Test Exam for IR

Problem 1.
A search engine has returned the following list in response to a query:

RNRRN NNRRR

There are 20 relevant documents in the collection for that query.

Calculate:

- Precision at cut-off 8
- Interpolated precision at 15.11% recall
- Avg. Prec.
- R-Prec.
- Draw the interpolated precision-recall curve

Problem 2.
Prof. A has proposed the following measure for queries for which there is only one relevant document in the collection:

MRR = 1 / k, where k is the rank where the relevant document is returned.
If the system did not return the relevant document then MRR = 0.

Prof. B insists that Average Precision (AP) should be used instead. Compare MRR with AP by reporting if MRR is always higher than AP, or MRR is always less than AP, or sometimes MRR is lower and sometimes higher.

Problem 3.
a)
Compress the string **the** using Lempel-Ziv. You need to use that the ASCII code of
t = 01110100, h = 01101000, e = 01100101

b)
Decompress the string and use an ASCII table to decode the resulting binary string.
00101010101000100110100100101100101

c)
Let **lz(x)** be the string that is output by the Lempel-Ziv algorithm when the binary string x is given as input. Let **len(y)** denote the length of some binary string y in bits. Let expand(x) denote the following operation on the string x: each 0 is replaced by 00 and each 1 is replaced by 11.
For example:

expand(010011) = 001100001111

c1)
Is is true that

2 * len(lz(x)) >= len(lz(expand(x))) for all (non-empty) strings x

c2) Which is the smallest (non-empty) string x with the property that x = lz(x)

Problem 4.
A source emits symbols independently according to the following distribution:

P(a) = 0.1
P(b) = 0.1
P(c) = 0.2
P(d) = 0.3
P(e) = 0.3

a) Encode each symbol naively the using the same number of bits for each symbol.
b) Encode each symbol according to the Huffman code
c) Compute the entropy of the source.
d) For a string of n characters compute and compare the average number of bits used under naïve encoding, Huffman encoding, the lower bound given by the entropy. The average number of bits for Huffman is obtained by averaging over all strings of length n.

Hint for d)
Average number of bits used in a string of length n under Huffman encoding =
n * avg_number_bits_per_symbol_for_huffman, where

avg_number_bits_per_symbol_for_huffman =
P(a) * length_code_word(a) + P(b) * length_code_word(b) + ... + P(e) * length_code_word(e),

length_code_word(a) is the number of bits in the code word assigned to the letter a, by the Huffman code.

Problem 5)
What is the purpose of stemming and stop wording? [Short answer]

Problem 6)
a. What is clustering? [Short definition and explanation]

b. What methods for collaborative filtering do you know?

c. Describe the major techniques that spammers use in the web to increase the scores of their web pages.

Problem 7)
This problem deals with the following scenario**. n** scientific papers were downloaded (n = 10 000) from the web. For each paper x, the number of papers that reference that paper were computed and that number is denoted by ref-count(x). (A paper x references paper y if x contains a citation or link to y.) Then the papers were sorted according to the ref-count in decreasing order,i.e. ref-count(1) is the count of the most referenced paper, and ref-count(n) is the count of the least referenced paper. We also know that the number of all references is N = 100 000. We will also assume that each paper is referenced at least once.
Assume that Zipf's law holds for ref-counts:
r * ref-count(r) = A * N, where A = 0.1 and N = 100 000 = total number of references

Compute:

a) How many papers are referenced between 20 and 21 times?

b) Which is the smallest r for which ref-count(r) = 11?

c) Suppose I am to draw the collection of papers as a graph, where the papers are the nodes and I put an edge from node x to node y if paper x references paper y. Then how many edges and how many nodes will I need to draw?

d) (Optional:) What is the connection between n, N and A?

Problem 8)

I have run a query on three search engines SE1, SE2 and SE3.

SE1 returned the following ranked list of document ids: a, c, d, b

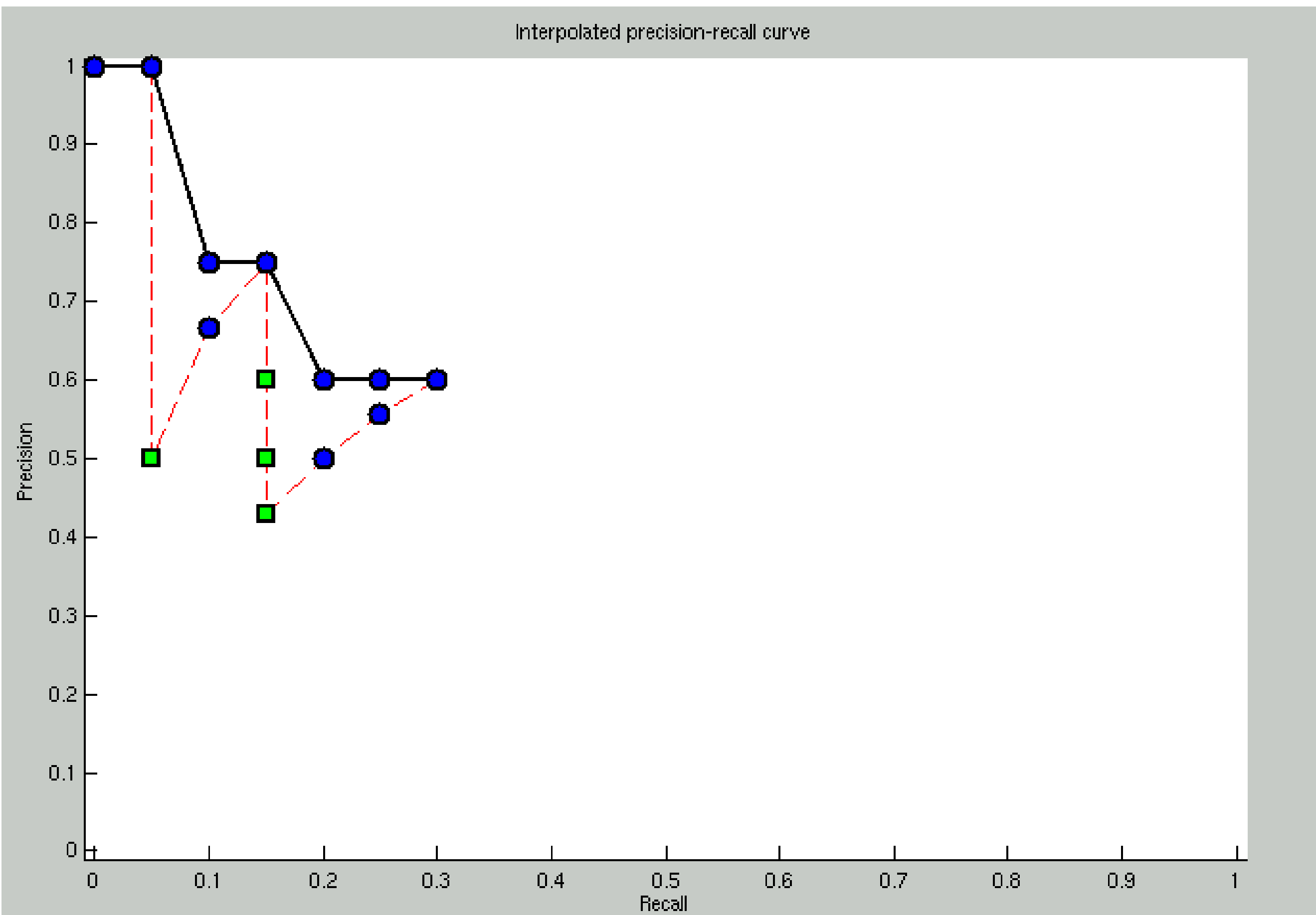
SE2 returned: c, a, d, b

SE3 returned: a, b, c, d

8.1) Metasearch using the Borda algorithm was run on those ranked list. Show the resulting list.

8.2) Same for Condorset algorithm.

Solution to Problem 1:



Solution to Problem 3:

Encoding of the

Step 1: Convert each letter to its ASCII code

01110100 01101000 01100101

Step 2: Parse the binary string into unique substrings
0, 1, 11, 01, 00, 011, 010, 000, 110, 0101,

Step 3: Replace each substring by its (back-ref, suffix-bit) pair
Note: if a substring is of length 1, by definition we will write 0 for back-ref
(0, 0), (0, 1), (1, 1), (3, 1), (4, 0), (2, 1), (3, 0), (3, 0), (6, 0), (3, 1),

Step 4: encode each back-ref by its binary representation
Note: the length of the bit string used to encode the back-ref depends on the consecutive number of the bit string, see below
back-ref 0 is at position 0 and therefore will be encoded by 0 bits
0 ==>
back-ref 0 is at position 1 and therefore will be encoded by 1 bits
0 ==> 0
back-ref 1 is at position 2 and therefore will be encoded by 2 bits
1 ==> 01
back-ref 3 is at position 3 and therefore will be encoded by 2 bits
3 ==> 11
back-ref 4 is at position 4 and therefore will be encoded by 3 bits
4 ==> 100
back-ref 2 is at position 5 and therefore will be encoded by 3 bits
2 ==> 010
back-ref 3 is at position 6 and therefore will be encoded by 3 bits
3 ==> 011
back-ref 3 is at position 7 and therefore will be encoded by 3 bits
3 ==> 011
back-ref 6 is at position 8 and therefore will be encoded by 4 bits
6 ==> 0110
back-ref 3 is at position 9 and therefore will be encoded by 4 bits
3 ==> 0011

Summary of Step 4:

(_, 0), (0, 1), (01, 1), (11, 1), (100, 0), (010, 1), (011, 0), (011, 0), (0110, 0), (0011, 1),

Result:
00101111110000101011001100110000111

Decode: 00101010101000100110100100101100101

Step 1
Split input string into blocks where:
the size of block 1 is 1 bit(s)
the size of block 2 is 2 bit(s)
the size of block 3 is 3 bit(s)
the size of block 4 is 3 bit(s)

the size of block 5 is 4 bit(s)
the size of block 6 is 4 bit(s)
the size of block 7 is 4 bit(s)
the size of block 8 is 4 bit(s)
the size of block 9 is 5 bit(s)
the size of block 10 is 5 bit(s)
0, 01, 010, 101, 0100, 0100, 1101, 0010, 01011, 00101,

Step 2
Split each block into two pieces: backref string and a suffix bit.
The last bit is the suffix bit, everything before is the backref string
Convert the backref string from binary to decimal
(0, 0), (0, 1), (1, 0), (2, 1), (2, 0), (2, 0), (6, 1), (1, 0), (5, 1), (2, 1),

Step 3
Process the list of pairs from left to right
If the first part of the pair is 0, simply write the suffix bit (which is the second part of the pair)
If the first part of the pair is n > 0, then find the string that was corresponds to the n-th pair before the current one
Consider pair (0, 0),
Simply print the suffix bit (0, 0) => 0
Consider pair (0, 1),
Simply print the suffix bit (0, 1) => 1
Consider pair (1, 0),
Obtain the bit string that appeared 1 steps before
It is 1
Append the suffix bit to obtain (1, 0) => 10
Consider pair (2, 1),
Obtain the bit string that appeared 2 steps before
It is 1
Append the suffix bit to obtain (2, 1) => 11
Consider pair (2, 0),
Obtain the bit string that appeared 2 steps before
It is 10
Append the suffix bit to obtain (2, 0) => 100
Consider pair (2, 0),
Obtain the bit string that appeared 2 steps before
It is 11
Append the suffix bit to obtain (2, 0) => 110
Consider pair (6, 1),
Obtain the bit string that appeared 6 steps before
It is 0
Append the suffix bit to obtain (6, 1) => 01
Consider pair (1, 0),
Obtain the bit string that appeared 1 steps before
It is 01
Append the suffix bit to obtain (1, 0) => 010
Consider pair (5, 1),
Obtain the bit string that appeared 5 steps before

It is 11
Append the suffix bit to obtain (5, 1) => 111
Consider pair (2, 1),
Obtain the bit string that appeared 2 steps before
It is 010
Append the suffix bit to obtain (2, 1) => 0101
Step 4: Merge blocks into a bit string
011011100110010101110101
Step 5: split the bit string into groups, each group being 8 bits; convert each group to its ASCII code
01101110 = 110 ==> n
01100101 = 101 ==> e
01110101 = 117 ==> u

Result: neu

Solution to problem 5 from homework 3:

a)

We want P(query | model for document A).

First we compute the probabilities for each word

P(cat | model for A) = term-freq("cat")/ document_length(doc. A) = 2/3
P(food| model for A) = term-freq("food") /document_length(doc. A) = 1/3
P(fancy| model for A) = term-freq("fancy") /document_length(doc. A) = 0/3

Notice that probabilities sum to 1: 1/3 + 2/3 + 0/3 = 3/3 = 1

Next, we compute the likelihood of the query given the model for document A.

The query is "cat cat cat food food food food fancy"

P(query| model for A) = P("cat cat cat food food food food fancy" | model for A) =
(we assume query words are generated independently according to the probablity model for document A, therefore we multiply probabilities)
= P(cat| model for A) * P(cat | model for A) * P(cat | model for A) *
P(food | model for A) * P(food | model for A) *P(food | model for A) * P(food | model for A) *
P(fancy | model for A)

= P(cat | model for A)^3 P(food | model for A)^4 P(fancy| model for A)^1 =
= (2/3)^3 * (1/3)^4 * (0/3)^1 = 0

For document B, we repeat the same procedure:

P(cat | model for B) = 1/5
P(food|model for B) = 3/5
P(fancy|model for B) = 1/5

P(query| model for B) = (1/5)^3 * (3/5)^4 * (1/5)^1 = 2.0736e-04

For document C:

P(cat | model for C) = 0/2
P(food|model for C) = 2/2
P(fancy|model for C) = 0/2

P(query| model for C) = (0/2)^3 * (2/2)^4 * (0/2)^1 = 0

b)

Here V = number of unique terms in the corpus = 3

P(cat | model for A) = [term-freq("cat") + 1]/ [document_length(doc. A) + V] = (2 + 1)/(3 + 3)
P(food| model for A) = [term-freq("food") + 1] /[document_length(doc. A) + V] = (1 + 1)/(3 + 3)
P(fancy| model for A) = [term-freq("fancy") + 1] /[document_length(doc. A) + V] = (0 + 1)/(3 + 3)

Again check probabilities sum to 1:

(2 + 1)/(3 + 3) + (1 + 1)/(3 + 3) + (0 + 1)/(3 + 3) = 1.0

P(query | model for A) = P(cat | model for A)^3 P(food | model for A)^4 P(fancy| model for A)^1 =
= ((2 + 1)/(3 + 3))^3 * ((1 + 1)/(3 + 3))^4 * ((0 + 1)/(3 + 3))^1 = 2.5720e-04

For B we have

P(query | model for B) = ((1 + 1) / (5 + 3))^3 * ((3 + 1)/(5 + 3))^4 * ((1 + 1)/(5 + 3))^1 =
2.4414e-04

Solution to Problem 7:
a)
Number of papers referenced between 20 and 21 times.

Number of papers referenced k times = n / [k * (k + 1)], where n is the number of all papers in the collection (n serves the same role as vocabulary size. It is number of unique referenced objects).

(See the formula from http://fiji4.ccs.neu.edu/stefan/IR/notes/zipfslaw.pdf)

For m = 20 and m = 21 we get

n / (20 * 21) + n / (21 * 22) = ...
(we know that n = 10 000 = number of papers in the collection)

b)
First approach: ref-count(r) = 11 and from Zipf's law r * ref-count(r) = A * N = 0.1 * 100 000
So, r * 11 = 0.1 * 100 000, which implies r = A*N/ 11 = ...
(N serves the same role as the number of total words in the collection)

Second approach: we use the definition of MaxRank from
http://fiji4.ccs.neu.edu/stefan/IR/notes/zipfslaw.pdf .
Reminder: MaxRank(k) = among all the objects with frequency k, this is the one with maximum rank.

So, what this question asks for is MaxRank(11 + 1) + 1. This means the following. MaxRank(12) is the rank of the last word that appears 12 times. Then the next word right after the last word that appears 12 times is the first word that appears 11 times.
MaxRank(12) = A * N / 12 [from the notes]

So, this approach gives A * N / 12 + 1 = ...

c) the graph will have n nodes and N edges.

d) Connection between, n, N, and A.
Apply Zipf's law to the least referenced paper. It's rank is n, since n is the number of papers in the collection. Zipf's law gives:
n * ref-count(n) = A * N.
We can assume ref-count(n) = 1 since least reference paper will be reference only once.
So, n * 1 = A*N, or 10 000 = 0.1 * 100 000

Topics for exam:

- General information about indexing, vector space model, boolean model, stop words, stemming, tf, idf, document length
  Reference: http://fiji4.ccs.neu.edu/stefan/IR/notes/notes2.pdf
- Precision, recall, Precision at cut-off, Precision at k% recall, Average Precision, R-precision, Interpolated Precision-Recall Curves
  Reference: http://fiji4.ccs.neu.edu/stefan/IR/notes/notes3.pdf
- Zipf's law
  Reference: http://fiji4.ccs.neu.edu/stefan/IR/notes/zipfslaw.pdf
- Language Models for search
  Reference: Problem 5 from homework 3, see solution above in this sheet
- Okapi Model. What is the purpose of different components in the formula
  Reference: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-356.pdf (only Part 1 and Part 2)
- Entropy, Huffman code, Average length of Huffman code, Lempel-Ziv, general idea why compression works
  Reference: http://fiji4.ccs.neu.edu/stefan/IR/notes/compression.pdf
  Problem 4 in this sheet and the hint about average length of Huffman code
- Metasearch, Borda, Condorset, Comb-sum algorithms
  Reference: http://fiji4.ccs.neu.edu/stefan/IR/lectures/datafusion.pdf
- General information about clustering. What is it? Where can it be used?
  Reference: The book ~~http://fiji4.ccs.neu.edu/stefan/IR/lectures/clustering.pdf~~
- General information about collaborative filtering. What is is? What kinds of methods exist to solve this problem?
  Reference: http://fiji4.ccs.neu.edu/stefan/IR/collaborarive.pdf
  http://en.wikipedia.org/wiki/Collaborative_filtering

- Page rank as a component of Web search:
  http://nlp.stanford.edu/IR-book/pdf/chapter21-linkanalysis.pdf
  http://research.microsoft.com/~najork/sigir2007.pdf

- ~~Spamming of web search engines~~
  ~~http://airweb.cse.lehigh.edu/2005/gyongyi.pdf~~
- Relevance feedback. What is the idea behind relevance feedback. Know some simple methods, e.g. how relevance feedback works in the vector space model.
  http://fiji4.ccs.neu.edu/stefan/IR/lectures/relevance_feedback.pdf