Assignment 1- Part 4 - Report

Joseph May

Code description

Firstly the code declares the ciphertext itself as a string, it then records the positions of the spaces to use later. The spaces, commas and apostrophes are then removed to make the ciphertext easier to work with. A string is then declared with the order of the frequencies of the letters in the English alphabet i.e e being index 0 etc. A list of common English words was also declared. The decryption function was then declared. This function takes in a key and iterates through the ciphertext shifting each character with the corresponding letter of the key and adds the decrypted character onto a plaintext string. There is then a function to add the spaces back to the plaintext to make it easier to read, it uses the stored position of the spaces from earlier. Then the scoring algorithm is defined. It works as follows, it iterates through each letter in the plaintext, for each letter the score is 26 minus the index of the character in the frequency alphabet defined at the start e.g e is index 0 would add 26 points, z is index 25 so would add 1 point. It also used the re library to check if the plaintext contains any of the common words defined previously, 100 points is added to the score for each of them. For the main function, it iterates through key lengths 1-4 as we know that 4 is the maximum size, it then uses the product function imported from itertools and the ascii_uppercase library to generate tuples of every single combination of letters for each of the key sizes. The tuples are then made into strings using the join function and then the decryption, spaces and scoring functions are called that were described above. Each plaintext and score and key is appended to a list that stores tuples. After each key has been tried the list is sorted and the top scoring keys are displayed. The makefile for part 4 is called Part4Makefile and can be run by the command
make -f Part4Makefile. This creates an executable called v_cipher.

Attack method

My attack method for this assignment was to use brute force to try all possible key combinations for the cipher and to use a scoring algorithm to seek the top results. Since the key size is known to be no larger than 4 it is feasible to use a brute force attack. To simplify finding the actual plaintext and the key itself, I used a scoring algorithm on each of the plaintexts that gave points for letter frequency and also for common words. I then sorted the results by score which made it much easier to find the real plaintext and the key.

Alternative method

If the key length was not known then it would not be feasible to brute force the attack as there would be too many possible keys to try. Instead an analysis would have to be done to estimate the key length. This would be done with periodicity analysis. What this does is look for repeated sections in the ciphertext and estimates the key length based on the gaps between them. To implement this in the code instead of having a for loop of all possible keys, there would be a function that finds repeated strings in the ciphertext, it would iterate through the ciphertext storing all possible sequences of a certain length and store them in a dictionary along with their position in the text, then all of the sequences with only one occurrence would be removed. Then the spaces between the repeated sequences would be calculated and the GCD of all of the spaces between them would be returned as the estimated key length and that value would be used as they key length for all the possibilities to be tried.

Results

Key: JGYB, Plaintext: COURAGE IS FOUND IN UNLIKELY PLACES