# Programmazione DataSecurity

Jacopo Manetti

April 2023

## 1 Analisi delle frequenze di un testo

Il codice Python di questo esercizio definisce tre funzioni per l'analisi statistica di un testo passato come argomento.

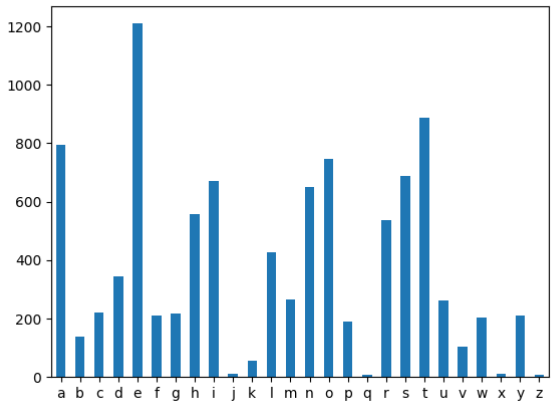
La prima funzione, denominata **count_letter**, conta le occorrenze di ogni carattere del testo, ne calcola l'istogramma e lo stampa a video. Il testo viene prima elaborato dalla funzione **set_text**, che lo rende tutto in minuscolo e rimuove alcuni caratteri speciali.

La seconda funzione, denominata **count_mgrams**, calcola gli m-grammi del testo passato come argomento e ne conta le occorrenze. Gli m-grammi sono sequenze di m caratteri, ad esempio i bigrammi sono sequenze di due caratteri. La funzione normalizza le frequenze degli m-grammi e le stampa a video.

La terza funzione, denominata **calc_ic_entropy**, calcola l'indice di coincidenza e l'entropia degli m-grammi del testo passato come argomento. L'indice di coincidenza è una misura di quanto il testo sia cifrato, mentre l'entropia è una misura dell'incertezza degli m-grammi. Anche in questo caso, la funzione normalizza le frequenze degli m-grammi e le stampa a video.

Il codice viene testato sul primo capitolo di Moby Dick (H. Melville, 1851).

1. L'istogramma della frequenza delle 26 lettere risulta essere:



2. La distribuzione empirica degli m-grammi rispecchia la distribuzione della lingua inglese per m=1, aumentando poi la dimensione del blocco per m=2,3,4 si hanno molte più combinazioni che però occorrono meno volte.

3. Gli indici di coincidenza e l' entropia usando il primo capitolo di moby dick sono:

   m=1

   Index of coincidence: 0.0660

   Entropy: 4.1630

   m=2

   Index of coincidence: 0.0077

   Entropy: 7.6968

   m=3

   Index of coincidence: 0.0015

   Entropy: 10.4935

   m=4

   Index of coincidence: 0.0003

   Entropy: 12.0438

   Dai valori degli indici di coincidenza e dell'entropia ottenuti, si può notare che all'aumentare della lunghezza degli n-grammi, l'entropia del testo aumenta e l'indice di coincidenza diminuisce. Questo significa che la probabilità di trovare ripetizioni di sequenze di caratteri diventa sempre più bassa man mano che la lunghezza degli n-grammi aumenta. Inoltre, l'entropia alta indica una maggiore diversità e casualità nel testo. L'indice di coincidenza invece è massimo quando il testo presenta molte ripetizioni di sequenze di caratteri, il che indica una maggiore regolarità e struttura nel testo. In questo caso, i valori degli indici di coincidenza e dell'entropia per m=1 sono tipici di un testo in lingua inglese , mentre i valori per m¿1 indicano una maggiore complessità e casualità nel testo.

## 1.1 Cifrario di Hill

Il codice Python di questo esercizio implementa un cifrario di Hill, un cifrario a blocchi che utilizza una matrice per cifrare un blocco di testo in chiaro. Il cifrario supporta la cifratura e la decifratura di testo e fornisce anche una funzione di attacco per cercare di determinare la chiave di cifratura data una coppia di testo in chiaro e testo cifrato.

Il codice utilizza la libreria NumPy per la gestione delle matrici e la libreria SymPy per la verifica della invertibilità delle matrici. In particolare, la funzione **"get_inverse"** controlla se la matrice passata come parametro è invertibile modulo 26 e, in caso affermativo, restituisce la sua inversa modulo 26.

La funzione **"convert_to_numbers"** converte una stringa di testo in una lista di numeri interi che rappresentano i caratteri del testo, utilizzando l'alfabeto inglese e la riduzione modulo 26.

La funzione **"encryption"** cifra un blocco di testo in chiaro utilizzando la chiave di cifratura fornita come matrice. La funzione divide il testo in blocchi di dimensione "m" e cifra ogni blocco utilizzando la matrice chiave, poi converte i numeri cifrati in caratteri del testo.

La funzione **"decryption"** decifra il testo cifrato utilizzando la chiave di decifratura fornita come matrice. La funzione divide il testo cifrato in blocchi di dimensione "m" e decifra ogni blocco utilizzando la matrice inversa della chiave, poi converte i numeri decifrati in caratteri del testo.

La funzione **"group_array"** prende una lista di numeri interi e restituisce una matrice "m x len(arr)/m" corrispondente. La funzione viene utilizzata per suddividere la lista di numeri interi in blocchi di dimensione "m" e creare la matrice corrispondente.

La funzione **"calculate_key"** calcola la chiave di cifratura utilizzando la matrice di testo in chiaro e la matrice di testo cifrato fornite come input.

La funzione **"extract_square_matrices"** estrae tutte le sottomatrici quadrate di dimensione "m" da una matrice di dimensione maggiore "m x n". La funzione viene utilizzata dalla funzione di attacco per estrarre tutte le possibili sottomatrici quadrate da una coppia di testo in chiaro e testo cifrato. La funzione restituisce una lista di tutte le sottomatrici quadrate estratte.

La funzione **"attack_hill"** implementa un attacco al cifrario Hill.Questa funzione richiede in input una coppia di testo in chiaro e testo cifrato, oltre alla dimensione "m" dei blocchi di testo utilizzati nel cifrario Hill.

L'algoritmo estrae tutte le possibili sottomatrici quadrate dalle matrici di testo in chiaro e testo cifrato, quindi verifica se ogni sottomatrice è invertibile modulo 26. Se lo è, l'algoritmo calcola la chiave di cifratura utilizzando le matrici di testo in chiaro e testo cifrato.

Se l'algoritmo riesce a trovare una chiave di cifratura corretta, restituisce la chiave. Se non trova nessuna chiave corretta, l'algoritmo restituisce un messaggio di errore.

**N.B** Per testare il codice è importante ricordarsi di scegliere come chiave k una matrice invertibile mod 26