20F-0386
BCS-8A

M-Usama Shahid
IR-Assigment: 05

---

1) NO, it is not necessary for all zones to use the same Boolean Match function in weighted zone scoring, it can be beneficial to use different match functions for different zones based on their importance.

2) Assuming there are three zones and each zone has a boolean match function. The score for the document can be calculated as:-

Score $= g_1 \times s_1 + g_2 \times s_2 + g_3 \times s_3$

Here all distinct score values a document may get:

1) $0.2 \times 0 + 0.31 \times 0 + 0.49 \times 0 = 0$
2) $0.2 \times 0 + 0.31 \times 0 + 1 \times 0.49 = 0.49$
3) $0.2 \times 0 + 0.31 \times 1 + 0.49 \times 0 = 0.31$
4) $0.2 \times 0 + 0.31 \times 1 + 0.49 \times 1 = 0.8$
5) $0.2 \times 1 + 0.31 \times 0 + 0.49 \times 0 = 0.2$
6) $0.2 \times 1 + 0.31 \times 0 + 0.49 \times 1 = 0.69$
7) $0.2 \times 1 + 0.31 \times 1 + 0.49 \times 0 = 0.51$
8) $0.2 \times 1 + 0.31 \times 1 + 0.49 \times 1 = 1$

3)

```
ZONESCORE (List(eV))
float score[N] = [0]
constant g[1]
PL- merge (List(eV))
while p (is not) NIL
        Scores[docID(P)] = weightZone (P.g)
        PL- next (P)
return (scores)
```

4) WeightZone(P_1, P_2, g)
S ← ()
Scores[docID(P_1)] = 0
for i ← 1 to 1
S[i] ← BooleanScore(q, docID(P_1))
Scores[docID(P_1)] = Scores[docID(P_1)] + g[i] * S[i].

5) By using the equation

$$g = \frac{n_{1or} \times n_{0in}}{n_{1or} + n_{1on} + n_{0ir} + n_{0in}}$$

Given the sample training set.
n_{1or} = 3 (from ϕ1, ϕ3, ϕ5)
n_{1on} = 1 (from ϕ2)
n_{0ir} = 0
n_{0in} = 1 (from ϕ4)

So,

$$g = \frac{3 \times 1}{3+1+0+1} = 3/5 = 0.6.$$

6) Phi g Relevance judgment

1 1R
2 3/4 NR
3 3/4 R
4 0NR
5 1 R
6 3/4 R
7 1/4 NR.

7) In cases where st(dt, qt) and sb(dt, qt) have same values, score is independent of g and so these cases do not play a role in optimizing g.

8) idf always finite when

$df_t \geq 1 \Rightarrow idf_t \leq \log N$.

9) $IDF(t) = \log(N/df(t))$

$IDF(t) = \log(N/N) \Rightarrow \log(1) = 0$

So, the IDF of a term that occurs in every document is 0. For a word that occurs in every document putting it in the stop list has the same effect as idf weighting: the word is ignored.

10)

	D1	D2	D3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

the tf-idf weights for the terms:

[redacted]	Doc1	Doc2	Doc3
car	44.55	6.6	39.6
auto	6.24	68.64	0
insurance	0	53.46	46.98
best	21	0	25.5

11) Yes, the tf-idf weight of a term in a document exceed 1.

12) For any base $b > 0$, $idf_t = \log_b(N/df_t)$

$= (\log_b 10) * (\log_{10}(N/df_t))$

$= C * (\log(N/df_t))$

where C is constant.

$tf\text{-}idf_{t,d,b} = tf_{t,d} * idf_t$

$= tf_{t,d} * C * (\log(N/df_t)) = C * tf\text{-}idf_{t,d}$

$Score(q,d,b) = \sum_q tf\text{-}idf_{t,b,d} = C * \sum_q tf\text{-}idf_{t,d}$

So, changing the base changes the score by a factor $C = (\log_b 10)$
The relative scoring of documents remains unaffected by changing the base.

13) It is the number of bits in the boolean representation of the idf.

14) If "jealous" and "jealousy" both appear in a document, they would contribute to the TF of the stem "jealous".

- The modified TF for a stemmed term ts in a document d can be calculated as.

$$TF_{stem}(ts, d) = \frac{\text{Number of times stemmed term ts appeer in d.}}{\text{Total number of term in d.}}$$

- The modified IDF for a stemmed term Ts can be calculated as:

$$IDF_{stem}(ts) = \log\left(N / df_{stem}(ts)\right)$$

15) Doc 1 = (0.897, 0.125, 0, 0.423)
Doc 2 = (0.076, 0.786, 0.613, 0)
Doc 3 = (0.595, 0, 0.706, 0.383).

16)
Doc 1 = $(0.897)^2 + (0.125)^2 + (0)^2 + (0.423)^2 = 0.999$
Doc 2 = $(0.076)^2 + (0.786)^2 + (0.613)^2 + (0)^2 = 0.999$
Doc 3 = $(0.595)^2 + (0)^2 + (0.706)^2 + (0383)^2 = 0.999$

Because they are normalized (unit) vectors.

18) $\Sigma(q_i - w_i)^2 = \Sigma q_i^2 - 2\Sigma q_i w_i + \Sigma w_i^2$

$= 2(1 - \Sigma q_i w_i)$

Thus:

$\Sigma(q_i - v_i)^2 < \Sigma(q_i - w_i)^2 \iff 2(1 - \Sigma q_i v_i) < 2(1 - \Sigma q_i w_i) \iff \Sigma q_i v_i > \Sigma q_i w_i$

19)

Word	query					document			
	tf	wf	df	idf	q_i = wf·idf	tf	wf	d_i = normalize wf	$q_i \cdot d_i$
digital	1	1	10,000	3	3	1	1	0.52	1.56
video	0	0	100,000	2	0	1	1	0.52	0
Cameras	1	1	50,000	2.3	2.3	2	1.3	0.68	1.56

Similarity, Score = 1.56 + 1.56 = 3.12.

Normalize similarity score is also correct = 3.12 / length (query)

= 3.12/3.78

= 0.825.

20) q_i = affection

v(q) = (1, 0, 0), v(SaS) = (0.996, 0.087, 0.017)

v(PaP) = (0.993, 0.120, 0), v(WH) = (0.847, 0.466, 0.254)

v(q) · v(SaS) = 0.996

v(q) · v(PaP) = 0.993

v(q) · v(WH) = 0.847

So, order of scores in this case is the reverse as in the query jealous gossip.

21) We can assign weights to query terms according to their idf in the collection, or use other collection statistics

22) Omit this term from the query and proceed, its contribution to the dot product with any documents will be zero.

24) NTC weight contribution by coyotes insurance in any document vector = weight contribution by coyote + weight contribution by insurance = 0 + k (can be calculated) (since coyote does not occur in any document of the collection). The otc contributed by coyote in the query vector need not to be calculated as the ntc weights for all documents is 0 for coyotes.