Resolucion del problema

GENERACION AUTOMATICA DE ORACIONES

(Septiembre de 2020)

Ivan Fernando Mujica Mamani  
Universidad Catolica Boliviana UCB

**Resumen**—En anos recientes el campo de la ciencia de datos ha tomado especial importancia en la toma de deciciones de las empresas e instituciones, una parte vital de este campo es el dominio de un lenguaje con una amplia implementacion de librerias y paquetes para el manejo e interpretacion de datos; Python es uno de los lenguajes mas populares para la aplicacion de ciencia de datos, en parte gracias a que tiene una de las curvas de aprendizaje menos pronunciadas. En el presente articulo se implementa un programa que resuelve el problema de la generacion automatica de oraciones mediante la conjugacion aleatoria de elementos de un conjunto de listas de articulos, sujetos, verbos, preposiciones y adverbios, el resultado del programa es el stdout del sistema operativo residente.

**Indice de terminos**—Python, Ciencia de Datos, stdout

——————————  ——————————

# 1 Introduccion

————————————————

* I.M. is with the Universidad Catolica Boliviana, BO. E-mail: ifmm87@gmail.com.

El presente articulo explicamos como se implementa un programa para el problema Generacion automatica de oraciones mediante el uso de Python en su version 3.8, para tal efecto se uso la libreria *random* perteneciente al core del lenguaje. Debido a que este lenguaje implementa tipado dinamico y programacion funcional es mucho mas sencillo y la curva de aprendizaje es menos pronunciada en relacion a otros lenguajes como Java o C++. Es necesario contar con un ambiente de desarrollo con Python y algun editor de codigo.

Se necesita un programa que genere N oraciones aleatorias, una oracion basica debe estar compuesta por una secuencia similar a articulo, sujeto, verbo y adverbio; siendo el adverbio opcional ya que solo enfatiza la accion de verbo, nuestro programa genera oraciones con cualquiera de estas dos estructuras como indica el enunciado.

# 2 **Metodologia**

## 2.1 Seleccion aleatoria

La seleccion aleatoria esta implementada gracias al paquete random desde el core de Python, mediante el metodo *choice,* tecnicamente lo que hace es elegir al azar un elemento de una listay la funcion *randint,* este ultimo recibe dos argumentos que serian el limite inferior y superior del numero entero aleatorio que se desea generar*.* La especificacion completa de estas funciones se encuentra en <https://docs.python.org/3/library/random.html>

## 2.2 **Conjugacion de palabras**

Para la conjugacion de palabras necesitamos una estructura tipo array para contener la oracion generada o simplemente podriamos enviarlo directamente al stdout del sistema operativo en una sola cadena formateada que contenga los elementos elegidos de cada array, la funcion print recibe una cadena que puede ser formateada con variables gracias al uso de llaves {} como comodin del template. La especificacion completa de la funcion print en sus diferentes alternativas esta en <https://docs.python.org/3/tutorial/inputoutput.html>.

## 2.3 **Construccion del programa**

Primeramente necesitamos un conjunto de listas, estas pueden contener cuantas palabras necesitemos. En el programa declaramos 4 arrays (Articulos, Sujetos, Verbos y Adverbios) .

# declaring arrays

articles = ['the', 'an', 'a', 'those', 'that', 'these', 'this']

subjects = [ 'cat', 'puppy','friend', 'lover', 'woman', 'children', 'parents', 'relatives', 'neighibors' ]

verbs = ['eat', 'loves', 'run away', 'shows', 'picks', 'sit down', 'end up', 'turn out' ]

adverbs = ['loudly', 'well', 'badly','always', 'quietly', '']

Una vez declarado los arrays necesitamos una bifurcacion para preguntar cual de las dos estructuras se va a generar, para eso utilizamos la funciones *randint y choice* .

for n in range(int(inp)) :

if random.randint(0,1) == 1 :

print(f'{n + 1} {random.choice(articles)} {random.choice(subjects)} {random.choice(verbs)} {random.choice(adverbs)}')

else :

print(f'{n + 1} {random.choice(articles)} {random.choice(subjects)} {random.choice(verbs)}')

# Conclusion

Por lo expuesto en el presente documento podemos apreciar la flexibilidad del lenguaje de programacion en cuanto a la mensajeria con el Sistema Operativo se refiere, asi como tambien un poco de la amplia coleccion de librerias implementadas en Python. Es importante profundizar en las funciones que ya tiene incorporado para el maximo de provecho al lenguaje.

**Acknowledgment**

The authors wish to thank A, B, C. This work was supported in part by a grant from XYZ.

**References**

1. J.S. Bridle, “Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition,” *Neurocomputing—Algorithms, Architectures and Applications,* F. Fogelman-Soulie and J. Herault, eds., NATO ASI Series F68, Berlin: Springer-Verlag, pp. 227-236, 1989. (Book style with paper title and editor)
2. W.-K. Chen, *Linear Networks and Systems.* Belmont, Calif.: Wadsworth, pp. 123-135, 1993. (Book style)
3. H. Poor, “A Hypertext History of Multiuser Dimensions,” *MUD History,* http://www.ccs.neu.edu/home/pb/mud-history.html. 1986. (URL link \*include year)