Criptografía: Aspectos Teóricos y Prácticos — Laboratorio 1: Criptoanálisis

En este laboratorio implementarán algunas herramientas para hacer criptoanálisis, y desencriptarán varios textos cifrados. Se entregará el **martes 22 de abril**. Quiero que

1. me pasen por mail un archivo llamado laboratorio1.py (o laboratorio.sage) que contenga todas las funciones indicadas en los ejercicios.
2. compartan conmigo un notebook de Sage en el servidor de CMAT para que pueda ver cómo rompieron el texto encriptado por los cifrados de sustitución y de Vigenère. Mi usuario es `nryan`.

Prefiero que trabajen en equipos de dos alumnos.

1. **Preprocesando textos.** Bajar el archivo que contiene el texto de *Crónica de una muerte anunciada* y convertirlo en una cadena de letras mayúsculas, sin símbolos de puntuación y sin espacios. Para evitar el tema de UTF-8, el texto está en ASCII (o sea, cambié todos los á, é, í, ó, ú, ü, y ñ, por, respectivamente, a, e, i, o, u ,u y n). La implementación deberá incluir una función con esta signatura:

   ```
   def preprocesar(nombre_archivo):
   ```

   La variable `nombre_archivo` es una cadena tipo `''XXX.txt''` y el resultado de llamar a cada función deberá ser un archivo nuevo con el mismo nombre pero con la cadena `''_preprocesada''` antes del `''.txt''`: tendrá la forma `''XXX_preprocesada.txt''`.

2. **Análisis de frecuencia.** Implementar una función que recibe un texto preprocesado y calcula una tabla de frecuencias para cada letra de la 'A' a 'Z'. La función deberá tener la signatura:

   ```
   def frecuencias(nombre_archivo):
   ```

   y deberá devolver un diccionario cuyos índices son las letras y cuyos valores son las frecuencias.

3. **Índice de coincidencia.** Hacemos una definición: Sea $s = c_0c_1c_2\dots c_n$ una cadena de letras mayúsculas. El *índice de coincidencia* de s es la probabilidad que dos carácteres elegidos aleatoriamente de s sean iguales. Implementar una función que recibe un texto preprocesado y calcula el índice de coincidencia del texto. La función deberá tener la signatura:

   ```
   def inco(nombre_archivo):
   ```

   y deberá devolver un número flotante, el índice de coincidencia.

4. **Método de Kasiski.** Un *trigrama* es una sucesión de tres letras seguidas en un texto. Implementar una función que halla todos los trigramas que occurren en un texto preprocesado y la distancia entre trigramas repetidos. Las función deberá tener la signatura:

   ```
   def kasiski(nombre_archivo):
   ```

   y deberá devolver un diccionario cuyos índices son trigramas y cuyos valores son una lista de distancias (factorizadas) entre las occurrencias del trigrama que corresponde al índice.

5. **Cifrado de Vigenère.** El cifrado de Vigenère se hace de esta manera: se toma el texto plano `MUCHOSANOSDESPUESFRENTEALPELOTON` como clave, se elige un palabra como `CLAVE` y se repite cíclicamente debajo del texto:

```
MUCHOSANOSDESPUESFRENTEALPELOTON
CLAVECLAVECLAVECLAVECLAVECLAVECL
```

y después, en cada columna se hace una suma como en un cifrado de César. Bajar el archivo `vigenere.txt` de la página EVA e implementar una manera de desencriptar el texto. En el archivo para entregar se deberán incluir el texto desencriptado (como una cadena), las funciones que usaron para desencriptar y un notebook compartido conmigo donde me muestran como lo desencriptaron.

6. **Un cifrado uruguayo.** En el libro *Ciao, Napolitano* por el autor (y profesor en esta Facultad) se encuentre el texto cifrado contenido en el archivo `ciao.txt` colgado en la página EVA. Bajar este archivo e implementar una manera de desencriptar el text. En el archivo para entregar se deberán incluir el texto desencriptado (como una cadena), las funciones que usaron para desencriptar y un notebook compartido conmigo donde me muestran como lo desencriptaron.