

Prácticas de SAR

Sistemas de Almacenamiento y Recuperación de información

Práctica 3: El Mono Infinito

El mono infinito

El mono infinito

Descripción del problema



*El teorema del mono infinito afirma que un mono pulsando teclas **al azar** sobre un teclado durante **un periodo de tiempo infinito** casi seguramente podrá escribir finalmente cualquier libro que se halle en la Biblioteca Nacional de Francia.*

El mono infinito

Objetivo de la práctica

Ya que no tenemos tanto tiempo, crearemos un programa en Python que procese un documento y que utilice la información extraída de él para ayudar al mono a escribir su libro.

Ejercicio

¿Qué debo hacer?

Se proporcionan 4 ficheros:

- `SAR_p3_monkey_indexer.py`: crea un índice a partir de un texto.
- `SAR_p3_monkey_info.py`: genera información a partir de un índice.
- `SAR_p3_monkey_evolved.py`: genera frases a partir de un índice.
- `SAR_p3_monkey_lib_plantilla.py` es una plantilla donde debes **completar y DOCUMENTAR** los siguientes métodos de la clase Monkey:

- `index_sentence`
- `compute_index`
- `generate_sentences`

SAR_p3_monkey_indexer.py

■ El programa `SAR_p3_monkey_indexer.py`:

- 1) Recibe como argumento el nombre de un fichero de texto,
- 2) lo divide en frases, las tokeniza y crea un índice donde acumula estadísticas de qué palabras siguen a otras, y
- 3) guarda el índice en un fichero binario con el mismo nombre del fichero de texto pero con extensión `.index`.

■ Ejemplo de ejecución:

```
prompt> python SAR_p3_monkey_indexer.py spam.txt
```

SAR_p3_monkey_indexer.py

Tokenización:

- Las frases se separan con “.”, “:”, “!”, “?” o dos saltos de línea.
- Se eliminarán todos los símbolos no alfanuméricos.
- Los tokens serán las palabras del documento en minúsculas.
- Se añadirá el símbolo “\$” que indica inicio y final de frase.

Creador de índices:

- El índice se guardará como un diccionario Python.
- Las claves del diccionario serán los tokens del documento más la palabra especial \$.
- Cada entrada del diccionario contendrá:
 - El número total de veces que ha aparecido el token.
 - Una lista con todas los tokens que han aparecido en el documento después de la clave (incluido "\$") y la frecuencia.
- La lista de sucesores debe estar ordenada por el número de apariciones del par de tokens.

SAR_p3_monkey_indexer.py

Fichero spam.txt utilizado en los ejemplos:

```
Egg and Bacon;  
Egg, sausage and Bacon;  
Egg and Spam;  
Spam Egg Sausage and Spam;  
Egg, Bacon and Spam;  
Egg, Bacon, sausage and Spam;  
Spam, Bacon, sausage and Spam;  
Spam, Egg, Spam, Spam, Bacon and Spam;  
Spam, Spam, Spam, Egg and Spam;  
Spam, Spam, Spam, Spam, Spam, Spam, Spam, baked beans, Spam, Spam, Spam  
and Spam;  
Lobster Thermidor aux crevettes with a Mornay sauce, garnished with  
truffle pate, brandy and a fried egg on top and Spam
```

SAR_p3_monkey_indexer.py

Objeto Python guardado en el .index

```
{'name': 'spam.txt', 'bi': {'$': (11, [(5, 'spam'), (5, 'egg'), (1, 'lobster')]), 'egg': (9, [(3, 'and'), (2, 'sausage'), (2, 'bacon'), (1, 'spam'), (1, 'on')]), 'and': (12, [(9, 'spam'), (2, 'bacon'), (1, 'a')]), 'bacon': (6, [(2, 'sausage'), (2, 'and'), (2, '$')]), 'sausage': (4, [(4, 'and')]), 'spam': (27, [(11, 'spam'), (9, '$'), (3, 'egg'), (2, 'bacon'), (1, 'baked'), (1, 'and')]), 'baked': (1, [(1, 'beans')]), 'beans': (1, [(1, 'spam')]), 'lobster': (1, [(1, 'thermidor')]), 'thermidor': (1, [(1, 'aux')]), 'aux': (1, [(1, 'crevettes')]), 'crevettes': (1, [(1, 'with')]), 'with': (2, [(1, 'truffle'), (1, 'a')]), 'a': (2, [(1, 'mornay'), (1, 'fried')]), 'mornay': (1, [(1, 'sauce')]), 'sauce': (1, [(1, 'garnished')]), 'garnished': (1, [(1, 'with')]), 'truffle': (1, [(1, 'pate')]), 'pate': (1, [(1, 'brandy')]), 'brandy': (1, [(1, 'and')]), 'fried': (1, [(1, 'egg')]), 'on': (1, [(1, 'top')]), 'top': (1, [(1, 'and')])}}
```

SAR_p3_monkey_info.py

■ Ejecución

```
python SAR_p3_monkey_info.py spam.index
```

■ **Funcionalidad** de SAR_p3_monkey_info.py:

- 1) Recibe como argumento el nombre de un fichero de índice,
- 2) crea un fichero de texto con la extensión `.info` con información del índice.

SAR_p3_monkey_info.py

Ejemplo de fichero spam.info generado:

```
#####
#      INFO      #
#####
filename: 'spam.txt'

#####
#      BIGRAMS   #
#####
$  =>  11  =>  spam:5 egg:5 lobster:1
a   =>   2  =>  mornay:1 fried:1
and =>  12  =>  spam:9 bacon:2 a:1
aux =>   1  =>  crevettes:1
bacon  =>  6  =>  sausage:2 and:2 $:2
.
.
spam   => 27  =>  spam:11 $:9 egg:3 bacon:2 baked:1 and:1
thermidor  =>  1  =>  aux:1
top  =>  1  =>  and:1
truffle =>  1  =>  pate:1
with   =>  2  =>  truffle:1 a:1
```

SAR_p3_monkey_evolved.py

■ Ejecución

```
prompt> python SAR_p3_monkey_evolved.py spam.index
```

■ Funcionalidad

- 1) Recibe como argumento el nombre de un fichero de índice y, opcionalmente, un nº entero como segundo argumento.
- 2) Utiliza la información del índice para generar frases.
- 3) Genera 10 frases si no se le indica una cantidad como segundo argumento.

SAR_p3_monkey_evolved.py

Ejemplo de ejecución

```
> python SAR_p3_monkey_evolved.py spam.index 7
```

```
egg and a fried egg on top and bacon
```

```
egg spam spam spam egg spam bacon sausage and spam spam spam baked  
beans spam
```

```
lobster thermidor aux crevettes with truffle pate brandy and spam  
spam
```

```
egg bacon and a mornay sauce garnished with truffle pate brandy  
and spam
```

```
spam spam bacon
```

```
spam
```

```
egg on top and bacon sausage and spam spam
```

SAR_p3_monkey_evolved.py

- ¿Cómo se inicia cada frase?
 - Se elige como **palabra inicial \$**.
- ¿Cómo se elige cada palabra siguiente?
 - Las palabras siguientes se eligen sucesivamente de forma “**aleatoria ponderada**” entre las sucesoras de la palabra actual teniendo en cuenta el número de veces que ha aparecido.
- ¿Cuándo se termina una frase?
 - La palabra siguiente elegida es la **palabra “final” especial \$**, o
 - se llega a un número **máximo de palabras, 25** en nuestro caso.

Cosas útiles

Guardar objetos python en un fichero

pickle

```
import pickle

def save_object(obj, filename):
    with open(file_name, 'wb') as fh:
        pickle.dump(obj, fh)

def load_object(filename):
    with open(file_name, 'rb') as fh:
        obj = pickle.load(fh)
    return obj
```

Números “aleatorios” en python

librería random

```
import random
```

```
random.randint(a, b)
```

Return a random integer N such that $a \leq N \leq b$.

```
random.choice(seq)
```

Return a random element from the non-empty sequence seq. If seq is empty, raises IndexError.

```
choices(population, weights=None, *, cum_weights=None, k=1)
```

Return a k sized list of population elements chosen with replacement.

If the relative weights or cumulative weights are not specified,
the selections are made with equal probability.

Ampliación

Ampliación: *trigramas*

- Al llamar a **SAR_p3_monkey_indexer.py** podremos añadirle un segundo argumento `tri` para que genere en el índice información sobre *trigramas*:

```
prompt> python SAR_p3_monkey_indexer.py spam.txt tri
```

- Se creará otro índice en el que cada entrada será una tupla (w_1, w_2) que enlazará con las palabras vistas detrás de w_1 y w_2 .
- Si se utilizan trigramas el nombre del índice debe terminar en `_tri.index`
- Se deberá modificar **SAR_p3_monkey_info.py** para que muestre también la información del índice de trigramas.
- La generación de frases por parte de **SAR_p3_monkey_evolved.py** se realizará teniendo en cuenta el índice de trigramas.