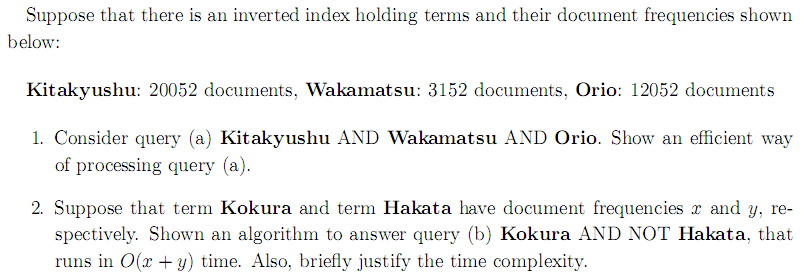
### Description



### Solution

1. Locate Wakamatsu in the dictionary

Retrieve its postings list from the postings file

Locate Orio in the dictionary

Retrieve its postings list from the postings file

Intersect the two postings lists (L)

Locate Kitakyushu in the dictionary

Retrieve its postings list from the postings file

Intersect the postings lists of Kitakyushu and L

Return intersecting to user

1. INTERSECT (p1, p2)

answer 🡨 <>

while p1 ≠ NIL and p2 ≠ NIL

do if docID(p1) < docID(p2)

then ADD(answer, docID(p1))

p1 🡨 next(p1)

else if docID(p1) > docID(p2)

then p2 🡨 next(p2)

else p1 🡨 next(p1)

p2 🡨 next(p2)

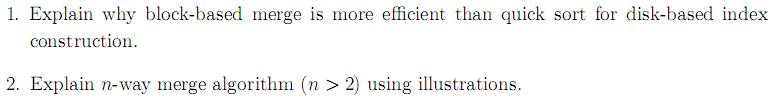
return answer

证明时间复杂度

Time is O(x+y). Instead of collecting documents that occur in both postings lists, collect those that occur in the first one and not in the second, i.e. the all member of two postings lists was scanned once, so they time is O(x+y).

## I-2

### Description



### Solution

1. There are too many disk seeks when constructing index based on disk. Frequent disk inputs and outputs lead to an inefficient speed. And block-based merge has less seek time. So block-based merge is more efficient.

**Sorted File**

1. Merging n blocks:

**Block2**

**Block 1**

…….

**Block N/2**

**…**

**Block 2 N**

**Block 1**

**Block N**

**…**

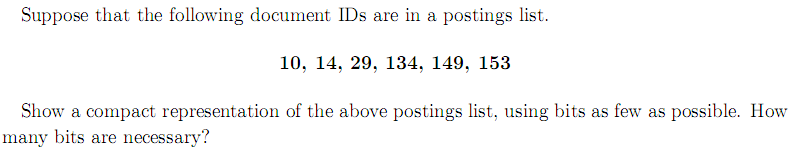
**Block 2**

**Block 1**

**Original File**

## I-3

### Description



### Solution

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| docIDs | 10 | 14 | 29 | 134 | 149 | 153 |
| gaps |  | 4 | 15 | 105 | 15 | 4 |

Method 1: VB code

10 🡪 10001010

4 🡪 10000100

15 🡪 10001111

105 🡪 11101001

15 🡪 10001111

4 🡪 10000100

Total: 10001010 10000100 10001111 11101001 10001111 10000100

Bits: 8\*6 = 48

Method 2: Gamma code

10 🡪 1010 🡪 1110 010

4 🡪 100 🡪 110 00

15 🡪 1111 🡪 1110 111

105 🡪 1101001 🡪 1111110 101001

15 🡪 1111 🡪 1110 111

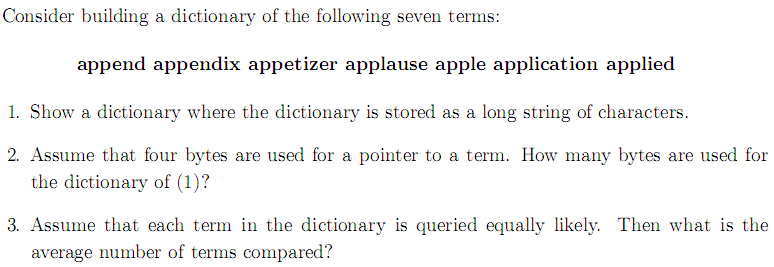
4 🡪 100 🡪 110 00

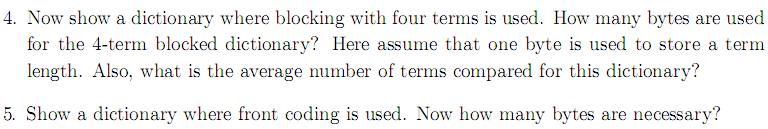
Total: 1110010 11000 1110111 1111110101001 1110111 11000

Bits: 7+5+7+13+7+5 = 44

## I-4

### Description





### Solution

1. Appendappendixappetizerapplauseappleapplicationapplied
2. In the case, when calculating the space of dictionary, I ignore the space of term frequency and term pointer to postings list.

4 bytes per pointer into string

54 bytes for all term in string

Space: 7 × 4 + 54 = 82 bytes

1. (0+1+1+2+2+2+2)/7 = 10/7
2. 6append8appendix9appetizer8applause5apple11application7applied
3. In the case, when calculating the space of dictionary, I ignore the space of term frequency and term pointer to postings list.

4 bytes per pointer into block

61 bytes for all term in string (there are 7 numbers for showing term length)

Space: 4×2+ 61 = 69 bytes

(0+1+2+3+4+1+2)/7 = 13/7

1. 6app\*end5◇endix6◇etizer5◇lause2◇le8◇lication4◇lied

In the case, when calculating the space of dictionary, I ignore the space of term frequency and term pointer to postings list.

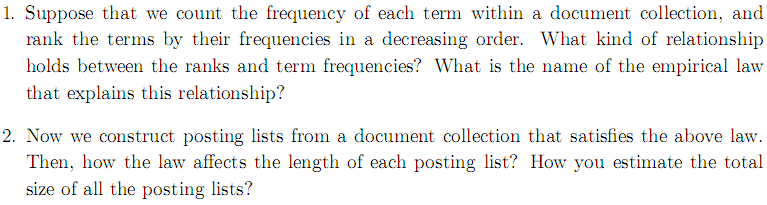
4 bytes per pointer into block

50 bytes for all term in string

Space: 4×2+ 50 = 58 bytes

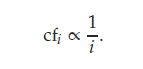
## I-5

### Description



### Solution

1. The relationship holds between the ranks and term frequencies is:

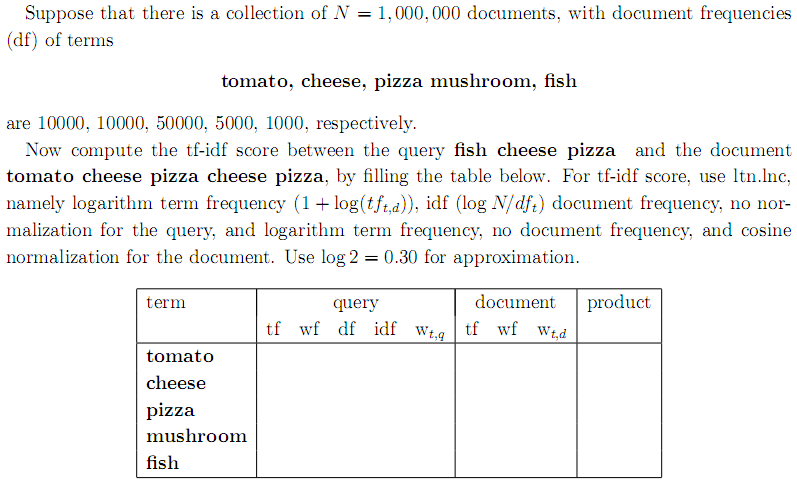


(c is a constant, is the term frequencies, is the rank)

And the name of the empirical law is Zipf’s law. It states that, if is the most common term in the collection, is the next most common, and so on, then the collection frequency of the most common term is proportional to 1/.

## I-6

### Description

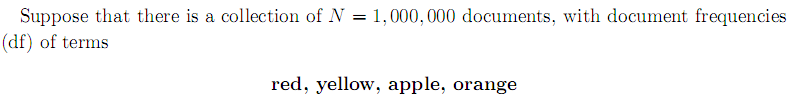


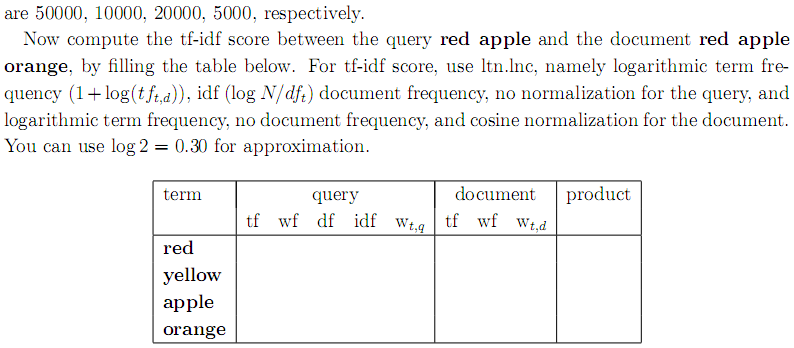
### Solution

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| term | query | | | | | document | | | product |
| tf | wf | df | idf |  | tf | wf |  |
| tomato | 0 | 0 | 10000 | 2 | 0 | 1 | 1 | 0.48 | 0 |
| cheese | 1 | 1 | 10000 | 2 | 2 | 2 | 1.3 | 0.62 | 1.24 |
| pizza | 1 | 1 | 50000 | 1.3 | 1.3 | 2 | 1.3 | 0.62 | 0.81 |
| mushroom | 0 | 0 | 5000 | 2.3 | 0 | 0 | 0 | 0 | 0 |
| fish | 1 | 1 | 1000 | 3 | 3 | 0 | 0 | 0 | 0 |

## I-7

### Description



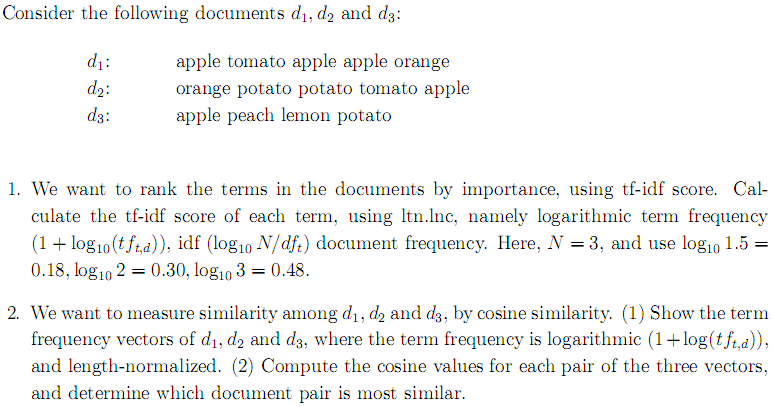


### Solution

同上

## I-8

### Description



### Solution

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| term | **document 1** | | | | | |
| tf | wf | df | idf |  | n’lized |
| apple | 3 | 1.48 | 3 | 0 | 0 | 0 |
| tomato | 1 | 1 | 2 | 0.18 | 0.18 | 0.71 |
| orange | 1 | 1 | 2 | 0.18 | 0.18 | 0.71 |
| potato | 0 | 0 | 2 | 0.18 | 0 | 0 |
| peach | 0 | 0 | 1 | 0.48 | 0 | 0 |
| lemon | 0 | 0 | 1 | 0.48 | 0 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| term | **document 2** | | | | | |
| tf | wf | df | idf |  | n’lized |
| apple | 1 | 1 | 3 | 0 | 0 | 0 |
| tomato | 1 | 1 | 2 | 0.18 | 0.18 | 0.52 |
| orange | 1 | 1 | 2 | 0.18 | 0.18 | 0.52 |
| potato | 2 | 1.3 | 2 | 0.18 | 0.234 | 0.68 |
| peach | 0 | 0 | 1 | 0.48 | 0 | 0 |
| lemon | 0 | 0 | 1 | 0.48 | 0 | 0 |
| term | **document 3** | | | | | |
| tf | wf | df | idf |  | n’lized |
| apple | 1 | 1 | 3 | 0 | 0 | 0 |
| tomato | 0 | 0 | 2 | 0.18 | 0 | 0 |
| orange | 0 | 0 | 2 | 0.18 | 0 | 0 |
| potato | 1 | 1 | 2 | 0.18 | 0.18 | 0.26 |
| peach | 1 | 1 | 1 | 0.48 | 0.48 | 0.68 |
| lemon | 1 | 1 | 1 | 0.48 | 0.48 | 0.68 |

apple: 0 + 0 + 0 = 0

tomato: 0.71 + 0.52 + 0 = 1.23

orange: 0.71 + 0.52 + 0 = 1.23

potato: 0+0.68 + 0.26 = 0.94

peach: 0 + 0 + 0.68 = 0.68

lemon: 0 + 0 + 0.68 = 0.68

so the rank is:

potato peach lemon tomato orange apple

2.

|  |  |  |  |
| --- | --- | --- | --- |
| term | **document 1** | | |
| tf | wf | n’lized |
| apple | 3 | 1.48 | 0.72 |
| tomato | 1 | 1 | 0.49 |
| orange | 1 | 1 | 0.49 |
| potato | 0 | 0 | 0 |
| peach | 0 | 0 | 0 |
| lemon | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| term | **document 2** | | |
| tf | wf | n’lized |
| apple | 1 | 1 | 0.46 |
| tomato | 1 | 1 | 0.46 |
| orange | 1 | 1 | 0.46 |
| potato | 2 | 1.3 | 0.60 |
| peach | 0 | 0 | 0 |
| lemon | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| term | **document 3** | | |
| tf | wf | n’lized |
| apple | 1 | 1 | 0.5 |
| tomato | 0 | 0 | 0 |
| orange | 0 | 0 | 0 |
| potato | 1 | 1 | 0.5 |
| peach | 1 | 1 | 0.5 |
| lemon | 1 | 1 | 0.5 |

(1)

= (0.72, 0.49, 0.49, 0, 0, 0)

= (0.46, 0.46, 0.46,0.60,0,0)

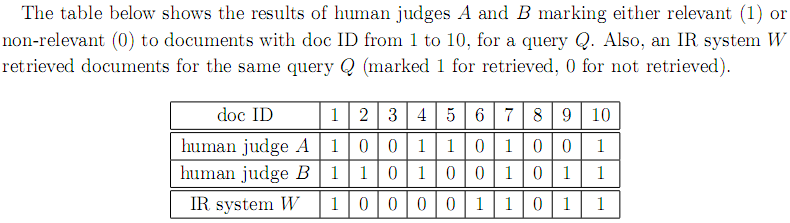
= (0.5, 0, 0, 0.5, 0.5, 0.5)

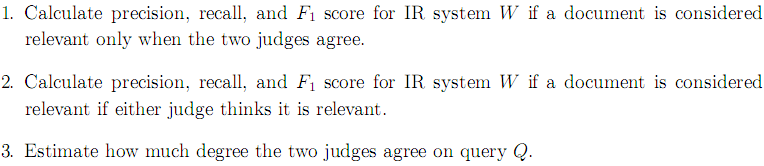
(2)

Most similar pair: d1 and d2.

## I-9

### Description





### Solution

|  |  |  |  |
| --- | --- | --- | --- |
|  | relevant | not relevant |  |
| retrieved | 3 | 2 | 5 |
| not retrieved | 1 | 4 | 5 |
|  | 4 | 6 |  |

P = 3 / 5 = 0.6

R = 3 / 4 = 0.75

F1 = 2 \* ( 1 / ( 1/0.6 + 1/0.75 ) ) = 0.67

|  |  |  |  |
| --- | --- | --- | --- |
|  | relevant | not relevant |  |
| retrieved | 4 | 1 | 5 |
| not retrieved | 3 | 2 | 5 |
|  | 7 | 3 |  |

P = 4 / 5 = 0.8

R = 4 / 7 = 0.57

F1 = 2 \* ( 1 / ( 1/0.8 + 1/0.57 ) ) = 0.67

3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B | A | | | |
|  | yes | no | total |
| yes | 4 | 2 | 6 |
| no | 1 | 3 | 4 |
| total | 5 | 5 | 10 |

P(A) = (4+3)/10 = 0.7

P(E) =

= +

= 0.2025 +0.3025

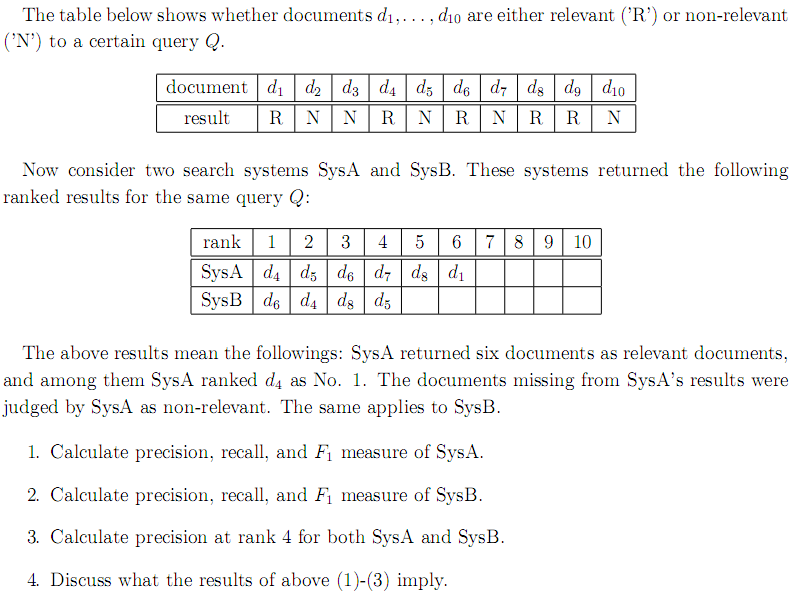
=0.505

K=

K<0.67, so, in this case, maybe the data providing a dubious basis for an evaluation.

## I-10

### Description



### Solution

1.

|  |  |  |
| --- | --- | --- |
|  | relevant | not relevant |
| retrieved | 4 | 2 |
| not retrieved | 1 | 3 |

P=tp/(tp+fp)=4/(4+2)=2/3

R=tp/(tp+fn)=4/5

F1=2PR/(P+R)=8/11

2.

P=3/4

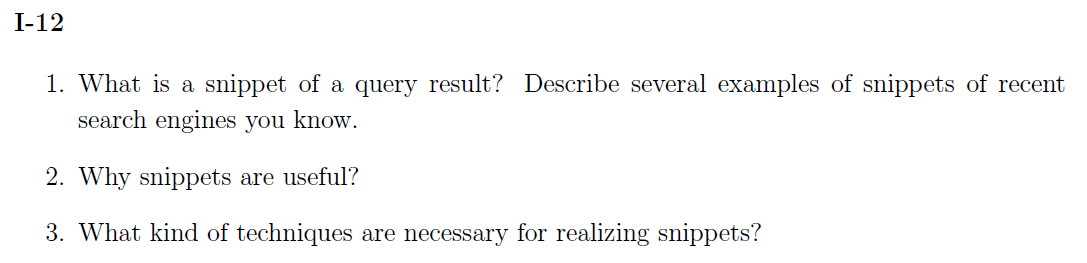
R=3/5

F1=2/3

3.

4.

As above, to system A, F1= 8/11, and to system B, F1=2/3. 8/11 is bigger than 2/3. So, in the case of overall performance, system A is better than system B. However, to system A, P@4=1/2, and to system B, P@4=3/4. So, in the case of “Precision at k”, system B is better than system A.



1.

The snippet, a short summary of the document, is designed so as to allow the user to decide its relevance. Typically, the snippet consists of the document title and a short summary, which is automatically extracted.

2.

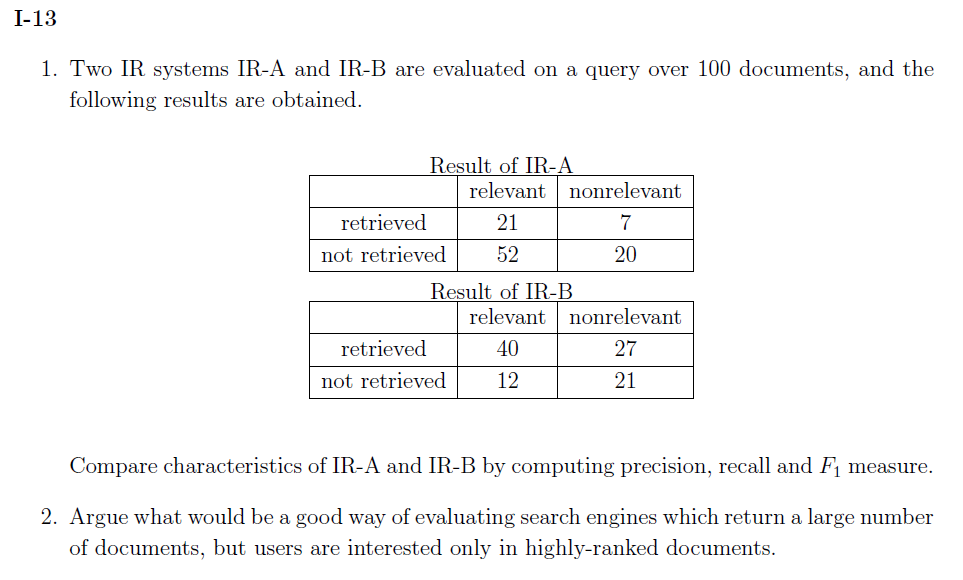
Having chosen or ranked the documents matching a query, we wish to present a results list that will be informative to the user. In many cases the user will not want to examine all the returned documents and so we want to make the results list informative enough that the user can do a final ranking of the documents for themselves based on relevance to their information need. The standard way of doing this is to provide a snippet.

3.

If we want to realizing snippets, the question is how to design the summary so as to maximize its usefulness to the user. The two basic kinds of summaries are static, which are always the same regardless of the query, and dynamic (or query-dependent), which are customized according to the user’s information need as deduced from a query. Dynamic summaries attempt to explain why a particular document was retrieved for the query at hand.

A static summary is generally comprised of either or both a subset of the document and metadata associated with the document. Besides, there has been extensive work within natural language processing (NLP) TEXT SUMMARIZATION on better ways to do text summarization.

Dynamic summaries display one or more “windows” on the document, aiming to present the pieces that have the most utility to the user in evaluating the document with respect to their information need.



1.

A:

P = 21/28

R = 21/73

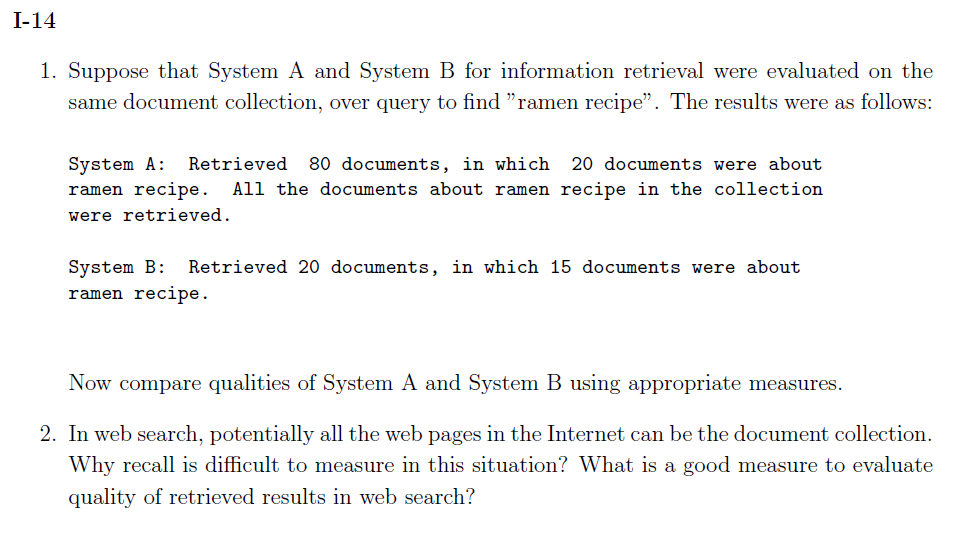
F1= 2PR/(P+R)

B:

```````

2.

Intuitively speaking, certainly, the p@K (“Precision at k”) is a good way satisfied the situation. Besides, an alternative is R-precision. R-precision adjusts for the size of the set of relevant documents: A perfect system could score 1 on this metric for each query, whereas, even a perfect system could only achieve a precision at 20 of 0.4 if there were only 8 documents in the collection relevant to information need.



1.

To system A,

P = 20/80 =0.25

R = 20/20 = 1

F1 = 2PR/(P+R) = 2\*0.25\*1/(0.25+1)= 0.4

To system B,

P = 15/20 = 0.75

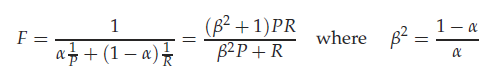
R = 15/20 = 0.75

F1 = 2PR/(P+R) = 2\*0.75\*0.75/(0.75+0.75) = 0.75

So, as above results, system B is better than system A.

2.

In web search, as the question describe, potentially all the web pages in the Internet can be the document collection. So the information retrieved is huge. In this case, maybe we can get a recall of 1 by retrieving all documents for all queries, but very low precision. Thus, recall is difficult to measure in this situation. *F* measure is a good measure to evaluate quality of retrieved results in web search. It is a single measure that trades off precision versus recall, which is the weighted harmonic mean of precision and recall:



where ∈ [0, 1] and thus ∈ [0,].