Laboratory 1 - AISC

Breaking classical encryption schemes

Agnetta Stefano, Brozzo Doda Umberto, Macario Davide

March 15$^{\text{th}}$, 2023

A.Y. 2021-2022

1 Introduction

To be filled

2 Exercise 1 - Monoalphabetic cipher

The monoalphabetic cipher is an encryption algorithm based on the use of a key which is a random permutation of the alphabet that allows a one-to-one mapping of the characters of the plaintext to the ones of the encrypted message. To break the cipher we used the following strategy.

At first, we considered the first ten most common trigrams, digrams, and monograms. We report them below. Our first assumption was that the trigram "MWI", being the most common, was the encryption of the trigram "the". This assumption was supported by the fact that the diagrams "th" and "he" are the most common in the English language and the trigram "the" is composed of them, analogously in our encrypted message the most common trigram is "MWI" and the two most commons digrams were "MW" and "WI" which compose the trigram. Also in English "e" is the most common letter as "I" is in our encrypted message.

We then assumed that "YKS" and "XKN" – respectively third and second most common trigrams- should be respectively "and " and "ing". This is supported by the fact that the decrypted trigrams share the same middle letter as the encrypted trigrams do. Moreover "in" is one of the most common diagrams and it is the beginning of "ing", analogously "XK" is among the most common digrams in the encrypted message.

The sixth most common trigram is "IPI" in our encrypted message. Our assumptions show that its partial decryption is "ePe". Among the most common trigrams, the only one to match this structure is "ere" so we assumed that "P" encrypts "r".

The tenth most common diagram is "ID", whose partial decryption with our assumption is "eD". The most commons diagram starting with "e" are: "es", "en", "ed", and "ea". Among these, the only one whose second letter has not already a match in our assumption is "es", so we assume that "s" encryption is "D".

At this point, we started to analyze the partially decrypted text and we started making

assumptions based on the fact that the text should have made sense in English. Below we report a list of parts of the partially encrypted message and our assumption on the missing letter.

- straightenhisCaGJandnarrLHhisAittAeCAFeeEes = straighten his back and narrow his little blue eyes;
- EesoRcourseiRitsRinetoQorrowsaid = yes of course if it is fine tomorrow said;
- needsaboUeall = needs above all;
- aneZtraordinaryBoyasifi = an extraordinary joy as if;

To automate this function we should write a program that initially evaluates the most common trigrams and tries to match them with the most common trigrams in the English language enforcing some constraints (e.g. considering the second and third most common trigrams if and only if their middle letter is the same as in English the second and third most common trigrams are "ing" and "and"). This would be a good way to start automating the process but we cannot be completely sure that these matches will be the right ones, but we can assume there's a high probability.

In order to continue the decryption, the table of frequency provided would not be enough. It would be useful to provide the program with some words that are more likely to be present in the plaintext. By doing this, after computing a partial decryption based on the matches found using the most common groups of letters, the program should try to match those words to parts of the partially encrypted message by passing them as a filter on the message and finding places where that word could fit. For example, in the message analyzed, using the word "above" and the partially decrypted part of the message "needsaboUeall", the program should find a match between "U" and "v". In general, we should refuse the match every time one of the decrypted letters does not match the ones in the word or if any decrypted letter in the considered part of the message would match the corresponding letters in the word but the remaining letters has already be decrypted (e.g. matching the word "better" to "bYtter" should not be allowed as the letter "e" has already been decrypted).

3 Exercise 2 - Vigenère cipher

The Vigenère is an encryption scheme based on the use of a key k of length m to encrypt the plaintext p as follows:

$$c_i = E(p,k) = (p_i + k_{i \mod m}) \mod 26 \quad (1)$$

This means that letters at distance m are shifted by the same amount, as in a generic Caesar's cipher. As a consequence, a possible approach for cryptanalysis can be that of looking at the ciphertext as a set of subsequences obtained by considering letters at distance m of the cipher. Then, it is possible to act on these subsequences by analyzing letter frequencies. In this case, a brute force approach would be time consuming as the subsequences do not contain english words, but rather (pseudo-)random letters. In any case the frequency analysis is feasible, since these letters are extracted from an (encrypted) English text.

The cracking mechanism is based on evaluating the circular correlation between the letter frequencies in each subsequence and the theoretical vector of letter densities for the English language. By finding the maximum of the correlation it is possible to choose the key character for the current subsequence, as this character will correspond to the shift value which makes the letter frequencies of the subsequence match with the theoretical frequencies. This, however requires to estimate the length of the key, and to achieve this another previous step is needed.

The estimate of the key length is done by extracting information from the ciphertext, once again revealing the main disadvantage of this encryption scheme. Indeed, having selected a possible key length (m) and extracted the subsequences of letters having the same distance (m sequences), it is possible to evaluate a score on each set, corresponding to the sum of squared frequencies of each letter in different subsequences. Each language has a specific value for this score, which, in the case of English is 0.065.

In this case, the overall score was evaluated as an average of the individual scores of each subsequence.

It is important to notice, however, that since the key length is not always a divisor of the total number of characters in the ciphertext, for some values of m, the generated subsequences may not be of the same length. As a consequence, by evaluating the scores and averaging them, the shorter sequences will introduce more noise than the longer ones. To solve this issue, which causes bad (biased) estimations of the score, it is necessary to truncate longer sequences.

By comparing the scores as functions of m, it is possible to find out the best value for the key length as the one for which the score is closest to 0.065.

In the case of the ciphertext found in document 'cryptogram02.txt', the estimated key length was 15 and the key was found to be "nowyouseethekey".

3.1 Complexity of the algorithm

In order to analyze how complex it is to break the Vigenère cipher, it is possible to evaluate the complexity of the whole procedure as a function of the (maximum) key length.

To be able to extract good statistical information from the subsequences to evaluate the letter densities used in the cracking procedure, a good compromise is that to force the subsequences to contain at least 100 characters. This means that the maximum key length which is tried is:

$$m_{max} = \left\lfloor \frac{len(c)}{100} \right\rfloor \tag{2}$$

The algorithm used to break the cipher was composed of two parts: first the key length estimation, then the actual key evaluation.

The second part only marginally depends on the key length. Indeed, the algorithm consists in extracting the subsequences (which is a linear operation in the length of the key) and on evaluating the letter frequencies for each, which has a cost depending on the total number of letters in the ciphertext and on the alphabet, hence independent on the key length. Last, what is done is the circular correlation, which again has a complexity only depending on the number of letters of the alphabet (the correlated sequences have as many elements as the alphabet). Lastly, also the search for the maximum is linear in the number of alphabet letters. The computational cost of this operation with respect to the maximum key length is therefore $O(n)$.

The estimation of the key length instead is quadratic in the maximum key length (in our implementation). The steps require an iteration on all the possible key lengths and, for each value of m, the extraction of the subsequences and the evaluation of the score on each, before carrying out the average and possibly selecting a new best value of m. As a consequence, the complexity is equivalent to performing a constant number of operations within two nested for loops having a number of iterations proportional to the maximum key length.

3.2 Comparison with monoalphabetic cipher

While in the monoalphabetic cipher each letter was mapped to another arbitrary letter, without a rotation of the alphabet as in the (generic) Caesar cipher, the subsequences in Vigenère's cipher are in fact examples of Caesar ciphers, as the letters of each subset have been "rotated" by a constant amount corresponding to the associated character in the cryptographic key.

As a consequence, to break the Vigenère cipher it is possible to perform an effective frequency analysis on the single subsequences, in the same way as for breaking a Caesar cipher, and this operation has to simply be repeated by a number of times equal to the key length.

In the monoalphabetic cipher, instead, the absence of a constant shift makes it more tricky to come up with a decryption as a lot of trial and error is needed and it is more difficult to find a systematic approach.

4 Bonus question (1)

By feeding the text file 'cryptogram03.txt' to the program developed for breaking the Vigenère cipher, which was intentionally written in the most general form possible, it has been observed that the ciphertext was indeed a result of that encryption technique, using the 21-letter key: "thiskeyisreallysecure".

5 Breaking stream cipher

The stream cipher can be described as a 'one-time pad' in which the random key has been replaced by a pseudorandom one. In the same way as the one-time pad, the stream cipher is secure (computationally, in this case) provided that the key is not reused for multiple messages.

Indeed, an attacker who can observe multiple messages encrypted with the same key finds himself in the same condition of an eavesdropper trying to crack a Vigenère cipher. By appending all the different messages into a single one, one can notice that the created ciphertext is composed of different sections encrypted by the same key.

The approach proposed in this laboratory can be a viable solution to break this cipher in the case of multiple messages using the same key.

6 Conclusions