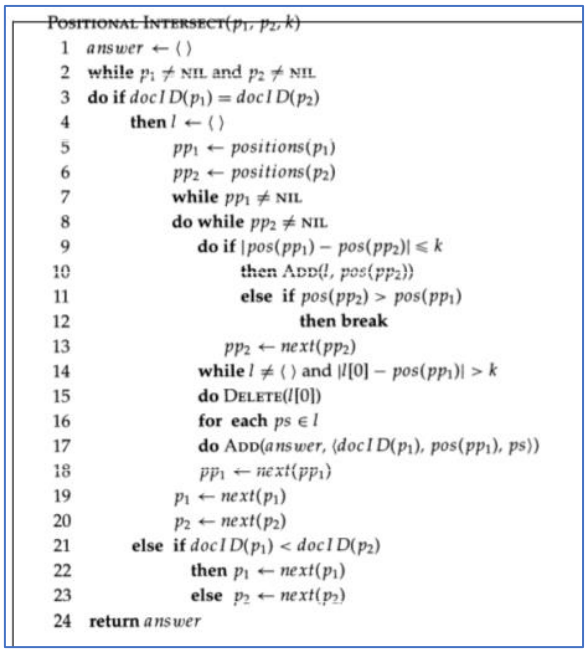
COMP6714 Assignment1 Meng Zhang z5187943

Q1:

Some Boolean retrieval systems (e.g., Westlaw) support the following proximity operators: /k, /S, and /P. Describe a simple modification to the positional inverted index to support all these three proximity operators.

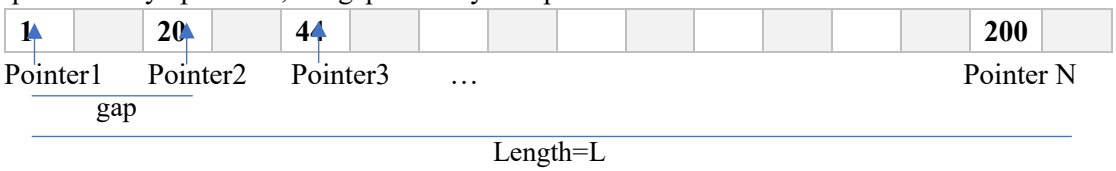
Propositional index	<term, number of docs containing term; Doc1: position1, position2…; Doc2: position1, position2…; Etc.>
Then actual Algorithm of /k in textbook	 POSITIONAL INTERSECT(p_1, p_2, k) 1 $answer \leftarrow \langle\,\rangle$ 2 **while** $p_1 \neq$ NIL and $p_2 \neq$ NIL 3 **do if** $docID(p_1) = docID(p_2)$ 4 **then** $l \leftarrow \langle\,\rangle$ 5 $pp_1 \leftarrow positions(p_1)$ 6 $pp_2 \leftarrow positions(p_2)$ 7 **while** $pp_1 \neq$ NIL 8 **do while** $pp_2 \neq$ NIL 9 **do if** $\lvert pos(pp_1) - pos(pp_2)\rvert \leqslant k$ 10 **then** ADD(l, $pos(pp_2)$) 11 **else if** $pos(pp_2) > pos(pp_1)$ 12 **then break** 13 $pp_2 \leftarrow next(pp_2)$ 14 **while** $l \neq \langle\,\rangle$ and $\lvert l[0] - pos(pp_1)\rvert > k$ 15 **do** DELETE($l[0]$) 16 **for each** $ps \in l$ 17 **do** ADD($answer$, $\langle docID(p_1), pos(pp_1), ps\rangle$) 18 $pp_1 \leftarrow next(pp_1)$ 19 $p_1 \leftarrow next(p_1)$ 20 $p_2 \leftarrow next(p_2)$ 21 **else if** $docID(p_1) < docID(p_2)$ 22 **then** $p_1 \leftarrow next(p_1)$ 23 **else** $p_2 \leftarrow next(p_2)$ 24 **return** $answer$
Examples for /k Word1 /10 Word2	<word1, m; Doc1: 1, 22, 28, 44…; Doc2: 3, 100, 156, 200…; Etc.> <word2, n; Doc1: 3, 10, 24, 88…; Doc2: 20, 88, 162, 280…; Etc.> When word 2's position – word1's position < 10 in the same document, this can be recorded. And the algorithm should be like the picture above.
Examples for /s	Except for the /k modification, we should add the locations of the end of sentences, such as '.?!'…etc. in the whole document, and if the two words location is in the same period of sentence, this can be recorded. <'.', n;

	Doc1: position1, position2 …; Doc2: position1, position2…; Etc.>
Examples for /p	Except for the /k modification, we should add the locations of '\n' or '\r\n' in the whole document. And if the two words location is in the same gap of '\n's, this can be recorded. <'\n', n; Doc1: position1, position2 …; Doc2: position1, position2…; Etc.>

Q2:

(1) Prove that choosing $\sqrt{L}$ skip pointers has the best worst-case performance.

Firstly, we assume that the length of the posting queue is L, and we put n skip pointers into the queue evenly-space. So, the gap of every two pointers is L/n



Because sequential search in the skip pointers array, sequential search in the target segment and every posting in the posting list has the same probability of being searched. The worst condition should be that the target just at the end of the whole queue.

Hence the cost of pointers is n, and the cost of searching on the segments is $\frac{L}{n}$.

Hence the total cost of both searching is

$$\text{Total cost} = n + \frac{L}{n}$$

And we want to keep the total cost as the minimum, so we calculate the derivation of total cost.

$$\text{Total cost}' = 1 - \frac{L}{n^2} = 0$$

$$1 = \frac{L}{n^2}$$

$$\text{L} = n^2$$

When $n > \sqrt{L}$, the derivation is over 0, and when $n < \sqrt{L}$, the derivation is less than 0. Hence when $n=\sqrt{L}$, the total cost can be the minimum. So, the $n=\sqrt{L}$ is the best worst-case performance under this kind of searching.

(2) Find the best number of skip pointers if we perform binary searches both Step I and II.

We use the model of (1), at this time, we perform binary searches both Step I and II, so the cost of pointers is $\log_2 n$, and the cost of searching on the segments is $\log_2 \frac{L}{n}$.

Hence the total cost of both searching is

$$\text{Total cost} = \log_2 n + \log_2 \frac{L}{n} = \log_2 L$$

Due to the rule of logarithm operation, the number of the pointers n has been deleted from the function, so the number of the pointers won't influence the result of the total cost. When we all use binary search in both processes, we can just regard the whole process as a binary search, whose efficiency always be $\log_2 L$.

(3) Find the best number of skip pointers if we perform binary search in Step I and sequential search in Step II.

Because binary search in the skip pointers array, sequential search in the target segment and every posting in the posting list has the same probability of being searched.

Hence the cost of pointers is $\log_2 n$, and the average cost of searching on the segments is $\frac{L}{2n}$.

Hence the total cost of both searching is

$$\text{Total cost} = \log_2 n + \frac{L}{2n}$$

And we want to keep the total cost as the minimum, so we calculate the derivation of total cost.

$$\text{Total cost}' = \frac{1}{nln2} - \frac{L}{2n^2} = 0$$

$$\frac{1}{nln2} = \frac{L}{2n^2}$$

$$\text{n} = \frac{Lln2}{2}$$

When $\text{n} > \frac{Lln2}{2}$, the derivation is over 0, and when $\text{n} < \frac{Lln2}{2}$, the derivation is less than 0.

Hence when $\text{n} = \frac{Lln2}{2}$, the total cost can be the minimum. So, the $\text{n} = \frac{Lln2}{2}$ is the best worst-case performance under this kind of searching.

Q3:

(1) Show that the maxscore for each keyword can be computed without examining the postings list.

$$\text{score}(\text{d}, \text{Q}) = \sum_{t \in Q} idf_t \cdot \frac{(k_1+1)tf_{t,d}}{k_1\left((1-b) + b\frac{L_d}{L_{ave}}\right) + tf_{t,d}} \cdot \frac{(k_3+1)tf_{t,Q}}{k_3 + tf_{t,Q}}$$

$$= \sum_{t \in Q} idf_t \cdot \frac{(2+1)tf_{t,d}}{2\left((1-0) + 0\frac{L_d}{L_{ave}}\right) + tf_{t,d}} \cdot \frac{(2+1)tf_{t,Q}}{2 + tf_{t,Q}}$$

$$= \sum_{t \in Q} idf_t \cdot \frac{3tf_{t,d}}{2 + tf_{t,d}} \cdot \frac{3tf_{t,Q}}{2 + tf_{t,Q}}$$

As the question said, the query is {A,B,C}. So the $tf_{t,Q}$ is fixed to be 1.

$$\text{score}(\text{d},\text{Q}) = \sum_{t \in Q} idf_t \cdot \frac{3tf_{t,d}}{2 + tf_{t,d}} \cdot \frac{3tf_{t,Q}}{2 + tf_{t,Q}}$$

$$= \sum_{t \in Q} idf_t \cdot \frac{3tf_{t,d}}{2 + tf_{t,d}}$$

$$= idf_t \cdot \frac{3max_{tf}}{2 + max_{tf}}$$

Hence, for A:

Maximum tf=1	$\text{maxscore}(A,d,\text{Q}) = 6 \cdot \frac{3 \cdot 1}{2+1} = 6$
Maximum tf=10	$\text{maxscore}(A,d,\text{Q}) = 6 \cdot \frac{3 \cdot 10}{2+10} = 15$
Maximum tf=100	$\text{maxscore}(A,d,\text{Q}) = 6 \cdot \frac{3 \cdot 100}{2+100} = 18$
Maximum tf=1000	$\text{maxscore}(A,d,\text{Q}) = 6 \cdot \frac{3 \cdot 1000}{2+1000} = 18$

And as the same precess for B and C:

	B	C
Maximum tf=1	2	1
Maximum tf=10	5	2.5
Maximum tf=100	6	3
Maximum tf=1000	6	3

As the increase of max tf, we can improve that

$$\text{maxscore}(X,d,\text{Q}) = \lim_{tf \to \infty} idf_t \cdot \frac{3max_{tf}}{2 + max_{tf}} \approx idf_t \cdot 3$$

(2) Using the maxscore obtained above, determine the postings that are accessed for scoring by the algorithm. You need to assume that each skipTo(x) call "magically" moves the cursor directly to the first posting with document ID at least x (i.e., it does not access any other postings).

Docs	1	2	3	4	5	6	7	8	9	10	11	…
A	1	8			3			10				
B	1				4	1	4					
C	1	2		1	2	3		1	1	3	7	
scores	9	15.9	0	3	16.3	3.8	4	16	1	1.8	2.33	
Max(A)=18 Max(B)=6 Max(C)=3												

Firstly, our target is to find the top-2 results whose score are the most ones. All the candidates should include document A as the MAXSCORE algorithm, because B+C cannot exceed 9 no matter there are how many B and C. So A is the necessary components of all possible candidates.

1. So we can only start with Doc1 and Doc2. Top2-Scores(9,15.9):
   Score(doc1)=6+2+1=9 and Score(doc2)=14.4+1.5=15.9
2. For there's no A in Doc3 and Doc4, so we skip doc3 and doc4.
3. Score(doc5)=10.8+4+1.5=16.3 which is larger than 9 in scores(9,15.9), so we change 9 to 16.3, Scores(9,15.9) turn to Scores(15.9,16.3).
4. For no A documents included, we skip Doc6 and Doc7.
5. Score(doc8)=15+1=16, which is larger than 15.9 in Scores(15.9,16.3), so change Scores(15.9,16.3) into Scores(16,16.3).
6. Because there are no A in Doc9 Doc10 and Doc11, so skip them.
7. After all, due to the Scores(16,16.3), we can get the top2 documents are Doc5 and Doc8.

Q4:

(1) What is the precision of the system on the top-20?

Precision=P(relevant|retrieved)=6/20*100%=30%

(2) What is the F1 on the top-20?

$$\mathrm{F} = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

$$\mathrm{F}_1 = \frac{1}{0.5\frac{1}{P} + (1-0.5)\frac{1}{R}} = \frac{(1+1)PR}{1*P+R} = \frac{2PR}{P+R}$$

Recall=P(retrieved|relevant)=6/8*100%=75%

Hence,

$$\mathrm{F}_1 = \frac{2PR}{P+R} = \frac{2*0.3*0.75}{0.3+0.75} \approx 0.429$$

(3) What is/are the uninterpolated precision(s) of the system at 25% recall?

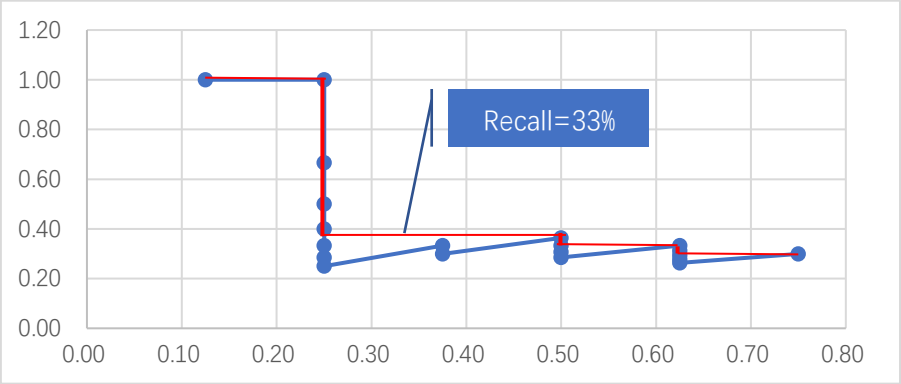
Due to the table above, the whole process can be converted into the table:

No.	R/N	P	R	No.	R/N	P	R
1	R	1	1/8	11	R	4/11	4/8
2	R	1	2/8	12	N	4/12	4/8
3	N	2/3	2/8	13	N	4/13	4/8
4	N	2/4	2/8	14	N	4/14	4/8
5	N	2/5	2/8	15	R	5/15	5/8
6	N	2/6	2/8	16	N	5/16	5/8
7	N	2/7	2/8	17	N	5/17	5/8
8	N	2/8	2/8	18	N	5/18	5/8
9	R	3/9	3/8	19	N	5/19	5/8
10	N	3/10	3/8	20	R	6/20	6/8

2/8*100%=25%

When R(recall)=25%, the uninterpolated precision(s) of the system are {1,2/3,2/4,2/5,2/6,2/7,2/8}

(4) What is the interpolated precision at 33% recall?



The uninterpolated precision(s) is as the blue lines in the graph. So the interpolated precision(s) should be look like the red lines. So when the recall=33%, we can see the precision is 36% (4/11).

(5) Assume that these 20 documents are the complete result set of the system. What is the MAP for the query?

MAP=(1+1+3/9+4/11+5/15+6/20)/8=0.416

Assume, now, instead, that the system returned the entire 10, 000 documents in a ranked list, and these are the first 20 results returned.

(6) What is the largest possible MAP that this system could have?

The largest MAP should be the condition that the left relevant answers are the 21st and 22nd ones. Hence,

MAP_{max}=(1+1+3/9+4/11+5/15+6/20+7/21+8/22)/8=0.503

(7) What is the smallest possible MAP that this system could have?

And the smallest MAP should be the condition that the left relevant answers are the 9999th and 10000th.

MAP_{min}=(1+1+3/9+4/11+5/15+6/20+7/9999+8/10000)/8=0.416

(8) In a set of experiments, only the top-20 results are evaluated by hand. The result in (5) is used to approximate the range (6) to (7). For this example, how large (in absolute terms) can the error for the MAP be by calculating (5) instead of (6) and (7) for this query?

The error should be the difference between the max and the min MAPs.

Hence,

Error= MAPmax - MAPmin=0.503-0.416=0.871