CS 276a Problem Set #1

Assigned: Tuesday, October 5, 2004

Due: Thursday, October 14, 2004 by 5:30 p.m.

Review session: Friday, October 8, 2004, 3:15-4:05 p.m. in Gates B01

Delivery: Local students should hand their solutions to Louis in class or leave them in the box by Professor Manning's office. SCPD students should use the SCPD courier system.

Late policy: Refer to the course webpage.

Honor code: Please review the collaboration and honor code policy on the course webpage. Also note: because some questions may be drawn from previous years' problem sets or exams, students are forbidden to consult solution sets from previous years unless we explicitly provide them.

7 questions and 70 points total.

#1. Integer coding (5 points)

Identify the set of all positive integers (show each integer using decimal, unary, and gamma coding) that:

a. have smaller length using unary coding than using gamma coding. (A correct specification of the set is sufficient for this part.)

b. have the same length using unary coding and using gamma coding. Prove that no integer outside your set has this property.

#2. Positional indexing (10 points)

Consider the following fragment of a positional index with the format
word: document: position, position … document: position …

Gates:	1:3	2:6	3:2,17	4:1	
IBM:	4:3	7:14			
Microsoft:	1:1		2:1,21	3:3	5:16,22,51

The NEAR/x operator in text retrieval has the following syntax: *Word1* NEAR/x

Word2 finds occurrences of *Word1* within x words of Word2, where x is a positive integer argument. Thus x = 1 demands that Word1 be adjacent to Word2 (but not necessarily in that order).

a. Describe the set of documents that satisfy the query **Gates** NEAR/2 **Microsof**t.

b. Describe each set of values for which the query **Gates** NEAR/x **Microsoft** has a different set of documents as answers.

c. Consider the general procedure for "merging" two positional postings for a given document, to determine where the document satisfies a NEAR/x clause (since in general there will be many positions at which each term occurs in a document). We begin with a pointer to the position of occurrence of each term and move each pointer along the list of occurrences in the document, checking as we do so whether we have a hit for NEAR/x. Each move of either pointer counts as a step. Let L denote the total number of occurrences of the two terms in the document. What is the big-O complexity of the merge procedure, if we wish to have postings for the result?

d. Suppose we wish to use the general "merging" procedure to determine whether the document satisfies a NEAR/x clause. Which of the following is true? Justify your answer.

i. The merge can be accomplished in a number of steps linear in L regardless of x, and we can ensure that each pointer moves only to the right.

ii. The merge can be accomplished in a number of steps linear in L, but a pointer may be forced to move non-monotonically (i.e., sometimes "back up").

iii. The merge can require xL steps in some cases.

#3. Postings lists (10 points)

We have a two-word query. For one term the postings list consists of the following 16 entries: [4,6,10,12,14,16,18,20,22,32,47,81,120,122,157,180].

For the other term it is the one-entry postings list [47].

a. Work out how many comparisons would be done to intersect the two postings lists with the following two strategies. Briefly justify your answers:

i) using standard postings lists

ii) using postings lists stored with skip pointers, with a skip length of sqrt(length), as recommended in class.

b. Explain briefly how skip pointers could be made to work if we wanted to make use of gamma-encoding on the gaps between successive docIDs.

#4. Index size computation (15 points)

Consider an index for 1 million documents each having a length of 1,000 words. Say there are 100K distinct terms in total. We do not wish to keep track of term-frequency information.

a. What is the space requirement for an uncompressed term-document incidence matrix that does not exploit sparsity?

b. Assuming a Zipf's Law distribution of terms, as in class, what is the space required for the postings in an inverted index that encodes each docID with fixed-length binary codes (without the use of gaps or gamma encoding)?

c. Under the same assumptions, but with gamma encoding of the gaps between successive docIDs in the postings lists?

#5. Dictionary storage (15 points)

Consider a lexicon (dictionary) in which there are no words of length 1 or 2. Assume the probability of a word in this lexicon having length i is proportional to $1/i^2$, for $i > 2$. Assume that the distribution is truncated so that the longest possible word length is 25. Further, the lexicon has 50,000 words in it.

a. Assume each character requires a byte of storage. Although inefficient, we could safely use 25 bytes per word for the dictionary, as in the first cut approach in class. How much storage is required for storing all the words in this fashion?

Now let us analyze a more efficient scheme.

b. What is the expected number of distinct words (types) of length i as a function of i?

c. From (b), infer the expected total number of characters to write down all words of length i as a function of i.

d. Now consider storing the words as one contiguous string, with a term pointer to the beginning of each word. How much space is used in total, including storage for the entire string as well as for the pointers that resolve the beginning of each word? Show a formula, and estimate the total as a number of bytes needed.

e. Next consider "blocking" the dictionary string so we point to the beginning of each eighth word rather than every word, storing the length of each word in one byte immediately preceding its appearance in the string. What is the total space used now? Show the formula, your estimates, and an estimated number of bytes.

#6. Stemming (5 points)

Which (if any) of the following pairs that are stemmed to the same form by the Porter stemmer definitely shouldn't be conflated? Explain your answers.

a. abandon/abandonment
b. absorbency/absorbent
c. marketing/markets
d. university/universe
e. volume/volumes

#7. More stemming (10 points)

The following is part of the beginning step of the stemming algorithm developed by Porter (1980). Porter's algorithm consists of a set of rules, in the form of S1 -> S2, which means that if a word ends with the suffix S1, then S1 is replaced by S2. If S2 is empty, then S1 is deleted:

IES -> I
SS -> SS
S ->

In a grouped set of rules written beneath each other (as above), only one is applied, and this will be the one with the longest matching S1 for the given word.

a. What is the purpose of including an identity rule such as SS -> SS?

Given the above set of rules, what do the following words map to?

b. PONIES
c. TIES
d. CARESS

What rules should you add to correctly stem the following?

e. CARESSES
f. PONY

g. The stemming for (b) (and hence (f)) given above might seem strange. Does it have a deleterious effect on retrieval?