_2)^p}{2})})^p + w_3^p}{2}}$$

Vector Space Model is an algebraic model for representing text documents as vectors of identifiers, such as, for example, index terms.

Documents and queries are represented as vectors.

$d_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$
$q = (w_{1,q}, w_{2,q}, \ldots, w_{t,q})$

- Each dimension corresponds to a separate term.
- If a term occurs in the document, its value in the vector is non-zero.
- Several different ways of computing these values, one of the best known schemes is tf-idf weighting.

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

where $n_{i,j}$ is the number of occurrences of the considered term (t_i) in document d_j, and the denominator is the sum of number of occurrences of all terms in document d_j, that is, the size of the document $| d_j |$.

$$\mathrm{idf}_i = \log \frac{|D|}{|\{d : t_i \in d\}|}$$

$| D |$: cardinality of D, or the total number of documents in the corpus

$|\{d : t_i \in d\}|$: number of documents where the term t_i appears. If the term is not in the corpus, this will lead to a division-by-zero.

It is therefore common to use 1+ $|\{d : t_i \in d\}|$

Then $(\text{tf-idf})_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_i$

Example

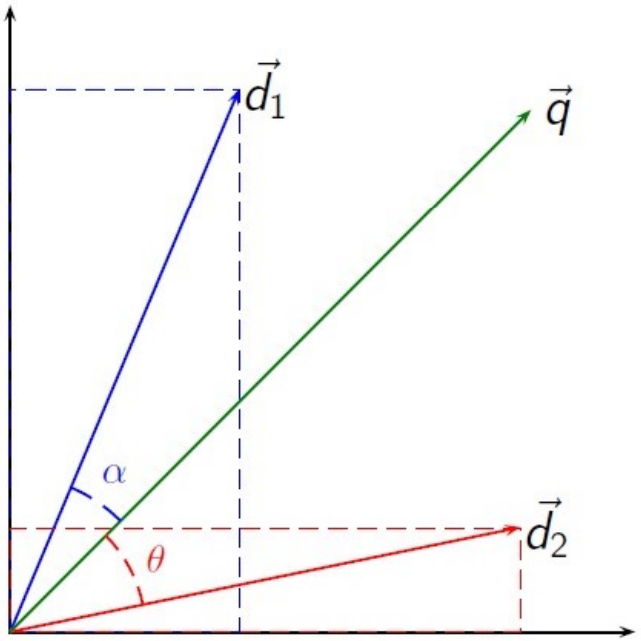
✓Consider a document containing 100 words wherein the word *cow* appears 3 times.
✓The term frequency (TF) for *cow* is then (3 / 100) = 0.03
✓Now, assume we have 10 million documents and *cow* appears in one thousand of these.
✓The inverse document frequency is calculated as
log(10 000 000 / 1 000) = 4

✓TF-IDF score is the product of these quantities:
0.03 × 4 = 0.12

❖Vector operations can be used to compare documents with queries.

❖In practice, it is easier to calculate the cosine of the angle between the vectors instead of the angle:

$$\cos\theta = \frac{\mathbf{d_2} \cdot \mathbf{q}}{\|\mathbf{d_2}\| \, \|\mathbf{q}\|}$$



❖A cosine value of zero means that the query and document vector are orthogonal and have no match.

Cosine Similarity is a measure of similarity between two vectors by measuring the cosine of the angle between them.

- The result of the Cosine function is equal to 1 when the angle is 0, and it is less than 1 when the angle is of any other value.
- Calculating the cosine of the angle between two vectors thus determines whether two vectors are pointing in roughly the same direction.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Vektör Uzayı Modeli

- Varsayım: Kelimeler birbirinden bağımsızdır

Eldekiler:

N doküman
1 sorgu

$$\begin{pmatrix} & T_1 & T_2 & .... & T_t \\ D_1 & d_{11} & d_{12} & ... & d_{1t} \\ D_2 & d_{21} & d_{22} & ... & d_{2t} \\ : & : & : & & : \\ : & : & : & & : \\ D_n & d_{n1} & d_{n2} & ... & d_{nt} \end{pmatrix}$$

$$Q \quad q_1 \quad q_2 \quad \ldots q_t$$

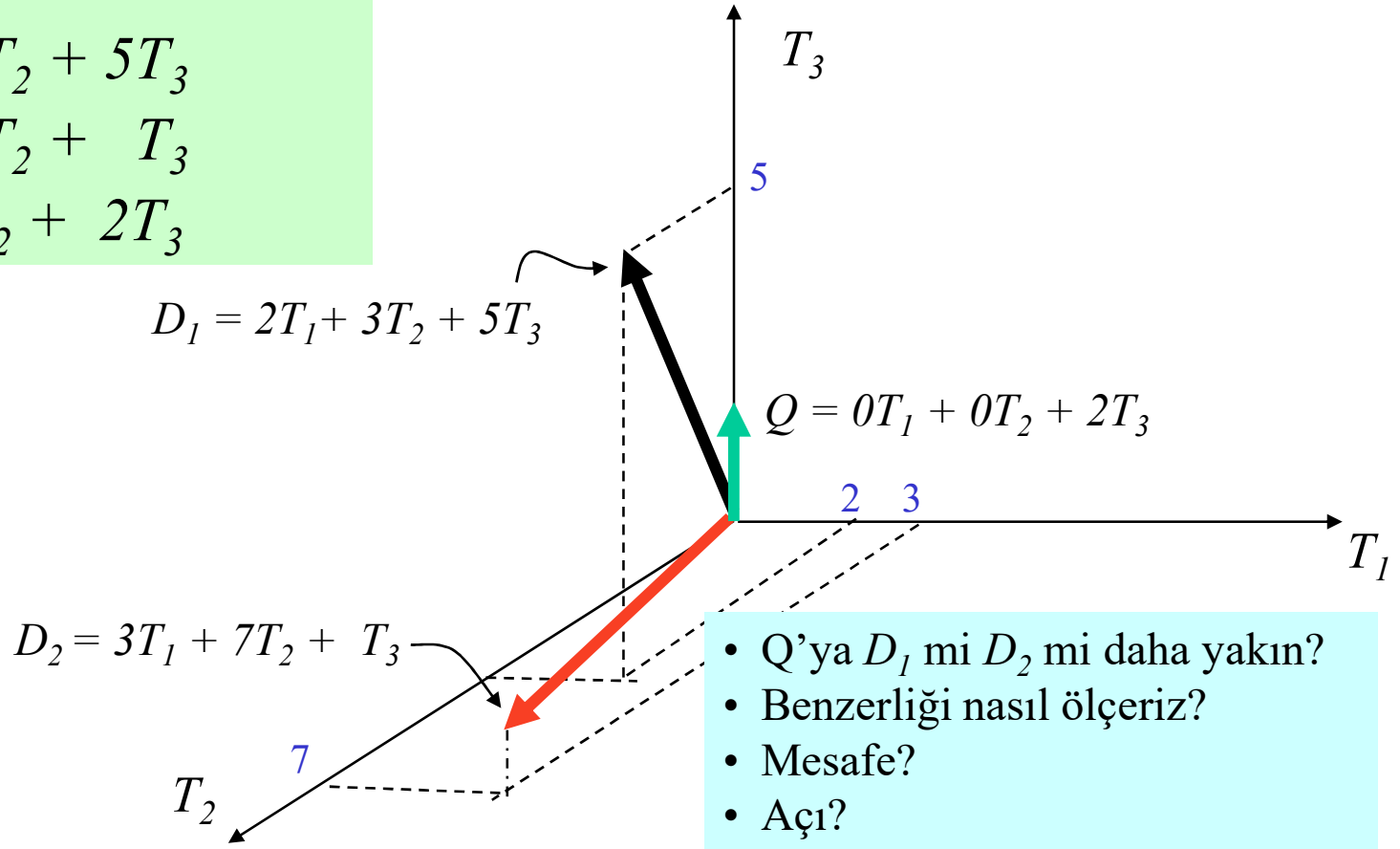
Grafiksel Gösterim

Örnek:

$D_1 = 2T_1 + 3T_2 + 5T_3$

$D_2 = 3T_1 + 7T_2 + T_3$

$Q = 0T_1 + 0T_2 + 2T_3$



- Q'ya D_1 mi D_2 mi daha yakın?
- Benzerliği nasıl ölçeriz?
- Mesafe?
- Açı?
- Projeksiyon?

Benzerlik Ölçümü- Inner Product

$$\text{sim}(D_i, Q) = \sum_{i=1}^{t} (D_i \bullet Q)$$

$$= \sum_{j=1}^{t} d_{ij} * q_j$$

Inner Product - Örnek

Binary:

	retrieval	database	architecture	computer	text	management	information
– D =	1,	1,	1,	0,	1,	1,	0
– Q =	1,	0 ,	1,	0,	0,	1,	1

→ sim(D, Q) = 3

- Vektör boyutu=Sözlük boyutu=7

Ağırlıklı:

$$D_1 = 2T_1 + 3T_2 + 5T_3$$

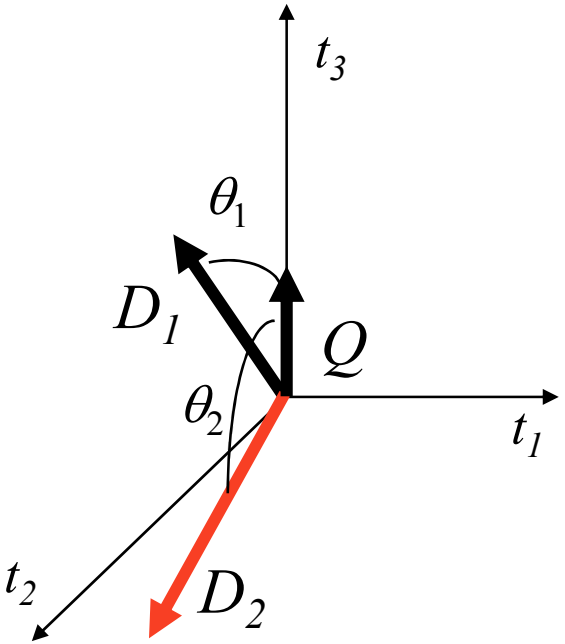
$$Q = 0T_1 + 0T_2 + 2T_3$$

$$\text{sim}(D_1, Q) = 2*0 + 3*0 + 5*2 = 10$$

Cosine Benzerlik Ölçümü

- İki vektör arasındaki açının cosinüsü
- Inner product vektör büyüklükleriyle normalize edilir.

$$\text{CosSim}(D_i, Q) = \frac{\sum_{k=1}^{t} (d_{ik} \bullet q_k)}{\sqrt{\sum_{k=1}^{t} d_{ik}^2} \bullet \sqrt{\sum_{k=1}^{t} q_k^2}}$$



Cosine Benzerlik: Örnek

$D_1 = 2T_1 + 3T_2 + 5T_3$ $\quad \text{CosSim}(D_1, Q) = 0.81$

$D_2 = 3T_1 + 7T_2 + T_3$ $\quad \text{CosSim}(D_2, Q) = 0.13$

$Q = 0T_1 + 0T_2 + 2T_3$

Doküman ve Terim Ağırlıkları

- Ağırlıklar dokümanlardaki frekanslarla (*tf*) ve tüm doküman kütüphanesindeki frekanslarla (*idf*) hesaplanır.

  tf_{ij} = *j*. terimin *i*. dokümandaki frekansı

  df_j = *j*. terimin doküman frekansı

  = *j*. terimi içeren doküman sayısı

  idf_j = *j*. terimin ters doküman frekansı

  = $\log_2 (D/ df_j)$ (D: toplam doküman sayısı)

Terim Ağırlıklarının Bulunması

- *j.* terimin *i.* doküman için ağırlığı:

$$d_{ij} = tf_{ij} \bullet idf_j = tf_{ij} \bullet \log_2 (N/df_j)$$

- TF → Terim Frekansı
  - Bir dokümanda sıkça geçen ancak diğer dokümanlarda pek bulunmayan terimin ağırlığı yüksek olur.
  - $max_l\{tf_{li}\}$ = *i.* dokümanda en çok geçen terimin frekansı
  - Normalizasyon: terim frekansı $= tf_{ij}/max_l\{tf_{li}\}$

$$w = tf/tfmax$$

$$w = IDF = log(D/d)$$

$$w = tf*IDF = tf*log(D/d)$$

$$w = tf*IDF = tf*log((D - d)/d)$$

Inverted file index

Metinler

$T0$ = "it is what it is"
$T1$ = "what is it"
$T2$ = "it is a banana“

inverted file index

- "a": {2}
- "banana": {2}
- "is": {0, 1, 2}
- "it": {0, 1, 2}
- "what": {0, 1}

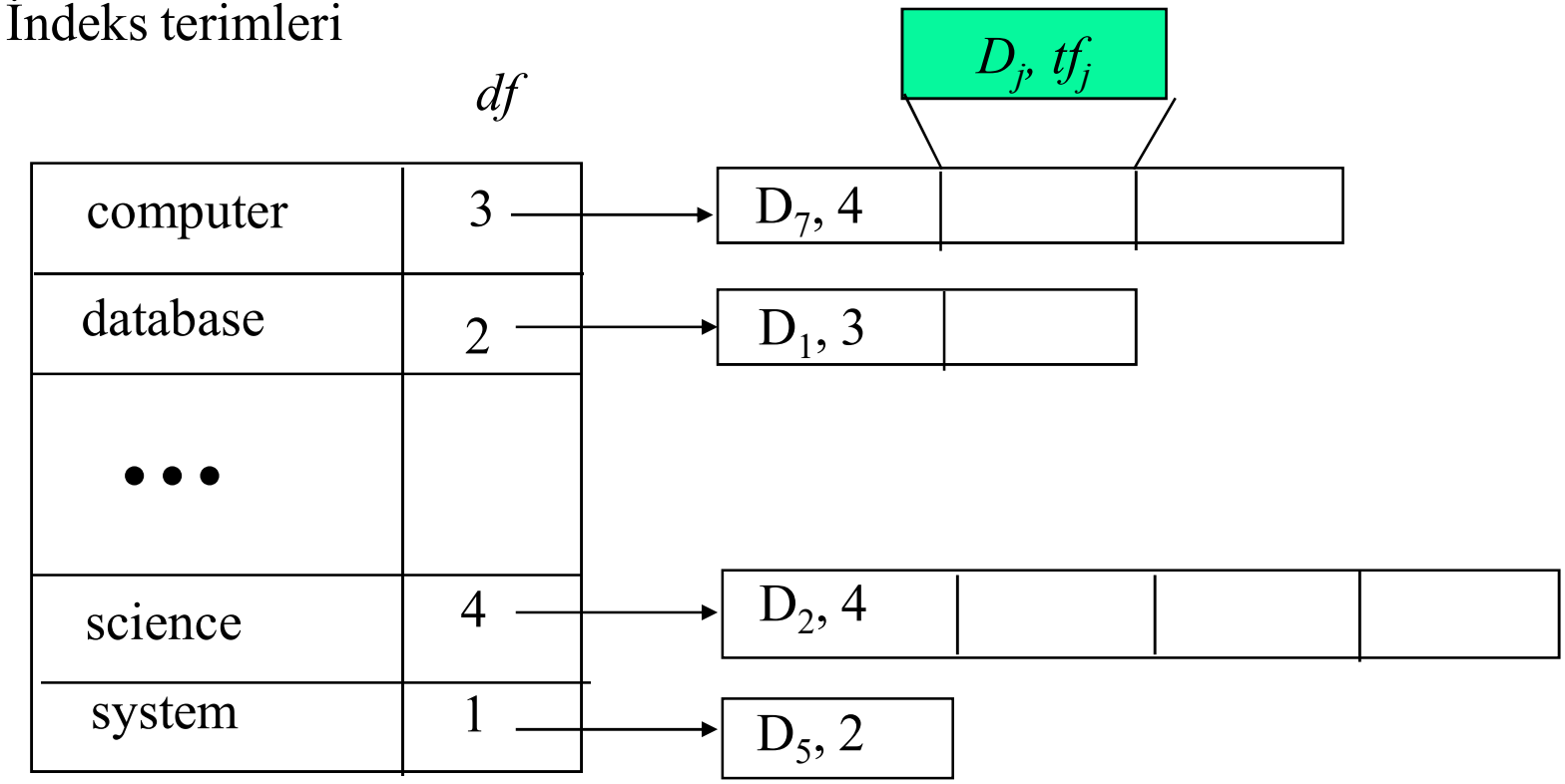
"what", "is" ve "it" kelimeleriyle arama yapılırsa.

$$\{0, 1\} \cap \{0, 1, 2\} \cap \{0, 1, 2\} = \{0, 1\}$$

Full inverted file index: (pozisyonları da içerir)

•"a": {(2, 2)}
•"banana": {(2, 3)}
•"is": {(0, 1), (0, 4), (1, 1), (2, 1)}
•"it": {(0, 0), (0, 3), (1, 2), (2, 0)}
•"what": {(0, 2), (1, 0)}

- Pratikte doküman vektörleri direkt olarak saklanmaz. Hafıza problemlerinden ötürü, arama için aşağıdaki gibi bir yapıda saklanırlar.



Matemetiksel Modeller

Küme-Kuramsal Modeller (Set-theoretic models)

Dokümanlar kelimeler veya ifadelerin kümesi olarak gösterilir.

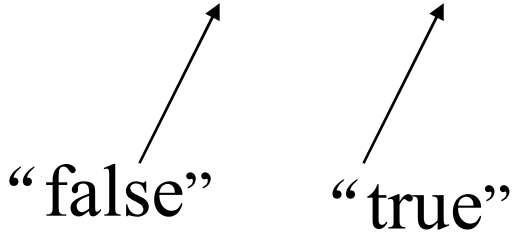
En çok bilinen modeller:

- Standard Boolean Model
- Extended Boolean Model
- **Fuzzy Retrieval**

Natural Language

- Consider:
  - Joe is tall -- what is tall?
  - Joe is very tall -- what does this differ from tall?

- Natural language (like most other activities in life and indeed the universe) is not easily translated into the absolute terms of 0 and 1.



Fuzzy Logic

- An approach to uncertainty that combines real values [0...1] and logic operations

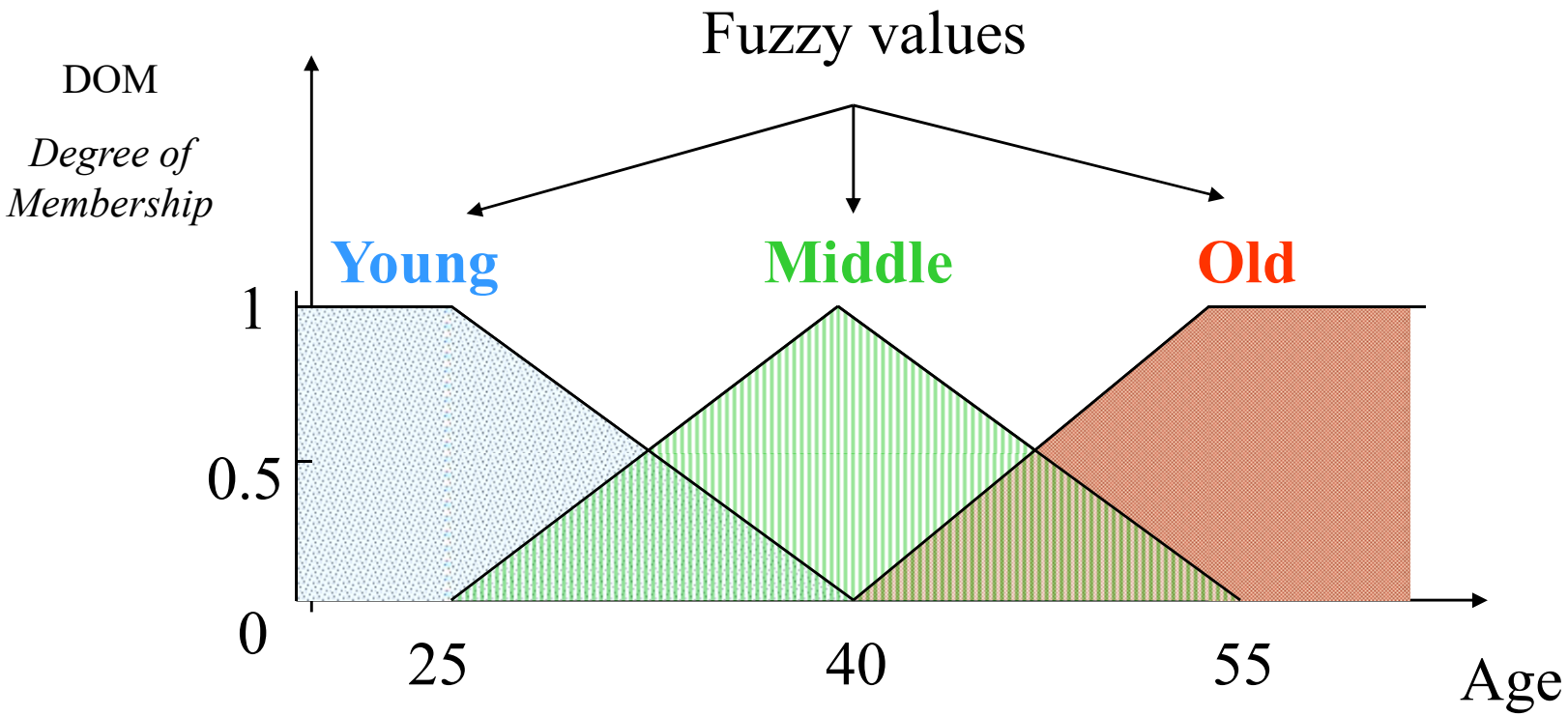
- Fuzzy logic is based on the ideas of fuzzy set theory and fuzzy set membership often found in natural (e.g., spoken) language.

Example: "Young"

- Example:
  - Ann is 28, 0.8 in set "Young"
  - Bob is 35, 0.1 in set "Young"
  - Charlie is 23, 1.0 in set "Young"

- Unlike statistics and probabilities, the *degree* is not describing *probabilities* that the item is in the set, but instead describes *to what extent* the item is the set.

Membership function of fuzzy logic



Fuzzy values have associated degrees of membership in the set.

Fuzzy Set Operations

- Fuzzy OR ($\cup$): the union of two fuzzy sets is the maximum (MAX) of each element from two sets.
- E.g.
  - A = {1.0, 0.20, 0.75}
  - B = {0.2, 0.45, 0.50}
  - $A \cup B$ = {MAX(1.0, 0.2), MAX(0.20, 0.45), MAX(0.75, 0.50)}
    = {1.0, 0.45, 0.75}

Fuzzy Set Operations

- Fuzzy AND ($\cap$): the intersection of two fuzzy sets is just the MIN of each element from the two sets.
- E.g.
  - $A \cap B = \{MIN(1.0, 0.2), MIN(0.20, 0.45), MIN(0.75, 0.50)\} = \{0.2, 0.20, 0.50\}$

Fuzzy Set Operations

- The *complement* of a fuzzy variable with DOM *x* is (1-x).
- Complement: The *complement* of a fuzzy set is composed of all elements′ *complement.*
- Example.
  - $A^c = \{1 - 1.0, 1 - 0.2, 1 - 0.75\} = \{0.0, 0.8, 0.25\}$

Mix Min Max Model

In fuzzy-set theory, an element has a varying degree of membership, say d_A, to a given set A instead of the traditional membership choice (is an element/is not an element).

The degree of membership for union and intersection are defined as follows in Fuzzy set theory:

$$d_{A \cup B} = max(d_A, d_B) \qquad d_{A \cap B} = min(d_A, d_B)$$

to define the similarity of a document to the *or* query to be $max(d_A, d_B)$ and the similarity of the document to the *and* query to be $min(d_A, d_B)$.

The MMM model tries to soften the Boolean operators by considering the query-document similarity to be a linear combination of the *min* and *max* document weights.

Given a document *D* with index-term weights *dA1, dA2, ..., dAn* for terms *A1, A2, ..., An*, and the queries:

Qor = (A1 or A2 or ... or An)
Qand = (A1 and A2 and ... and An)

the query-document similarity in the MMM model is computed as follows:

*SlM(Qor, D) = Cor1 * max(dA1, dA2, ..., dAn) + Cor2 * min(dA1, dA2, ..., dAn)*
*SlM(Qand, D) = Cand1 * min(dA1, dA2, ..., dAn) + Cand2 * max(dA1, dA2 ..., dAn)*

Cor1, Cor2 are "softness" coefficients for the *or* operator,
Cand1, Cand2 are softness coefficients for the *and* operator.

Cor1 > Cor2 and Cand1 > Cand2
Cor1 = 1 - Cor2 and *Cand1 = 1 - Cand2*

Experiments indicate that the best performance usually occurs with *Cand1* in the range [0.5, 0.8] and with *Cor1* > 0.2

The computational cost of MMM is low, and retrieval effectiveness is much better than with the Standard Boolean model.

Matemetiksel Modeller
(Algebraic model)

Algebraic models represent documents and queries usually as vectors, matrices, or tuples. The similarity of the query vector and document vector is represented as a scalar value.

Latent Semantic Indexing

Gizli Anlam Indeksleme (Latent Semantic Indexing) nedir?

- LSI, doğal dil işlemede dokümanlar ve dokümanların içerdiği terimler arasındaki anlamsal ilişkilerin analizinde kullanılan bir tekniktir.

- Klasik yöntemler, dokümanların aranan terimi içerip içermediğine bakarak sınıflandırır ve bir dokümanın başka bir dokümanla ilişkisini göz önünde bulundurmaz.

- İki doküman, ortak kelimeleri olmasa bile semantik olarak birbirine benzer olabilir.

- LSI doküman setlerini bir bütün olarak değerlendirir ve aranan terimin geçtiği dokümanların yanısıra yakın anlamdaki terimlerin bulunduğu dokümanları da bularak sonuç setini genişletir.

➢LSI matematiksel bir yaklaşım kullanır, kelimelerin anlamlarını çıkarmakla ve kelimeleri analiz etmekle uğraşmaz.

Örnekler:
Associated Press haber veritabanında Saddam Hüseyin için yapılan bir arama sonuç olarak:

• Körfez Savaşı, BM yaptırımı, benzin ambargosu makalelerini ve ayrıca
• Irak hakkında Saddam Hüseyin isminin geçmediği makaleleri de döndürmüştür.

• Aynı veritabanında Tiger Woods için yapılan bir arama sonucu, Ünlü golfçünün pekçok hikayesinin anlatıldığı makalelerin yanısıra Tiger Woods yer almadığı ancak, büyük golf turnuvaları hakkındaki makalelerde döndürülmüştür.

- LSI, doğal dillerde çokça geçen ve semantik olarak bir anlamı olmayan kelimeleri eler.

- LSI, sadece semantik olarak bir anlamı olan “content word”ler üzerinde çalışır.

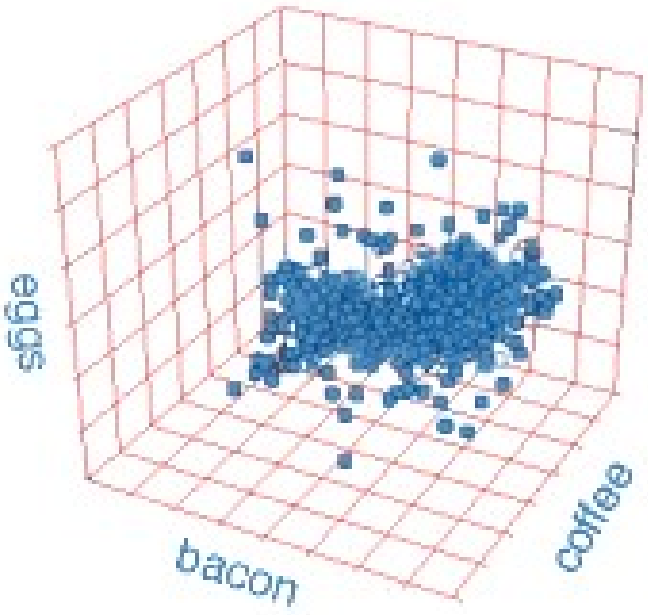
- Content word’ler belirlendikten sonra terim doküman matrisi oluşturulur. Yatay eksende content word ler, dikey eksende de dokümanlar bulunur.

- Her content word için ilgili satıra gidilir ve o content word’ün geçtiği dokumanların bulunduğu sütunlar 1 olarak değerlendirilir. Kelimenin geçmediği sütunlara ise 0 verilir.

- Oluşturulan matrise “Singular Value Decomposition(SVD)” yöntemi uygulanarak matrisin boyutları indirgenir.

Singular Value Decomposition(SVD)

- Term Space



Üç tane keyword'den oluşan bir term space'in grafik olarak gösterilmesi

- Keyword sayısı çok fazla olursa terim uzayının boyutları büyür.

- LSI, SVD yöntemini kullanarak bu çok boyutlu uzayı daha küçük sayıdaki boyutlara bölerek çalışır. Bu şekilde semantik olarak yakın anlamlı olan kelimeler bir araya getirilmiş olur.

LSI için örnek

O'Neill Criticizes Europe on Grants PITTSBURGH (AP)

Treasury Secretary Paul O'Neill expressed irritation Wednesday that European countries have refused to go along with a U.S. proposal to boost the amount of direct grants rich nations offer poor countries.

The Bush administration is pushing a plan to increase the amount of direct grants the World Bank provides the poorest nations to 50 percent of assistance, reducing use of loans to these nations.

Başlıklar, noktalama işaretleri ve büyük harfler kaldırılır.

o'neill criticizes europe on grants treasury secretary paul o'neill expressed irritation wednesday that european countries have refused to go along with a us proposal to boost the amount of direct grants rich nations offer poor countries the bush administration is pushing a plan to increase the amount of direct grants the world bank provides the poorest nations to 50 percent of assistance reducing use of loans to these nations

- Content word'ler ayrılır. Bunun için semantik anlamı olmayan "stop words" kelimeleri çıkarılır.

o'neill criticizes europe grants treasury secretary paul o'neill expressed irritation european countries refused US proposal boost direct grants rich nations poor countries bush administration pushing plan increase amount direct grants world bank poorest nations assistance loans nations

- Çoğul ekleri ve fiil ekleri kaldırılır. İngilizce dili için için Porter Stemmer Algoritması, Türkçe için de Zemberek kullanılabilir.

Content word'ler:

administrat amountassist bank boost bush countri (2) direct europ express grant (2) increas irritat loan nation (3) o'neill paul plan poor (2) propos push refus rich secretar treasuriUS world

- Bu işlem eldeki tüm dokümanlara uygulanır, bir dokümanda ve her dokümanda geçen kelimeleri eleriz ve terim-doküman matrisini elde ederiz.

Term-Document Matris

Document:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	{3000 more columns}
aa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...
amotd	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...
aaliyah	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...
aarp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	...
ab	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...
zywicki	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	...

- Sıfır olmayan her terim-doküman çifti için “terim ağırlığı (term weighting)” değeri bulunur.

1. Bir doküman içinde fazla görünen kelimelerin sadece bir kere görünen kelimelerden daha fazla anlamı vardır.
2. Seyrek kullanılan kelimeler, daha yaygın kullanılan kelimelerden daha ilginç olabilir.

Birincisi her doküman için tek tek uygulanır, buna “yerel ağırlık (local weighting)” denir.

İkincisi bütün dokümanlara birden uygulanır, buna da “global terim ağırlığı (global term weighting)” denir.

- Normalizasyon yapılır.
- Bu üç değer, yani yerel ağırlık, global ağırlık ve normalizasyon faktörü çarpılarak terim-doküman matrisinin sıfır olmayan yerlerinde kullanılacak nümerik değerler bulunur.
- Bundan sonra SVD algoritması çalıştırılır.

	a	b	c	d	e	f	g	h	i	j	k	
aa	-0.0006	-0.0006	0.0002	0.0003	0.0001	0.0000	0.0000	-0.0001	0.0007	0.0001	0.0004	...
amotd	-0.0112	-0.0112	-0.0027	-0.0008	-0.0014	0.0001	-0.0010	0.0004	-0.0010	-0.0015	0.0012	...
aaliyah	-0.0044	-0.0044	-0.0031	-0.0008	-0.0019	0.0027	0.0004	0.0014	-0.0004	-0.0016	0.0012	...
aarp	0.0007	0.0007	0.0004	0.0008	-0.0001	-0.0003	0.0005	0.0004	0.0001	0.0025	0.0000	...
ab	-0.0038	-0.0038	0.0027	0.0024	0.0036	-0.0022	0.0013	-0.0041	0.0010	0.0019	0.0026	...
zywicki	-0.0057	0.0020	0.0039	-0.0078	-0.0018	0.0017	0.0043	-0.0014	0.0050	-0.0020	-0.0011	...

Matris daha az sıfır değeri içerir. Her doküman çoğu content word için benzerlik değeri içerir.

- Bazı değerler negatiftir. Bu Terim Doküman Matrisin'de bir kelimenin bir dokümanda sıfırdan daha az sayıda görünmesi demektir. Bu imkansızdır, aslında doküman ile kelimenin semantik olarak birbirlerine çok uzak olduklarına işaret eder.

- Bu matris bizim dokümanlarımızda arama yapmada kullanacağımız matristir. Bir veya daha fazla terim içeren bir sorguda her terim-doküman kombinasyonu için değerlere bakarız ve her doküman için kümülatif bir skor hesaplarız. Bu dokümanların arama sorgusuna olan benzerliklerini ifade eder.

LSI'nın Kullanım Alanları

- Informal Retrieval
- Synonymy (eş anlamlı)
- Polysemy (yazılışı aynı anlamı farklı)
- Arşivleme
- Otomatik Doküman Sınıflandırma
- Doküman Özetleme
- Metinsel Tutarlık Hesaplama
- Bilgi Filtreleme
- Teknik Raporların Benzerliği
- Yazar Tanıma
- Görüntü dosyalarının otomatik olarak anahtar bir kelime ile işaretlenmesi

Singular Value Decomposition (SVD)

Problem: Compute the full SVD for the following matrix:

$$A = \begin{bmatrix} 4 & 0 \\ 3 & -5 \end{bmatrix}$$

Step 1. Compute its transpose $\mathbf{A^T}$ and $\mathbf{A^TA}$.

Since $A^T = \begin{bmatrix} 4 & 3 \\ 0 & -5 \end{bmatrix}$ then, $A^TA = \begin{bmatrix} 4 & 3 \\ 0 & -5 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 3 & -5 \end{bmatrix}$

$$A^TA = \begin{bmatrix} 25 & -15 \\ -15 & 25 \end{bmatrix}$$

Step 2. Determine the eigenvalues of $\mathbf{A^TA}$ and sort these in descending order, in the absolute sense. Square roots these to obtain the singular values of **A**.

$$A^TA - cI = \begin{bmatrix} 25 - c & -15 \\ -15 & 25 - c \end{bmatrix} \qquad |A^TA - cI| = (25 - c)(25 - c) - (-15)(-15) = 0$$

characteristic equation $\longrightarrow$ $c^2 - 50c + 400 = 0$

The quadratic equation gives two values.
In decreasing order, these are $\longrightarrow$ $|40| > |10|$

eigenvalues $\longrightarrow$ $c_1 = 40 \quad c_2 = 10$

singular values $\longrightarrow$ $s_1 = \sqrt{40} = 6.3245 > s_2 = \sqrt{10} = 3.1622$

Step 3. Construct diagonal matrix S by placing singular values in descending order along its diagonal. Compute its inverse, S^{-1}.

$$S = \begin{bmatrix} 6.3245 & 0 \\ 0 & 3.1622 \end{bmatrix} \qquad S^{-1} = \begin{bmatrix} 0.158 & 0 \\ 0 & 0.3162 \end{bmatrix}$$

Step 4. Use the ordered eigenvalues from step 2 and compute the eigenvectors of A^TA. Place these eigenvectors along the columns of V and compute its transpose, V^T.

for $c_1 = 40$

$$A^TA - cI = \begin{bmatrix} 25 - 40 & -15 \\ -15 & 25 - 40 \end{bmatrix} = \begin{bmatrix} -15 & -15 \\ -15 & -15 \end{bmatrix}$$

$$(A^TA - cI)\, X_1 = 0$$

$$\begin{bmatrix} -15 & -15 \\ -15 & -15 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$-15 x_1 + -15 x_2 = 0$$
$$-15 x_1 + -15 x_2 = 0$$

Solving for x_2 for either equation: $x_2 = -x_1$

$$X_1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ -x_1 \end{bmatrix}$$

Dividing by its length,

$$L = \sqrt{x_1^2 + x_2^2} = x_1\sqrt{2}$$

$$X_1 = \begin{bmatrix} x_1 / L \\ -x_1 / L \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 0.7071 \\ -0.7071 \end{bmatrix}$$

for $c_2 = 10$

$$A^TA - cI = \begin{bmatrix} 25 - 10 & -15 \\ -15 & 25 - 10 \end{bmatrix} = \begin{bmatrix} 15 & -15 \\ -15 & 15 \end{bmatrix}$$

$$(A^TA - cI)\, X_2 = 0$$

$$\begin{bmatrix} 15 & -15 \\ -15 & 15 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$15 x_1 + -15 x_2 = 0$$
$$-15 x_1 + 15 x_2 = 0$$

Solving for x_2 for either equation: $x_2 = x_1$

$$X_2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_1 \end{bmatrix}$$

Dividing by its length,

$$L = \sqrt{x_1^2 + x_2^2} = x_1\sqrt{2}$$

$$X_2 = \begin{bmatrix} x_1 / L \\ x_1 / L \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix}$$

$$V = \begin{bmatrix} x_1 & x_2 \end{bmatrix} = \begin{bmatrix} 0.7071 & 0.7071 \\ -0.7071 & 0.7071 \end{bmatrix}$$

$$V^T = \begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \end{bmatrix}$$

Step 5. Compute U as $U = AVS^{-1}$. To complete the proof, compute the full SVD using $A = USV^T$.

$$U = AVS^{-1} = \begin{bmatrix} 4 & 0 \\ 3 & -5 \end{bmatrix} \begin{bmatrix} 0.7071 & 0.7071 \\ -0.7071 & 0.7071 \end{bmatrix} \begin{bmatrix} 0.1581 & 0 \\ 0 & 0.3162 \end{bmatrix}$$

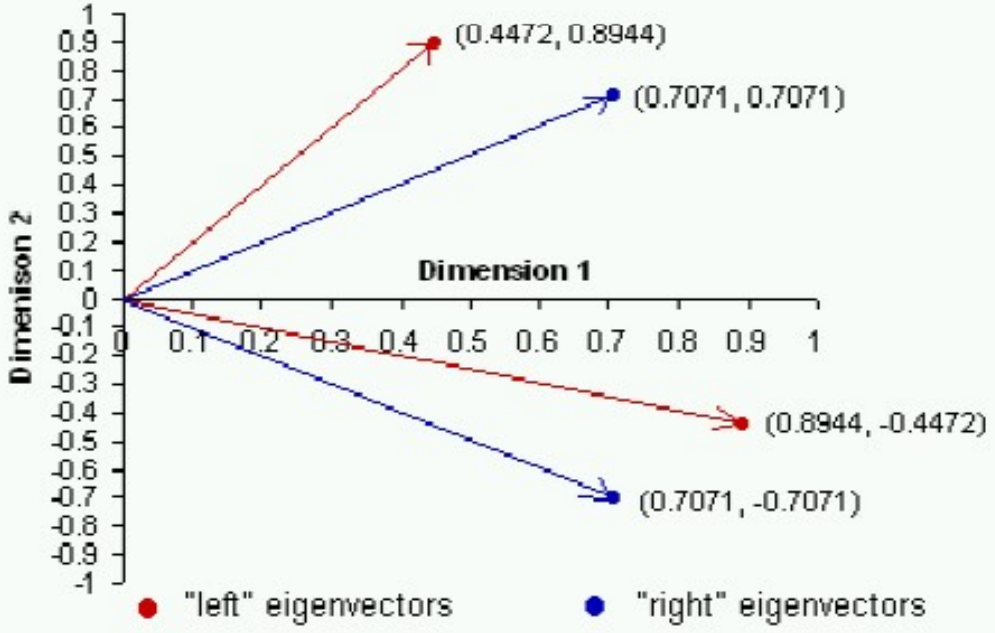
$$U = AVS^{-1} = \begin{bmatrix} 4 & 0 \\ 3 & -5 \end{bmatrix} \begin{bmatrix} 0.1118 & 0.2236 \\ -0.1118 & 0.2236 \end{bmatrix}$$

$$U = AVS^{-1} = \begin{bmatrix} 0.4472 & 0.8944 \\ 0.8944 & -0.4472 \end{bmatrix}$$

$$A = USV^T = \begin{bmatrix} 0.4472 & 0.8944 \\ 0.8944 & -0.4472 \end{bmatrix} \begin{bmatrix} 6.3245 & 0 \\ 0 & 3.1622 \end{bmatrix} \begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \end{bmatrix}$$

$$A = USV^T = \begin{bmatrix} 0.4472 & 0.8944 \\ 0.8944 & -0.4472 \end{bmatrix} \begin{bmatrix} 4.4721 & -4.4721 \\ 2.2360 & 2.2360 \end{bmatrix}$$

$$A = USV^T = \begin{bmatrix} 3.9998 & 0 \\ 2.9999 & -4.9997 \end{bmatrix} \approx \begin{bmatrix} 4 & 0 \\ 3 & -5 \end{bmatrix}$$



m x n Term-Document Matrix, A

$w_{ij} = L_{ij} G_i N_j$ current models

$w_{ij} = L_{ij} = tf_{ij}$ old model

m = rows = terms

n = columns = documents

$a_{ij} = w_{ij}$ **= term weights**

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Term Count Model

The weight of term i in document j is defined as a local weight (*Lij*):

Equation 1: *wij = Lij = tfij*

where *tfij* is term frequency or number of times term i occurs in document j.

Equation 2: *wij = LijGiNj*

where *Li, j* is the local weight for term *i* in document *j*.

Gi is the global weight for term *i* across all documents in the collection.

Nj is the normalization factor for document *j*.

Latent Semantic Indexing (LSI)

A “collection” consists of the following “documents”

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.

The authors used the Term Count Model to score term weights and query weights, so local weights aredefined as word occurences. The following document indexing rules were also used:

· stop words were not ignored
· text was tokenized and lowercased
· no stemming was used
· terms were sorted alphabetically

Problem: Use Latent Semantic Indexing (LSI) to rank these documents for the query *gold silver truck*.

Step 1: Score term weights and construct the term-document matrix **A** and query matrix:

Terms	d1	d2	d3	q
a	1	1	1	0
arrived	0	1	1	0
damaged	1	0	0	0
delivery	0	1	0	0
fire	1	0	0	0
gold	1	0	1	1
in	1	1	1	0
of	1	1	1	0
shipment	1	0	1	0
silver	0	2	0	1
truck	0	1	1	1

$A =$ (the matrix above) $\quad q =$ (the q column above)

Step 2: Decompose matrix **A** matrix and find the **U**, **S** and **V** matrices, where

$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathbf{T}}$

$$U = \begin{bmatrix} -0.4201 & 0.0748 & -0.0460 \\ -0.2995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & -0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{bmatrix} \quad S = \begin{bmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.4945 & 0.6492 & -0.5780 \\ -0.6458 & -0.7194 & -0.2556 \\ -0.5817 & 0.2469 & 0.7750 \end{bmatrix} \quad V^{T} = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \\ -0.5780 & -0.2556 & 0.7750 \end{bmatrix}$$

Step 3: Implement a Rank 2 Approximation by keeping the first columns of **U** and **V** and the first columns and rows of **S**.

$$U \approx U_k = \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \qquad k = 2$$

$$S \approx S_k = \begin{bmatrix} 4.0989 & 0.0000 \\ 0.0000 & 2.3616 \end{bmatrix}$$

$$V \approx V_k = \begin{bmatrix} -0.4945 & 0.6492 \\ -0.6458 & -0.7194 \\ -0.5817 & 0.2469 \end{bmatrix} \qquad V^T \approx V_k^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \end{bmatrix}$$

Step 4: Find the new document vector coordinates in this reduced 2-dimensional space.
Rows of **V** holds eigenvector values. These are the coordinates of individual document vectors, hence

d1(-0.4945, 0.6492)
d2(-0.6458, -0.7194)
d3(-0.5817, 0.2469)

Step 5: Find the new query vector coordinates in the reduced 2-dimensional space.

$$\mathbf{q} = \mathbf{q}^{T}\mathbf{U}_{k}\mathbf{S}_{k}^{-1}$$

$$q = q^T U_k S_k^{-1} \qquad k = 2$$

$$q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \begin{bmatrix} \frac{1}{4.0989} & 0.0000 \\ 0.0000 & \frac{1}{2.3616} \end{bmatrix}$$

$$q = \begin{bmatrix} -0.2140 & -0.1821 \end{bmatrix}$$

Step 6: Rank documents in decreasing order of query-document cosine similarities.

$$\text{sim}(q, d) = \frac{q \bullet d}{|q||d|}$$

$$\text{sim}(q, d_1) = \frac{(-0.2140)(-0.4945) + (-0.1821)(0.6492)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.4945)^2 + (0.6492)^2}} = -0.0541$$

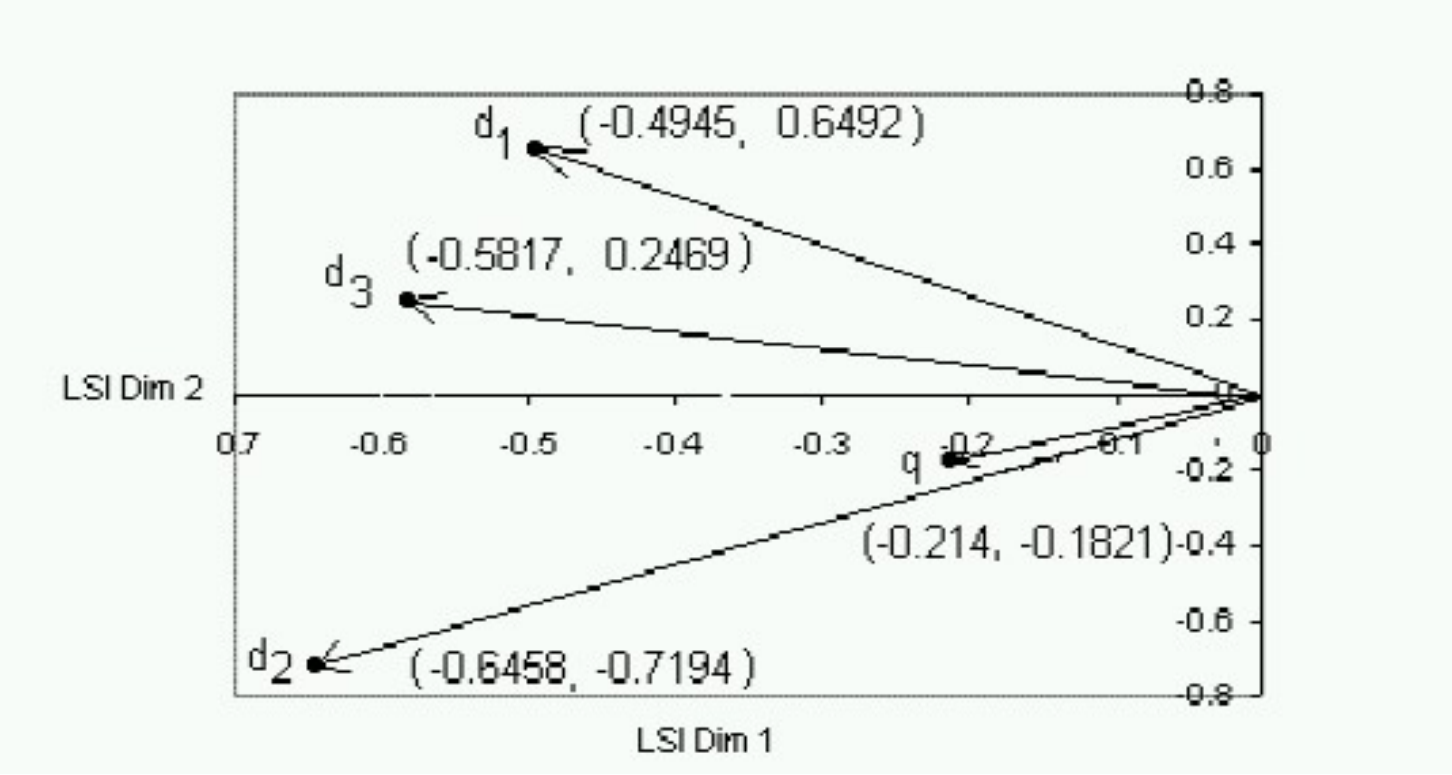
$$\text{sim}(q, d_2) = \frac{(-0.2140)(-0.6458) + (-0.1821)(-0.7194)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.6458)^2 + (-0.7194)^2}} = 0.9910$$

$$\text{sim}(q, d_3) = \frac{(-0.2140)(-0.5817) + (-0.1821)(0.2469)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.5817)^2 + (0.2469)^2}} = 0.4478$$

Ranking documents in descending order

$$d_2 > d_3 > d_1$$

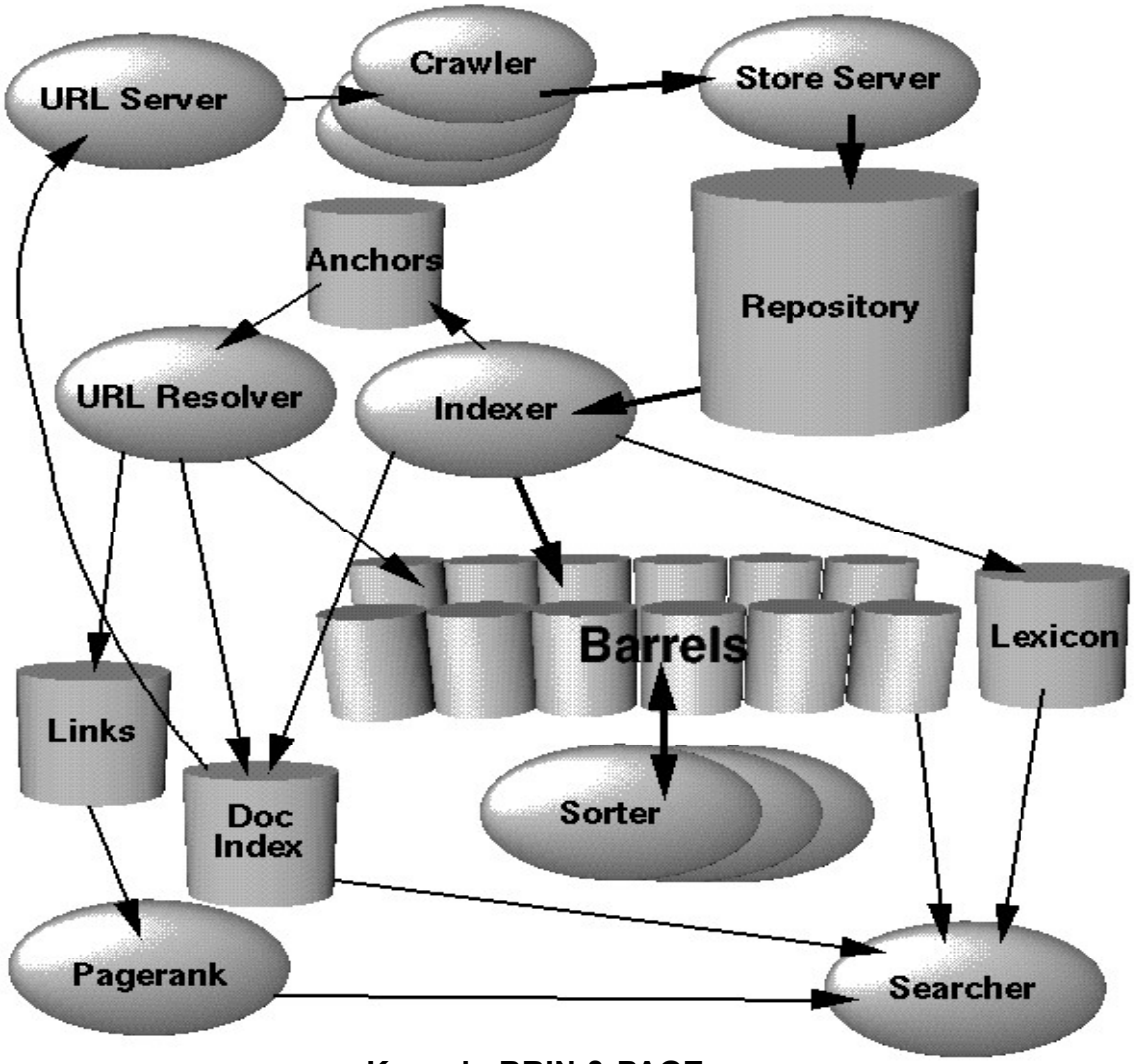
We can see that document d2 scores higher than d3 and d1. Its vector is closer to the query vector than the other vectors. Also note that Term Vector Theory is still used at the beginning and at the end of LSI.



Web Katalogları vs. Arama Motorları

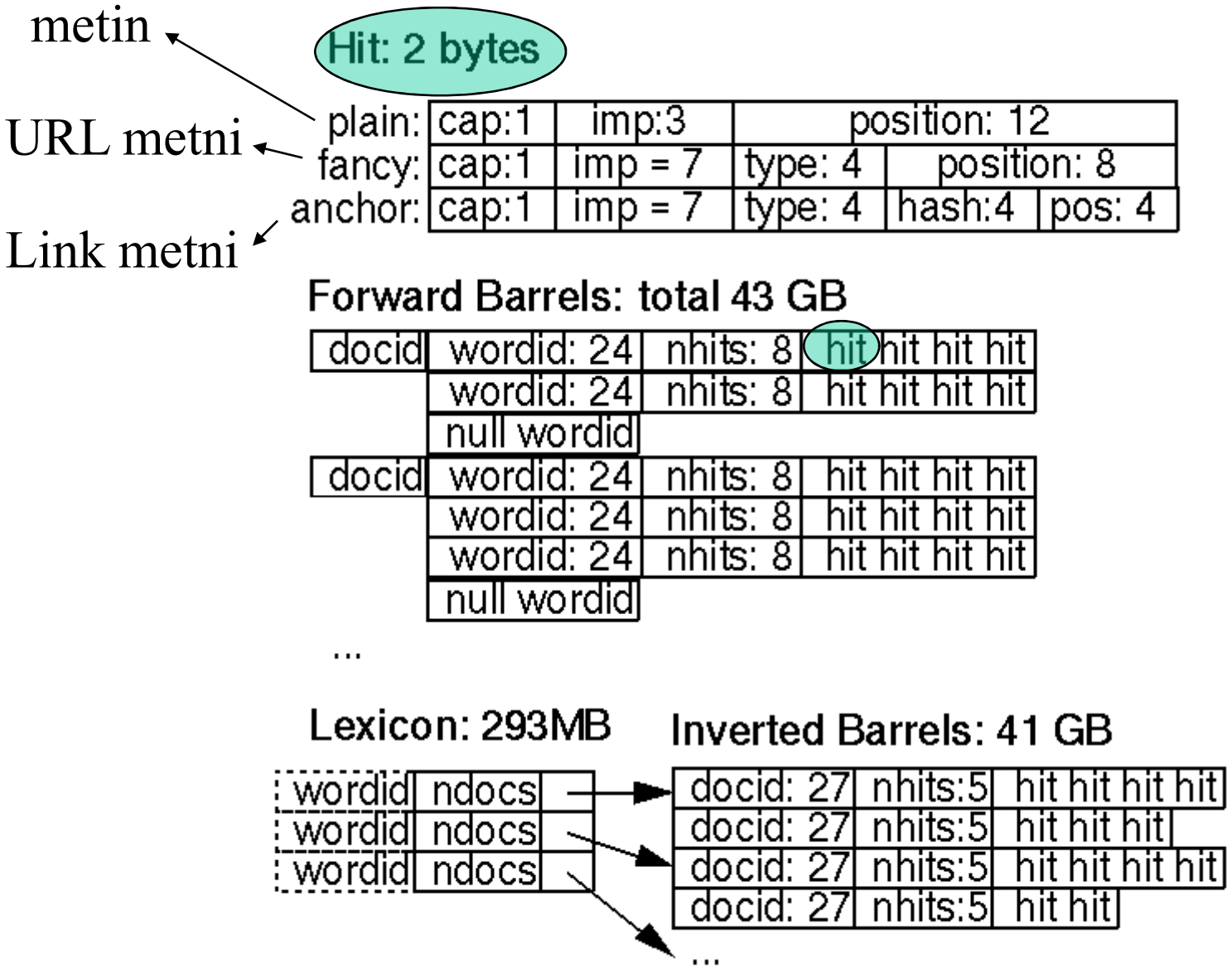
- Web Katalogları
  - Elle seçilmiş siteler
  - Sayfaların içeriğinde değil, tanımlarında arama
  - Hiyerarşik kategorilere atanırlar
- Arama Motorları
  - Tüm sitelerdeki tüm sayfalar
  - Sayfaların içeriğinde arama
  - Sorgu geldikten sonra bulunan skorlara göre sıralanırlar.

Google'ın Mimarisi



Kaynak: BRIN & PAGE

- URL Server- kaydedilecek URL'ler
- Crawler paralel
- Store Server – crawler'lardan gelen sayfaları sıkıştırıp repository'e kaydeder.
- Repository – sayfaların HTML kodları
- Indexer – forward barrel'leri ve anchors'u oluşturur.
- Lexicon – tekil kelime listesi, yer aldığı docID'ler
- Barrels (docID, (wordID, hitList*)*)* ler
- Anchors – sayfalarda bulunan link bilgileri (from, to, anchor text)
- URL Resolver – Anchors'daki rölatif URL'leri gerçek URL'lere dönüştürür. DocID'lerini verir. Links'i oluşturur. Anchor metinlerini forward barrel'lara ekler
- Sorter – inverted barrel'leri oluşturur.
- Doc Index – her sayfa hakındaki bilgiler (durum, repository'deki pointer vs.)
- Links – docID ikilileri
- Pagerank – her sonuç sayfasının önemini, popülerliğini (links'ten) hesaplar
- Searcher – sorguları cevaplar


metin
URL metni
Link metni
Hit: 2 bytes
plain: cap:1 imp:3 position: 12
fancy: cap:1 imp = 7 type: 4 position: 8
anchor: cap:1 imp = 7 type: 4 hash:4 pos: 4
Forward Barrels: total 43 GB
docid wordid: 24 nhits: 8 hit hit hit hit
wordid: 24 nhits: 8 hit hit hit hit
null wordid
docid wordid: 24 nhits: 8 hit hit hit hit
wordid: 24 nhits: 8 hit hit hit hit
wordid: 24 nhits: 8 hit hit hit hit
null wordid
...
Lexicon: 293MB
Inverted Barrels: 41 GB
wordid ndocs
wordid ndocs
wordid ndocs
docid: 27 nhits:5 hit hit hit hit
docid: 27 nhits:5 hit hit hit
docid: 27 nhits:5 hit hit hit hit
docid: 27 nhits:5 hit hit
...

Puanlama Sistemi

- Kriterler
  - Pozisyon, Font Büyüklüğü, Büyük Harfle/ Bold/Italik yazılma
  - Sitenin popülerliği (PageRank)
  - Başlık, link metni (Anchor Text), URL metni vs.

Sistem Performansı

- 26 milyon site 9 günde indirilmiş.(Saniyede 48.5 sayfa)
- Indexer ve Crawler aynı anda çalışıyor
- Indexer saniyede 54 sayfayı indeksliyor
- Sorter'lar 4 makinede paralel çalışarak 24 saatte inverted index'i oluşturuyor

Anahtar Kelime Problemleri

- Eşanlamlı kelimeleri içeren dokümanlar bulunamaz.
  - "restaurant" vs. "cafe"
  - "PRC" vs. "China"
- Eşsesli kelimeler ilgisiz dokümanların bulunmasına sebep olabilir.
  - "bat" (baseball, mammal)
  - "Apple" (company, fruit)
  - "bit" (unit of data, act of eating)

Zeki IR Teknikleri

- Kelimelerin anlamları
- Sorgudaki kelimelerin sırası
- Kullanıcılardan döndürülen sonuçların kalitesiyle ilgili alınan geri bildirimler
- Aramayı ilgili kelimelerle genişletmek
- İmla denetimi
- Kaynakların güvenilirliği

Sensitivity ve Specificity

- İstatistiksel olarak performans ölçüm metrikleridir (Binary Classification).
- **Sensitivity** , bazı alanlarda **Recall** olarak adlandırılır (IR)

Tanım

Bir hastalığın teşhisi için insanlara bir test yapılacağını düşünün.
Test sonucunda kişiler hasta veya değil diye etiketlenecek.
Test sonucu pozitif ise kişi hasta, negatif ise hasta değil demektir.

True positive: Sick people correctly diagnosed as sick
False positive: Healthy people incorrectly identified as sick
True negative: Healthy people correctly identified as healthy
False negative: Sick people incorrectly identified as healthy.

Detection dog

Köpek 1 : Sadece cocaine bulmak için eğitilmiş.

Köpek 2: Cocaine, heroin ve arijuana gibi farklı kokuları almak için eğitilmiş.

Specificity : Birinci köpek, cocaine kaçırmaz ve hatalı tanıma yapma olasılığı azdır
(so it is very *specific).*

Sensitivity : İkinci köpek daha fazla sayıda maddeyi tanır. Fakat cocaine karşı daha az duyarlıdır. Hata yapma olasılığı daha fazladır.

$$\text{specificity} = \frac{\text{number of True Negatives}}{\text{number of True Negatives} + \text{number of False Positives}}$$

$$\text{sensitivity} = \frac{\text{number of True Positives}}{\text{number of True Positives} + \text{number of False Negatives}}$$

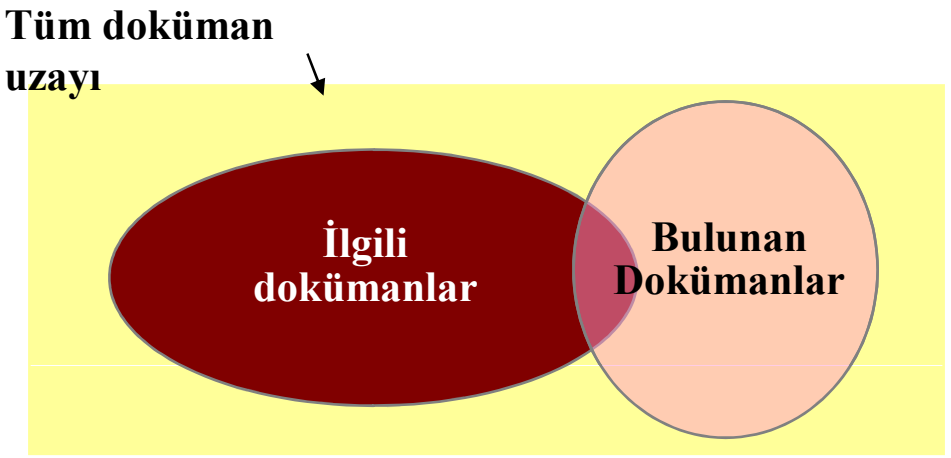
		Condition (as determined by "Gold standard")		
		Positive	***Negative***	
Test outcome	***Positive***	**True Positive**	**False Positive** (Type I error, P-value)	Positive predictive value
	Negative	**False Negative (Type II error)**	**True Negative**	Negative predictive value
		↓ **Sensitivity**	↓ **Specificity**	

Sensitivity=True Positive Rate=TP/(TP+FN)
Specificity=True Negative Rate=TN/(TN+FP)

False Negative Rate(β)=1-Sensitivity=FN/(TP+FN)
False Positive Rate(α)=1-Specificity=FP/(FP+TN)

Positive Predictive Rate Value=TP/(TP+FP)
Negative Predictive Rate Value=TN/(TN+FN)

Değerlendirme



relevant / irrelevant	retrieved	not retrieved
	retrieved & irrelevant	Not retrieved & irrelevant
	retrieved & relevant	not retrieved but relevant

F-measure , Precision ve Recall ‘ın harmonik ortalamasıdır.

$$F = 2 \times \frac{\mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

Information Retrevial kullanımı

$$\mathrm{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\mathrm{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

Kaynaklar

- http://www.cs.huji.ac.il/~sdbi/2000/google/index.htm
- Searching the Web ,Ray Larson & Warren Sack
- Knowledge Management with Documents, Qiang Yang
- Introduction to Information Retrieval, Rada Mihalcea
- Introduction to Information Retrieval, Evren Ermis
- The Anatomy of a Large-Scale Hypertextual Web Search Engine, Sergey Brin, Lawrence Page (http://infolab.stanford.edu/~backrub/google.html)
- http://en.wikipedia.org/wiki/Fuzzy_retrieval

Son