CS 533 - Information Retrieval Systems - Homework V

Alican Büyükçakır
alicanbuyukcakir@bilkent.edu.tr

due 28.12.2017

1 Maximal Marginal Relevance

MMR is defined as follows:

$$MMR = \underset{d_i \in R-S}{\operatorname{argmax}}[\lambda\ s(d_i, q) - (1-\lambda)\ \max_{d_j \in S}\{s(d_i, d_j)\}]$$

And in the question, we are given:

$$s(q, d_1) = 0.80$$
$$s(q, d_2) = 0.70$$
$$s(q, d_3) = 0.40$$
$$s(q, d_4) = 0.60$$

Initially, our result set S is always empty. Therefore, we pick d_1 as our first element to S no matter what. After each part, the corresponding diversity value is found by the following formula:

$$div(S) = 1 - \frac{1}{\binom{|S|}{2}} \sum_{\forall d_i, d_j} s(d_i, d_j)$$

(a) When $\lambda = 1$:

When $\lambda = 1$, diversity term is always zero. We only care about the similarity of the documents w.r.t. the query.

$$MMR(\lambda = 1) = \underset{d_i \in R-S}{\operatorname{argmax}}\, s(d_i, q)$$

We firstly pick d_1, since $s(q, d_1) = 0.80$ and it is the largest. d_2 follows that with $s(q, d_2) = 0.70$. Hence, as a result, $S = \{d_1, d_2\}$.

Corresponding diversity value $div(S) = 1 - s(d_1, d_2) = 1 - 0.67 = 0.33$

(b) When $\lambda = 0$:

When $\lambda = 0$, this time the relevance term is zero and the diversity term dominates.

$$MMR(\lambda = 0) \; = \; \underset{d_i \in R-S}{\text{argmax}}[-\max_{d_j \in S}\{s(d_i, d_j)\}] \; = \; \underset{d_i \in R-S}{\text{argmin}} \; \max_{d_j \in S}\{s(d_i, d_j)\}$$

Firstly, S is empty, we pick d_1 as it is the most relevant to the query. Then, to maximize the diversity term in MMR, we minimize the negative similarity (comparing all docs with $S = \{d_1\}$). We have: $s(d_1, d_2) = 0.67$, $s(d_1, d_3) = 0.50$ and $s(d_1, d_4) = 0.20$. We pick the smallest among these, which will make the whole equation maximum. That element is d_4. The resulting set is $S = \{d_1, d_4\}$.

Corresponding diversity value $div(S) = 1 - s(d_1, d_4) = 1 - 0.2 = 0.8$

(c) When $\lambda = 0.5$:

When $\lambda = 0.5$, we give equal importance to the relevance term and the diversity term.

$$MMR(\lambda = 0.5) = \underset{d_i \in R-S}{\text{argmax}}[(0.5)\; s(d_i, q) - (0.5)\; \max_{d_j \in S}\{s(d_i, d_j)\}]$$

Initially, we pick d_1 again. Then:

$$MMR(S + d_2) = (0.5)(0.70) - (0.5)(0.67) = 0.015$$
$$MMR(S + d_3) = (0.5)(0.40) - (0.5)(0.50) = -0.05$$
$$MMR(S + d_4) = (0.5)(0.60) - (0.5)(0.20) = 0.20$$

Therefore, we pick d_4 which has the bigger MMR. Resulting set is $S = \{d_1, d_4\}$.

Corresponding diversity value $div(S) = 1 - s(d_1, d_4) = 1 - 0.2 = 0.8$

(d) Including the Third Document to S

- **When $\lambda = 1$:** We had $S = \{d_1, d_2\}$. Again, looking at the most relevant document only, we pick d_4 since $s(q, d_4) > s(q, d_3)$. Resulting set is: $S = \{d_1, d_2, d_4\}$. Corresponding diversity value $div(S) = 1 - \frac{1}{3}(0.67 + 0.20 + 0.10) = 1 - 0.323 = 0.677$.
- **When $\lambda = 0$:** We had $S = \{d_1, d_4\}$. This time, we will pick the document that will increase the maximum similarity among S the least. When d_2 is included to S, we get MMR value of -0.67 whereas when d_3 is included to S, we get MMR value of -0.50. We pick d_3. Resulting set is $S = \{d_1, d_3, d_4\}$. Corresponding diversity value $div(S) = 1 - \frac{1}{3}(0.50 + 0.20 + 0.0) = 1 - 0.233 = 0.767$.
- **When $\lambda = 0.5$:** We had $S = \{d_1, d_4\}$. Let's look at the possible actions:

  $$MMR(S + d_2) = (0.5)(0.70) - (0.5)(0.67) = 0.015$$

  $$MMR(S + d_3) = (0.5)(0.40) - (0.5)(0.80) = -0.2$$

  Hence, we pick d_2 which has the bigger MMR value. Resulting set is $S = \{d_1, d_2, d_4\}$. Corresponding diversity value $div(S) = 1 - \frac{1}{3}(0.67 + 0.20 + 0.10) = 1 - 0.323 = 0.677$

Considering the results of all parts and the diversity values that are obtained by the definition of diversity we are given, we can claim that MMR does its job, as decreasing λ constant actually increases the diversity of the resulting sets while decreasing the relevance (accuracy) of the resulting set.

2 Performance Evaluation for Diversification

(a) S-Recall

S-Recall@N is the fraction of unique meanings covered at first N documents. By this definition,

$$S - Recall@5 = \frac{6}{6} = 1$$
$$S - Recall@10 = \frac{6}{6} = 1$$

(b) Intent-Aware Precision

Precision-IA@N is the number of meanings covered at first N documents divided by all possible meanings that can be covered at first N documents. By this definition:

$$PIA@5 = \frac{1}{6}(\frac{1}{5} + \frac{1}{5} + \frac{1}{5} + \frac{1}{5} + \frac{1}{5} + \frac{2}{5}) = \frac{7}{30} = 0.233$$

$$PIA@10 = \frac{1}{6}(\frac{2}{10} + \frac{2}{10} + \frac{2}{10} + \frac{2}{10} + \frac{2}{10} + \frac{2}{10}) = \frac{1}{5} = 0.200$$

3 Data Fusion

Throughout the questions, the symbol $>$ is used to indicate a document's rank is better, i.e. a document is ranked higher. $a > b$ means that document a is ranked higher than document b.

(a) Reciprocal Rank

Reciprocal Rank for document i is defined as follows:

$$RR(d_i) = \frac{1}{\sum_{\forall j} \frac{1}{r_{ij}}}$$

where r_{ij} is the rank of document i in the system j. Also, if a document does not exist in a ranking list, then its corresponding term in the denominator is zero. Here are the reciprocal ranks for all documents:

$$RR(a) = \frac{1}{\frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{1}} = \frac{4}{9} = 0.444$$
$$RR(b) = \frac{1}{\frac{1}{1} + \frac{1}{1} + \frac{1}{1} + 0} = \frac{1}{3} = 0.333$$
$$RR(c) = \frac{1}{\frac{1}{4} + 0 + \frac{1}{2} + \frac{1}{2}} = \frac{4}{5} = 0.80$$
$$RR(d) = \frac{1}{\frac{1}{3} + \frac{1}{3} + \frac{1}{3} + \frac{1}{3}} = \frac{3}{4} = 0.75$$
$$RR(e) = \frac{1}{0 + 0 + 0 + \frac{1}{4}} = 4$$
$$RR(f) = \frac{1}{0 + \frac{1}{4} + 0 + 0} = 4$$

According to Reciprocal Ranks, we have: $b > a > d > c > e = f$.

(b) Borda Count

In Borda Count, the highest ranked document gets n vote in an n-way vote scenario. Having 6 different documents, the highest ranked document in a system will get 6 votes, and the next 5 votes etc. Non-ranked documents share the remaining votes evenly [1]. Borda Count of a document is the sum of all Borda Counts throughout all systems.

$$BC(d_i) = \sum_{\forall j} BC_j(d_i)$$

Borda Counts for all documents:

$$BC(a) = 5 + 5 + 3 + 6 = 19$$
$$BC(b) = 6 + 6 + 6 + 1.5 = 19.5$$
$$BC(c) = 3 + 1.5 + 5 + 5 = 14.5$$
$$BC(d) = 4 + 4 + 4 + 4 = 16$$
$$BC(e) = 1.5 + 1.5 + 1.5 + 3 = 7.5$$
$$BC(f) = 1.5 + 3 + 1.5 + 1.5 = 7.5$$

According to Borda Counts, we have: $b > a > d > c > e = f$.

(c) Condorcet

In Condorcet, we count how many times a document won, lost or tied versus another document. We put those into a matrix and sort documents according to how many times in total a document won versus another documents (when tie, look at number of losses) to get our rankings.

The matrix:

Table 1: Condorcet Win-Lose-Draw Table for Documents

	a	**b**	**c**	**d**	**e**	**f**
a	-	(1,3,0)	(3,1,0)	(3,1,0)	(4,0,0)	(4,0,0)
b	(3,1,0)	-	(3,1,0)	(3,1,0)	(3,1,0)	(3,0,1)
c	(1,3,0)	(1,3,0)	-	(2,2,0)	(3,0,1)	(3,1,0)
d	(1,3,0)	(1,3,0)	(2,2,0)	-	(4,0,0)	(4,0,0)
e	(0,4,0)	(1,3,0)	(0,3,1)	(0,4,0)	-	(1,1,2)
f	(0,4,0)	(0,3,1)	(1,3,0)	(0,4,0)	(1,1,2)	-

And the table that shows how many times a document won/lost/tied:

Table 2: Condorcet Count Results

	WIN	**LOSE**	**TIE**
a	4	1	0
b	5	0	0
c	2	2	1
d	2	2	1
e	0	4	1
f	0	4	1

Hence, according to Condorcet Count, we have: $b > a > c = d > e = f$.

4 Signature Files

For 512,000 objects, each one having a signature of size 1024 bits (128 bytes), we will have same signature file in terms of size for both Sequential and Bit-Sliced Signature files. The difference will be the way they are put to the memory: the former is put in row-major order whereas the other one in column-major order.

- **Size for Sequential Signatures (SS):** $512k * 128$ bits $= 64$ MB.
- **Size for Bit-Sliced Signatures (BS):** $512k * 128$ bits $= 64$ MB.

5 Number of Page Accesses

Page size is $0.5KB = 512$ bytes $= 4,096$ bits. Since each signature has 1024 bits, a page can hold $\frac{4096}{1024} = 4$ signatures in SS. On the other hand, each column has 512,000 bits in Bit-Sliced Signatures, which means that we need $\frac{512,000}{4,096} = 125$ pages to store one column in BS.

- **Number of Page Accesses for Sequential Signatures (SS):** We need to access all pages to check whether the query signature and the the block signatures match (AND'ing results on Q_s). We have 512,000 signatures, and pages can contain 4 signatures. Therefore:
$$PA_{SS} = \frac{512,000}{4} = 128,000$$
- **Number of Page Accesses for Bit-Sliced Signatures (BS):** We need to access all the columns that correspond to the bit position that is 1. Hence, we need to access 5 columns, each one consisting of 125 pages.
$$PA_{BS} = 125 * 5 = 625$$

6 Fixed Prefix with $k = 2$

For the given signatures, when FP with $k = 2$ is applied, we get the following buckets:

Table 3: Partitions as a result of Fixed Prefix with $k = 2$

Partition	**00**	**01**	**10**	**11**
Signatures	S5, S8	S1, S2	S6, S7, S9, S10	S3, S4

Consider the following queries for the upcoming parts of this question:

- Q1: 1110 0001
- Q2: 0110 0011
- Q3: 1100 1100

(a) PAR and SAR values

The partition is to be accessed if `AND`ing Query signature's prefix with our prefix yields query's signature. Partition Activation Ratio is the number of partitions to be accessed over all partitions. Signature Activation Ratio is the number of signatures to be accessed in the partitions over all signatures.

- For Q1: It starts with 11. Only possible partition it can go is 11.
- For Q2: It starts with 01. It can go to both 01 and 11.
- For Q3: It starts with 11. Only possible partition it can go is 11.

Table 4: PAR and SAR Values

Query	**Q1**	**Q2**	**Q3**
PAR	1/4	2/4	1/4
SAR	2/10	4/10	2/10

(b) Avg Turnaround Time for Sequential and Parallel Processing

I assume that each access to a partition takes a unit time. We will spend 1 unit time to process $Q1$, 2 units of time for $Q2$ and 1 unit of time for $Q3$. Also, I assume that the queries come to the system at $t = 0$, in the given order (Q1, Q2, Q3).

- **Sequential Processing:**

Query:	Q1	Q2	Q3
t_{end} :	1	3	4

Hence, Avg Turnaround Time for Sequential Processing is:

$$ATT(Seq) = Avg(t_{end} - t_{arrival}) = \frac{1}{3}((1-0) + (3-0) + (4-0)) = \frac{8}{3} = 2.666$$

- **Parallel Processing:** $PE(XX)$ represents the Parallel Processing Engine that is responsible for the queries that correspond to prefix XX.

PE(00)	←	∅		
PE(01)	←	Q2		
PE(10)	←	∅		
PE(11)	←	Q1	Q2	Q3
t_{end} :		1	2	3

Hence, Avg Turnaround Time for Parallel Processing is:

$$ATT(Par) = Avg(t_{end} - t_{arrival}) = \frac{1}{3}((1-0) + (2-0) + (3-0)) = \frac{6}{3} = 2$$

Speed-up ratio is how much we sped-up the process by using parallel processing.

$$SUR = \frac{ATT(Seq)}{ATT(Par)} = \frac{8/3}{2} = \frac{4}{3} = 1.333$$

7 EPP and FKP (Lee and Leng 1989 paper)

According to Table 5, the partitions and the signatures in them are as follows:

(a) Extended Prefix Partitioning (z=2)

In EPP, we form the partitions according to the bits until a specific number of zeroes is seen. In this case, $z = 2$, which means we create partitions from the signatures until we see 2 zeroes [2].

Table 5: EPP and FKP Partitions for the Given Signatures

S	**Signatures**	Partitions according to...	**EPP (z=2)**	**FKP (k=2)**
S1	0110 1010	$\rightarrow$	0110	01-1
S2	0100 0110	$\rightarrow$	010	00-3
S3	1110 0011	$\rightarrow$	11100	00-5
S4	1100 0011	$\rightarrow$	1100	00-3
S5	0011 1010	$\rightarrow$	00	00-1
S6	1010 0101	$\rightarrow$	1010	10-1
S7	1011 0010	$\rightarrow$	10110	00-5
S8	0000 1111	$\rightarrow$	00	00-1
S9	1010 0110	$\rightarrow$	1010	10-1
S10	1011 0100	$\rightarrow$	10110	00-7

Table 6: Partitions as a Result of EPP

Partition	**00**	**010**	**0110**	**1010**	**10110**	**1100**	**11100**
Signatures	S5, S8	S2	S1	S6, S9	S7, S10	S4	S3

(b) Floating Key Partitioning (k=2)

In FKP, we are examining each consecutive non-overlapping k-substrings in the signatures to create out partitions. Leftmost substring with smallest weight is chosen as a key. Since $k = 2$, we are looking for 2-substrings in the signatures. Therefore, we need to look at the occurrences of **00**s and denote in which bit we encountered the zeroes. If we cannot find any, we look at the first 2 bits, which has to be either **01** or **10**. We also denote those with the index they are seen, which is 1 [2].

Table 7: Partitions as a Result of FKP

Partition	**00-1**	**00-3**	**00-5**	**00-7**	**01-1**	**10-1**
Signatures	S5, S8	S2, S4	S3, S7	S10	S1	S6, S9

(c) Pages that Need to be Accessed for EPP and FKP

1. **For EPP:** AND the variable-sized prefix of our queries with possible partition prefixes. If the result is the query's prefix, then access that page.
   - **Q1: 1110 0001.** AND'ing this query with only the partition **11100** yields the query's prefix itself. Hence, we access to only that partition.
   - **Q2: 0110 0011.** AND'ing this query with partitions **0110** and **11100** yields the query's prefix itself. Need to access 2 partitions.
   - **Q3: 1100 1100.** AND'ing this query with partitions **1100** and **11100** yields the query's prefix itself. Need to access 2 partitions.
2. **For FKP:** For each query, look at the first 2 bits and search for **01** or **10**. Otherwise, look at the nonoverlapping consecutive **00**s.
   - **Q1: 1110 0001.** Need to access **00-5** partition only.
   - **Q2: 0110 0011.** Need to access **01-1** and **00-5** partitions. Two of them.
   - **Q3: 1100 1100.** Need to access **00-3** and **00-7** partitions. Two of them.

As a result, we needed to access 5 partitions in both of the partitioning methods.

8 Linear Hashing Using Suffixes (Quick Filter)

When the LF is reached, we add $Bkfr \times LF = 3 \times \frac{2}{3} = 2$ more documents and then re-distribute the current page by adding one more page to the end of the file and one more bit to the representation of the page.

Table 8: Iterations of Quick Filter

	0	**S1**	**S2**						The first 4 docs are added. LF is reached.
	1	**S3**	**S4**						Add 2 more blocks: S5 and S6.
bv →	0	S1	S2	**S5**					
	1	S3	S4	**S6**					Now distribute! Increment bv.
	00								
bv →	1	S3	S4	S6					LF is reached.
	10	S1	S2	S5					Add 2 more blocks: S7, S8.
	00								
bv →	1	S3	S4	S6	-	**S8**			Now distribute! Increment bv.
	10	S1	S2	S5	-	**S7**			bv will be power of two. Reset!
bv →	00								
	01	S6							
	10	S1	S2	S5	-	S7			LF is reached.
	11	S3	S4	S8					Add 2 more blocks: S9 and S10.
bv →	00	**S10**							
	01	S6							
	10	S1	S2	S5	-	S7	**S9**		
	11	S3	S4	S8					Now distribute!
	000								
bv →	01	S6							
	10	S1	S2	S5	-	S7	S9		
	11	S3	S4	S8					
	100	S10							End.

For a query to access to a page, we look at the k-suffix of the query. If it does not exist in our page table, we increase k, until we have the same-sized suffixes in our table. Then, we again AND the query signature suffix with table signature. If the result is the query suffix, then we try that block.

- For Q1: First valid suffix is 01. Therefore, possible pages to be accessed are: **01** and **11**.
- For Q2: First valid suffix is 11. The only possible page to be accessed is **11**.

9 Information Filtering Using Ranked Key Method

Ranked Key Method is as follows: We create the directory for each highest-ranked document in the profiles. For each highest-ranked document, we create its postings list consisting of elements that has (1) Profile Name, (2) The number of preceding docs after that document, (3) The list of preceding documents for that profile [4]. Hence, the resulting directory with the inverted index structure is as follows:

Table 9: Directory and Inverted List Structure

Directory			Inverted Lists										
a	o	→	P1	5	b, c, d, e, f		P2	3	b, e, f		P5	2	c, f
b	o	→	P3	2	c, f		P4	2	d, f				
c	X												
d	X												
e	X												
f	X												

References

[1] Nuray, Rabia and Can, Fazli. Automatic Ranking of Information Retrieval Systems Using Data Fusion. Information Processing & Management 42.3 (2006): 595-614.

[2] Lee, Dik Lun, and Chun-Wu Leng. Partitioned signature files: Design issues and performance evaluation. ACM Transactions on Information Systems (TOIS) 7.2 (1989): 158-180.

[3] Zezula, Pavel, Fausto Rabitti, and Paolo Tiberio. Dynamic partitioning of signature files. ACM Transactions on Information Systems (TOIS) 9.4 (1991): 336-367.

[4] Yan, Tak W., and Hctor Garca-Molina. Index structures for selective dissemination of information under the boolean model. ACM Transactions on Database Systems (TODS) 19.2 (1994): 332-364.