Practicum 1 Opzet DAR

Gerben Aalvanger
3987051
Universiteit Utrecht

Sam van der Wal
3962652
Universiteit Utrecht

Docent:
Hans Phillipi
12 mei 2014

1 Metadatabase configuratie

1.1 IDF$_k$

Bereken voor elk attribuut(A_k berekenen we de IDF$_k$ voor elke value v in A_k, h is de bandbreedte parameter en n is het totaal aantal attributen. Voor numerieke attributen berekenen we dit met de formule: (1)

$$IDF_k(v) = log(\frac{n}{\sum_i^n e^{-\frac{1}{2}(\frac{v_i - v}{h})^2}}) \tag{1}$$

Voor gewone attributen berekenen we de IDF met de formule: (2) waarbij F_k(v) het aantal tuples in R is, met A_k = v

$$IDF_k(v) = log(\frac{n}{F_k(v)}) \tag{2}$$

Voor elke tupel voegen we voor elke attribuut A_k een nieuw attribuut IDF$_k$ aan de metadatabase toe met de waarde van de IDF van het attribuut.

1.2 Jaccard

De Jaccard coefficient berekenenen we alleen op categorische attributen.(3) Voor elk categorische attribuut maken we een nieuwe tabel, met de attributen (jaccard, value1, value2) In deze tabel staan 2 mogelijke categorische attribuutwaarden met hun jaccard-coefficient

$$J(W(v), W(q)) = \frac{\mid W(v) \cap W(q) \mid}{\mid W(v) \cup W(q) \mid} \tag{3}$$

1.3 QF

QF kunnen we bereken met behulp van(4)

$$\frac{RQF(v) + 1}{RQFMax + 1} \tag{4}$$

Hierin is RQF(v) de frequentie van waarde v in alle queries en RQFMAX de frequentie van de meest genoemde waarde in de queries. De QF slaan we net als de IDF op door een attribuut toe te voegen in de metadatabase. Dit attribuut bevat de waarde van de QF van de value van het oorspronkelijke attribuut. De QF wordt tegelijk met Jaccard berekend op het moment dat wij de jaccard van een value v met zichzelf berekenen. Op dit moment hogen wij de RQF(v) op. Als alle RQF(v) bekend zijn dan berekenen we met de RQFMAX de QF voor alle attribuutwaarden.

2 Query verwerken

2.1 IDF

Voor een query ranken wij door allereerst voor elk gegeven attribuut de similaritycoefficient te berekenen op basis van de IDF. Voor nummerieke attributen gebruiken wij de formule: (5)

$$S_{num}(v,q) = e^{-\frac{1}{2}(\frac{v-q}{h})^2} IDF(q) \tag{5}$$

Voor een enkel categorisch attribuut gebruiken wij de formule: (10)

$$S_{cat}(v,q) = \begin{cases} \text{Als q = v} & IDF(v) \\ \text{Anders} & 0 \end{cases} \tag{6}$$

Voor een enkel categorisch attribuut met meerdere mogelijke attribuutwaarden gebruiken wij de formule: (7)

$$S_{cat+}(v,q) = \begin{cases} \text{Als q in v} & IDF(v) \\ \text{Anders} & 0 \end{cases} \tag{7}$$

2.2 QF

Voor de similaritycoefficient berekenen kunnen we met QF gebruik maken van de formule: (8)

$$S(v,q) = \begin{cases} \text{Als q = v} & QF(v) \\ \text{Anders} & 0 \end{cases} \tag{8}$$

2.3 JACCARD

Jaccard is te gebruiken in combinatie met IDF of/en QF, we veranderen de similaritycoefficient naar:

$$S(v,q) = J(W(t), W(q))QF(q) \tag{9}$$

Als de Jaccard-coefficient niet in de tabel aanwezig is, is deze 0, tenzij q = v, dan is de coefficient 1.

2.4 combinaties

De combinatie van QF, IDF en JACCARD levert de volgende formule op:

$$S(v,q) = J(W(v), W(q))QF(q)IDF(q) \tag{10}$$

Dit geld enkel voor categorische waarden. Voor numerieke waarden gebruiken wij enkel de IDF uit formule (5)

3 Top-k

In onze metadata gebruiken wij voor alle attributen een aparte index. Dit betekent wel dat we lang aan het preprocessen zijn om de metadata te krijgen. Maar hierna kunnen we het treshold algoritme sneller resultaten laten opleveren. Door deze manier kunnen we dus met indexes snel de queries laten verlopen. Buiten de query om zullen we dan de top-k selectie maken. De query levert dan wel een gesorteerde lijst op, maar het algortime hoeft niet langs alle n elementen van de database. Als we zoeken naar value V doen wij het volgende bij categorische waarden: We selecteren de bovenste (ordening v.a. grootste) coefficienten van de QF, IDF en de JACCARD (waar value1 $\in V$), dit gaat handig met een index.

Voor numerieke attributen hebben we twee ideeen om de dichtsbijzijnde waarden bij value v te selecteren.

1: Als de gezochte numerieke waarde v is, zoeken we in een range van 0.9 * v en 1.1 * v. Hierbij kennen we waardes die dichter bij x liggen een hogere score toe dan waardes die daar verder van af liggen. Daarna sorteren we op deze waarde.

2: We selecteren m waarden die dicht bij v liggen, dit doen hierbij kiezen wij de m/2 kleinste elementen $>$ v en de m/2 grootste elementen $<$ v.