

DAR practicumopgave 1

ranking in IR/DB

Ranking op database queries

Motivating example: "X-files query"

```
SELECT * FROM Xfile
WHERE skin_color = 'grey'
AND skin_hair = 0
AND pupil_dilatation = 'oval'
AND verbal_sound = 'gargling'
AND length BETWEEN 3 AND 4
AND number_of_toes = 8
AND ESP_capability = 'high'
AND character = 'charming'
```

Ranking op database queries

Twee mogelijke problemen

1. many answers

er zijn veel tuples in de resultaattabel;
kunnen we ranking-technieken uit de
IR gebruiken voor een top-k selectie?

2. zero answers

er zijn geen (of te weinig) tuples die aan alle eisen
voldoen, maar zouden we mbv. ranking
een “zo goed mogelijk” antwoord kunnen
geven door verzwakking van de eisen?

Ranking op database queries

Aanpak op basis van:

"Automated Ranking of Database Query Results"

Agrawal, Chaudhuri, Das, Gionis

Microsoft Research/Stanford

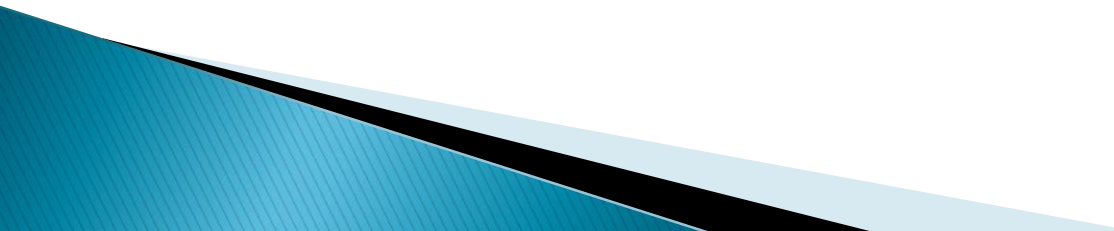
Ranking op database queries

IR:

*Welke documenten passen het beste
bij mijn query*

DB:

*Welke tupels passen het beste
bij mijn query*



Ranking op database queries

Kernvraag:

Hoe formaliseer je “passen”
in de context van relationele queries?

Hoe generaliseer je de IR-maten en scores
naar het relationele paradigma?

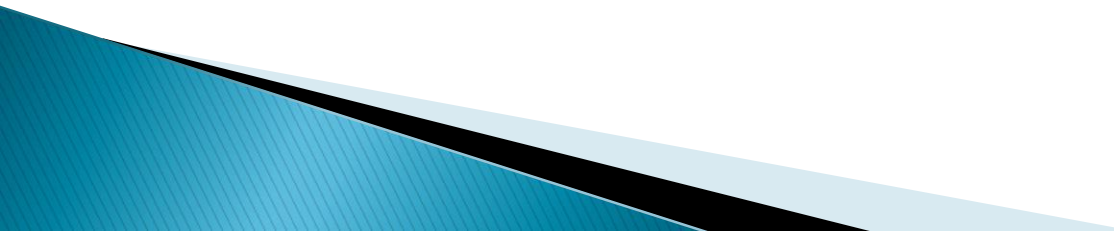
Definities:

We gaan vooralsnog uit van één tabel R
met attributen A_1, \dots, A_m
en tupels t_1, \dots, t_N

Generaliseer klassieke IR

(Her)definitions

Wat is in deze database-context de tegenhanger van het begrip:

- Document
 - Term
 - Term frequency
 - Idf
 - Cosine similarity score (tussen query en tuple)
- 

Ranking op database queries

Conjunctive equality query

```
SELECT * FROM R WHERE  
C1 AND ... AND Ck
```

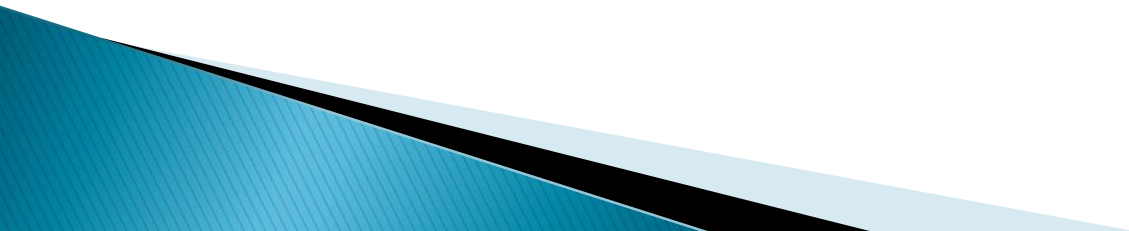
Hierin is conditie c_i van de vorm
 $A_j = q_j$

Een meer geavanceerde c_i is
 $A_j \text{ IN } (q_{j1}, \dots, q_{j1})$

Generaliseer klassieke IR

(Her)definitions

Hoe definiëren we TF?



Generaliseer klassieke IR

(Her)definities

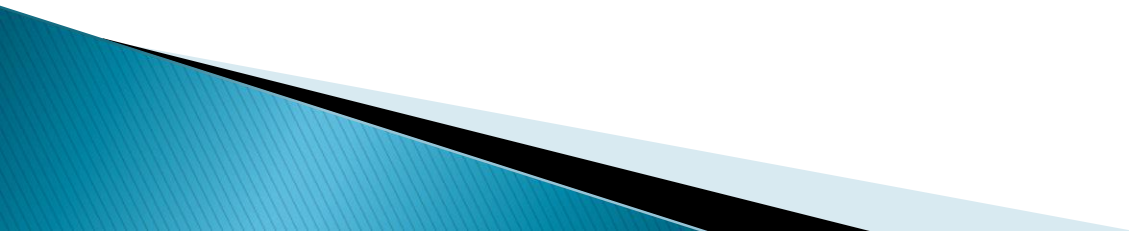
Een document komt overeen met een tupel

Direct gevolg is dat in ons model $TF = 1$
als er een match is op attribuut/waarde,
anders 0

Generaliseer klassieke IR

(Her)definitions

Hoe definiëren we IDF ?



Generaliseer klassieke IR

(Her)definities

Gegeven attribuut A_k

Voor elke waarde v uit $\text{dom}(A_k)$

is $F_k(v)$ het aantal tuples in R met $A_k = v$

Laat $N = |R|$;

we definiëren $\text{idf}_k(v) = \log(N/F_k(v))$

Generaliseer klassieke IR

Tegenhanger TF-IDF:

We definiëren de similarity coefficient

$$S(u,v) = \text{idf}_k(u) \text{ als } u = v, \text{ anders } 0$$

u = waarde voor A_k in query

v = waarde voor A_k in tuple

Generaliseer klassieke IR

(Her)definities

Gegeven een tuple $T = \langle t_1, \dots, t_m \rangle$
en een query $Q = \langle q_1, \dots, q_m \rangle$

de similarity tussen T en Q is:

$$\text{sim}(T, Q) = \sum_{i=1}^m S(t_i, q_i)$$

Generaliseer klassieke IR

Complicaties

1. attribuut is niet categorisch maar numeriek
2. conditie heeft vorm
 $A_j \text{ IN } (q_{j1}, \dots, q_{j1})$

IDF-similarity ?

Observatie 1

In een onroerend goed database zijn nieuwe woningen vaker vertegenwoordigd dan oude. Hun IDF is dus lager. Toch is de vraag naar nieuwe woningen groter.

Observatie 2

In een boekwinkel is de vraag naar een bepaalde auteur niet afhankelijk van het aantal boeken dat hij/zij geschreven heeft. De IDF geeft hier ook geen goede indicatie. (Vestdijk)

IDF similarity of ... ?

Oplossing 1

Laat een domeinexpert de weging bepalen:
duur, applicatie-afhankelijk

Oplossing 2

Haal je weging uit de query workload (automatisch);
dit is de verzameling queries die gedurende de
levensduur van je database verzameld zijn

QF-similarity

QF similarity: gebruik de workload

Definities

$RQF_k(v)$ is de “raw query frequency” van waarde (term) v onder attribuut A_k in de workload:

Hoeveel queries zijn er uitgevoerd met

```
SELECT ... FROM ...  
WHERE  $A_k = v$  AND ...
```

(Bijdrage queries wegen met aantal calls)

QF-similarity

QF similarity: gebruik de workload

Definities

$RQFMAX_k$ is de frequentie van de meest voorkomende term

$$QF_k(v) = RQF_k(v) / RQFMAX_k$$

$QF_k(v)$ is een maat voor de populariteit van de zoekterm v

QF-similarity

We kunnen QF-similarity als volgt definiëren:

$S(u, v) = QF(u)$, als $u = v$, anders 0

En voor tuple T en query Q :

$$\textit{sim}(T, Q) = \sum_{i=1}^m S(t_i, q_i)$$

Attribute value similarity

Observatie

Query op attributen merk, type:
<Ferrari, 430>

tuples:

<Lamborghini, Gallardo>

<Kia, Picanto>

in beide gevallen is de $S(q_i, t_i) = 0$
maar ...

Hoe pakken we dit aan?

Attribute value similarity

$W(v)$ is de subset van de queries in de workload waarin waarde v voorkomt in een IN-clause voor een specifiek attribuut:

$v \text{ IN } (\text{Toyota}, \text{Nissan}, \text{Honda})$

De Jacquard coefficient meet de similarity tussen sets $W(t)$ en $W(q)$:

$$J(W(t), W(q)) = \frac{|W(t) \cap W(q)|}{|W(t) \cup W(q)|}$$

Attribute value similarity: voorbeeld

$Q1: \{ Opel, VW, Ford \}$

$Q2: \{ VW, Ford, Renault \}$

$Q3: \{ Fiat, Opel, Peugeot, Citroën \}$

$Q4: \{ VW, Opel, Citroën \}$

$W(Opel) = \dots ?$

$W(VW) = \dots ?$

$W(Ford) = \dots ?$

$W(Citroën) = \dots ?$

Attribute value similarity: voorbeeld

$Q1: \{ Opel, VW, Ford \}$

$Q2: \{ VW, Ford, Renault \}$

$Q3: \{ Fiat, Opel, Peugeot, Citroën \}$

$Q4: \{ VW, Opel, Citroën \}$

$W(Opel) = \{ Q1, Q3, Q4 \}$

$W(VW) = \{ Q1, Q2, Q4 \}$

$W(Ford) = \{ Q1, Q2 \}$

$W(Citroën) = \{ Q3, Q4 \}$

$J(W(Opel), W(VW)) = \dots ?$

$J(W(Ford), W(VW)) = \dots ?$

$J(W(Citroën), W(Ford)) = \dots ?$

Attribute similarity: voorbeeld

$Q1: \{ Opel, VW, Ford \}$

$Q2: \{ VW, Ford, Renault \}$

$Q3: \{ Fiat, Opel, Peugeot, Citroën \}$

$Q4: \{ VW, Opel, Citroën \}$

$W(Opel) = \{ Q1, Q3, Q4 \}$

$W(VW) = \{ Q1, Q2, Q4 \}$

$W(Ford) = \{ Q1, Q2 \}$

$W(Citroën) = \{ Q3, Q4 \}$

$J(W(Opel), W(VW)) = 2/4 = 1/2$

$J(W(Ford), W(VW)) = 2/3$

$J(W(Citroën), W(Ford)) = 0$

Attribute similarity

We kunnen de similarity tussen een queryterm q en een term t nu als volgt definiëren :

$$S(t,q) = J(W(t), W(q)) * QF(q)$$

Wat was het probleem ook alweer?

- ▶ zero answers
- ▶ many answers

Zero answers

Aanpak

In plaats van stricte match op alle condities
verzwak je je eisen en geef je een top-k selectie
op basis van ranking

Many answers

Aanpak

Breid je notie van similarity uit met de voorstellen die we hiervoor hebben gezien en geef een top-k

Implementatie

- Programma 1: analyseer de data file en de query workload; creëer een metadb ter ondersteuning van o.a. QF en attribute similarity
 - Programma 2: verwerkt een query(reeks) waarbij zero answers en many answers opgelost worden
 - technieken voor numerieke attributen
 - extra: prototype gebruikt een top-k algoritme van Fagin
 - extra: berekening attribute similarity slimmer dan nalopen van alle paren domeinwaarden
 - onderbouwing van je gekozen scorefuncties
- 