Problem Set 1

Due in class on Tuesday, September 26, 2006.

Problem 1: Automatic Cryptanalysis of the Caeser Cipher

The goal of this problem is to show the feasibility of fully automating a brute-force attack on the Caeser cipher. A brute force attack on a cryptosystem means trying all possible keys to see which one "unlocks" the encrypted message.

The difficulty in automating such an attack, aside from the time it takes to explore a potentially large key space, is knowing when you have succeeded. How can one distinguish the correct decryption from the wrong one? In general one cannot, but if some messages are more likely than others to be the decryption of a given ciphertext string c, then it is sensible to guess the message m' that is the most likely among the possiblities. The guess m' might or might not be equal to the original message m. If it is, we say the attack *succeeds*; otherwise it *fails*.

For this problem, we assume a fixed known probability distribution P on single letters, so for example, $P(\text{'A'})$ is the probability of choosing letter 'A'. Messages are of a fixed length r and are composed of letters chosen independently at random according to P. Thus, if $r = 3$,

$$\text{prob}[m = \text{'CAT'}] = P(\text{'C'}) \cdot P(\text{'A'}) \cdot P(\text{'T'}).$$

Given a ciphertext $c = E_k(m) = (k + m) \bmod 26$, the possible decryptions are $D_{k_0}(m)$, $\ldots$, $D_{k_{25}}(m)$ corresponding to each of the 26 possible keys. Each of these messages has a certain a priori probability as defined above. Your program should select as its guess the message (and corresponding key) with the highest probability, where ties are broken by choosing the message corresponding to the smaller key.

An *experiment* takes two inputs: A probability distribution P on single letters and a length r. The steps of conducting an experiment are:

(a) Choose a random message m of length r according to the distribution on length-r strings induced by P.

(b) Choose a key k uniformly at random from $\{0, \ldots, 25\}$.

(c) Compute $c = E_k(m)$, where E is the Caeser encryption function on length-r strings.

(d) Run the procedure cryptanalyze(P, r, c), where "cryptanalyze" is the analysis procedure described above to compute m' and k'.

(e) If $m' = m$, output "success" and the pair (m', k'). Otherwise, output "failure".

Finally, you should gather data by running a bunch of experiments. For each $r = 1, 2, \ldots$ up to some reasonable number, run a large number of experiments (say 100) and compute the fraction of successes. Repeat this several times to get a feeling for the variance in your results. Plot the results and find numbers r_1 and r_2 (if possible) such that the observed success rate is less than 10% for $r < r_1$, between 10% and 90% for $r_1 \le r \le r_2$, and greater than 90% for $r > r_2$.

You will be provided with one or more files describing probability distributions on which to test your code. Each such file will contain 26 whitespace-delimited integers, where the i^{th} integer gives the frequency of occurrence of the i^{th} letter in the alphabet. To convert these frequencies to probabilities, you will need to normalize by dividing each by the sum of the frequencies.

Beyond this description, you are free to organize your code any way you see fit. However, your code should produce enough intermediate output for you to be able to make a convincing case for its correctness.

When your program is working, you should gather data by running the experiments described above on the designated probability distribution. The results of your experiments should be presented in both tabular and graphical form.

You should submit your work using the submission script in the `/c/cs467/bin` course directory on the Zoo. Your submission should include the program or programs you have written, test runs showing the correctness of the various pieces, and the data tables and graphs resulting from your experiments. You should also submit written documentation describing in some detail what you have done.