Practica 2 Seguridad Informática

March 11, 2021

Codificación Aritmética (expresión en decimal)

[66]:
```
# Se importan las librerias necesarias
from fractions import Fraction
import math
```

1 Partiendo de una fuente de información F equiprobable con alfabeto: alf = "ABCDEFGHIJKLMNNOPQRSTUVWXYZ ." Después de aplicar el proceso de codificación aritmética (expresión en decimal) a un mensaje msj, donde se utiliza como fuente base la fuente F, se obtiene el número decimal "0.16136576072167237983461105 83". Sabiendo que el mensaje msj tiene longitud 19, calcula dicho mensaje.

[67]:
```
message1 = decodeNumber(Fraction("0.161365760721672379834611058 3"), 19,␣
↪"ABCDEFGHIJKLMNÑOPQRSTUVWXYZ .")
print("El mensaje decodificado es: " +message1)
```

```
El mensaje decodificado es: ESTO ES UNA PRUEBA.
```

2 A partir del texto en el fichero "dataEx2.txt" y usando como referencia los símbolos que aparecen en el texto (diferenciando mayúsculas de minúsculas y tomando el cambio de línea como dos espacios) y el número de veces que aparece cada símbolo, calcula:

2.1 La fuente de información F asociada al texto y apunta la entropía en bits de dicha fuente.

[68]:
```
text2 = getText("dataEx2.txt")
entropy2 = getEntropy(getOdds(text2))
print("Entropia de la fuente de informacion:", entropy2)
```

```
print("Entropia de la fuente de informacion redondeada al tercer decimal:",␣
 ↪round(entropy2,3))
```

```
Entropia de la fuente de informacion: 4.096886085245035
Entropia de la fuente de informacion redondeada al tercer decimal: 4.097
```

2.2 Después de aplicar el proceso de codificación aritmética (expresión en decimal) a un mensaje msj, usando como fuente base la fuente F, se obtiene el número decimal: "0.964028162700367367709579755642556305640 09". Sabiendo que el mensaje msj tiene longitud 27, calcula dicho mensaje.

[69]:
```
message2 = decodeNumber(Fraction("0.247276109705412160222"), 17, text2)
print("El mensaje decodificado es: " +message2)
```

```
El mensaje decodificado es: el tiempo es vida
```

3 Usando como referencia los símbolos que aparecen en el texto del fichero "dataEx3.txt" (diferenciando mayúsculas de minúsculas y tomando el cambio de línea como dos espacios) y el número de veces que aparece cada símbolo, calcula:

3.1 La fuente de información F asociada al texto y apunta la entropía en bits de dicha fuente.

[70]:
```
text3 = getText("dataEx3.txt")
entropy3 = getEntropy(getOdds(text3))
print("Entropia de la fuente de informacion:", entropy3)
print("Entropia de la fuente de informacion redondeada al tercer decimal:",␣
 ↪round(entropy3,3))
```

```
Entropia de la fuente de informacion: 4.359154745324074
Entropia de la fuente de informacion redondeada al tercer decimal: 4.359
```

3.2 Después de aplicar el proceso de codificación aritmética (expresión en decimal) a un mensaje msj, usando como fuente base la fuente F, se obtiene el número decimal: "0.964028162700367367709579755642556305640 09". Sabiendo que el mensaje msj tiene longitud 27, calcula dicho mensaje.

[71]:
```
message3 = decodeNumber(Fraction("0.
 ↪964028162700367367709579755642556305640 09"), 27, text3)
print("El mensaje decodificado es: " +message3)
```

```
El mensaje decodificado es: (primera parte, capítulo 1)
```

4 Funciones auxiliares

[72]:
```
# Dar formato a un texto de un fichero

def getText(path):
    # Se obtiene el texto del fichero
    file = open(path, 'r', encoding='utf8')
    text = file.read()
    file.close()

    # Se cambia cada salto de linea por dos espacios como manda el guión
    text = text.replace("\n", "  ")

    return text
```

[73]:
```
# Getter de frecuencias y probabilidades dado un alfabeto

def getFrequencies(alphabet):
    frequencies = dict()
    for symbol in alphabet:
        if symbol not in frequencies:
            frequencies[symbol] = alphabet.count(symbol)
    return frequencies

def getOdds(alphabet):
    odds = dict()
    for symbol in alphabet:
        if symbol not in odds:
            odds[symbol] = Fraction(alphabet.count(symbol), len(alphabet))
    return odds
```

[74]:
```
# Calculador de la entropia dada sus probabilidades de aparicion

def getEntropy(odds):
    entropy:float = 0
    for key in odds:
        entropy += odds[key] * math.log2(1/odds[key])
    return entropy
```

[75]:
```
# Decodificador iterativo del numero decimal

def decodeNumber(number, messageLength, alphabet):
    # Se obtienen los rangos para cada probabilidad.
    intervals = dict()
    actualMin = 0
    odds = getOdds(alphabet)
    for symbol in odds:
```

```
        intervals[symbol] = [Fraction(actualMin),␣
↪Fraction(actualMin+odds[symbol])]
        actualMin += Fraction(odds[symbol])

    message = ""
    for i in range(messageLength):
        # Se recorren todos los posibles intervalos para cada simbolo.
        for symbol, interval in intervals.items():
            if Fraction(interval[0]) <= number < Fraction(interval[1]):
                # Si el numero esta en el intervalo correcto se añade la letra␣
↪y se actualiza el numero para el siguiente caracter.
                number = Fraction((number - interval[0]) / (interval[1] -␣
↪interval[0]))
                message += symbol
                # Ya hemos añadido el simbolo, pasamos al siguiente simbolo.
                break

    return message
```

por Adrián Pérez Garcia, 12/03/2021