CS160 - Homework 2 solutions

1. (10 points) Distributed indexing

   Figure 4.5 in the book shows an example MapReduce framework for distributed indexing.

   (a) Given n documents and m machines, describe a good method for splitting up the documents. Justify your answer.

   The main thing is that we want to spread out the load evenly among the machines. The easiest way to do this is to pick a moderate splite size, say 16-64MB, and split all of the data into such blocks. The size is large enough not to everwhelm the master with administrative work, but small enough to deal with imbalances in processing speed, etc.

   (b) In the reduce phase, the example suggests partitioning the data by $a - f$, $g - p$, and $q - z$. Is this a good approach? Explain your answer. Describe a better partition of 3 parts (or a better method of how to partition into 3 parts). Why is your partition better?

   As with part a, we'd like partitions that are roughly the same size. It's likely that the provided splits would not give this result since there are many fewer words (and frequent) words that start with $q-z$ than from the other data sets. Note that the amount of work is proportional not to the number of words in a dictionary that start with a given letter, but the number of occurrences of these words. We can take take a random sample of our data and calculate how many occurrences of words start with each letter in our corpus. We could then use either a greedy (assign each group the largest) or dynamic programming (if you wanted to get fancy) approach that creates groups that have similar frequency on our sample data. If there is too big a discrepency, then we could repeat the experiment, this time partitioning on the first 2 letters.

2. (10 points) Zipf's law

   From our data set from assignment 1, I counted the frequency of each word and sorted them by frequency. Below are a few data points:

Rank	Frequency
1	417,667
10	70,848
100	8,508
1000	842
10,000	37

   (a) Create a plot using these points like Figure 5.2 (on paper is fine, or you can use a program). Do the points seem to follow Zipf's law?

   In log-log space, the points are fairly close to linear, though not perfect. The points to tend to have a bit of a curve to it, but the fit is reasonable.

   (b) For all points ranked 10 or lower, estimate what the value should be using the point above it (for 10, it would be 1, for 100 it would be 10, etc.) and, percentage-wise, how far away the real values are from this estimate. Does Zipf's law seem like an appropriate model for the data?

rank	estimate	% off
10	41,767	+69%
100	7,085	+21%
1000	851	-1%
10000	84	-56%

   Although while some of the estimates are off by as much as 90% (or 40% depending on which way you compute the percentages), given the simplicity of the model, Zipf's law is still a reasonable approximation for the data for many applications. In addition, when considering approximations based on only one point (i.e. the previous point) we're bound to see variation. If we look at the data over multiple points (like part a), then the values are more reasonable.

3. (10 points) Book problem 5.5

   Feel free to use a decimal/binary converter on this problem and the next. To make my life easier, delimit the bytes with a space for the variable codes and delimit each entry in the gamma codes by a space

and use a ',' like done in the book within a code.

Gap lengths: [777,16966,276325, 30957268]
in binary: [1100001001, 100001001000110, 1000011011101100101, 11101100001011111011010100]

- variable:
  00000110 10001001
  00000001 00000100 11000110
  00010000 01101110 11100101
  00001110 01100001 00111101 11010100
- gamma
  1111111110,100001001
  111111111111110,00001001000110
  11111111111111110,000011011101100101
  1111111111111111111111110,11011000010111011010100

4. (10 points) Decoding

   For the codes below, show the gaps and the corresponding postings for the encoded strings.

   (a) variable code: 11010110 01011101 10111011 01101010 01110101 01101100 10000011

   gaps: [86, 11963, 224228867] docIDs: [86, 12049, 224240916]

   (b) gamma code: 11110001111011100111111101011011

   gaps: [19, 7, 2, 219] docIDs: [19, 26, 28, 247]