

## **1. Objetivo**

Este proyecto busca la implementación de un software que contenga distintas metodologías para analizar el estilo de distintos textos escritos con el fin de colaborar con una herramienta práctica a la atribución de autoría. El código está en el lenguaje de programación Python.

## **2. Alcance del proyecto**

El presente trabajo busca crear el primer diseño de un software que pueda ser utilizado en el área de la Lingüística Forense, en el campo de los textos escritos y reconocimiento de autoría.

Ya se han elaborado herramientas para alcanzar el fin anterior, como Sistema Automático para Estudios Estilométricos (SAUTEE), diseñado por el Grupo de Ingeniería Lingüística de la UNAM,<sup>1</sup> sin embargo, aún pueden implementarse algunas mejoras. Por ejemplo, el SAUTEE trabaja con un grupo específico de textos cargados, previamente, por el autor del sistema. El SAE busca ampliar la accesibilidad del programa para que pueda usarse en distintos ámbitos. El desarrollo del SAUTEE no es accesible al público, por lo tanto, el presente proyecto es un desarrollo completamente original.

## **3. Introducción**

La estilística es el estudio del uso del lenguaje. Principalmente, se enfoca en textos escritos, sin embargo, en la actualidad también se ha empleado en otros ámbitos artísticos, tales como la música o la pintura.

Su uso ha sido ampliamente examinado por lingüistas ya que se han desarrollado diversas formas de analizar el lenguaje por medio de los distintos niveles lingüísticos presentes, que van desde rasgos tan individuales como los n-gramas, que son los elementos consecutivos en una cierta secuencia, hasta la cantidad de palabras contenidas en el texto, por nombrar algunas.

---

<sup>1</sup> La recopilación de herramientas se puede encontrar en <http://www.corpus.unam.mx/geco>

Ciertamente, un estudio estilométrico contribuye al reconocimiento de autores de un texto, ya que cada persona presente una forma personal de escritura. Se ha llegado a decir que el uso del lenguaje, incluso, puede fungir como la huella personal, es decir, que cada individuo es único en este aspecto.

El desarrollo de la tecnología y de la computación son factores de gran relevancia en el asunto, ya que todos los conocimientos en el área descrita pueden ser realizados automáticamente, lo que provee una mayor eficiencia y rapidez.

Python, por ejemplo, es un lenguaje de programación bastante útil en el estudio del lenguaje, por ello es empleado en distintos campos, como el procesamiento del lenguaje natural. Con ayuda de Python, será posible descomponer los textos deseados de distintas maneras con el fin de obtener diversos resultados que contribuyan a decidir la autoría de un texto.

Actualmente, es bastante complicado llegar a un resultado inmediato o absoluto que nos revele con exactitud el autor de un texto, sin embargo, a lo largo de los años, se han desarrollado varios métodos con los cuales se pueden realizar análisis diversos que nos den una idea o una aproximación hacia nuestra búsqueda. Por ello, es necesario tener presente, en todo momento, que los resultados deben ser interpretados por una persona, de preferencia si es un experto en el tema. No obstante, esto último no descalifica el programa en cuestión, ya que los resultados que éste arroje serán fácilmente entendidos por cualquier persona. Holmes (1994) hizo un recuento de los métodos que muchos investigadores han probado; por lo tanto, en este proyecto, se han tomado aquellos que tienen una buena aceptación, debido a su eficacia, en el área de atribución de autoría:

### **3.1. Riqueza léxica**

Aunque no siempre resulte ser verdadero, la postura de que una persona con mayor nivel de educación tiene un vocabulario más amplio que aquella con menor nivel de estudios es un buen indicio para discriminar textos de distintos autores. Este método puede tratarse por separado o puede tener algunos subíndices que contribuyan a un mejor desempeño.

### 3.2 Dice similarity

El Dice Similarity es una prueba estadística que sirve para medir la distancia entre dos textos, es decir, para saber qué tan similares son por medio de su longitud total. Para calcular la similitud entre textos, se usará la siguiente ecuación, donde  $D_n$  son los textos analizados,  $m_c$  las palabras que ambos tienen en común y  $m_n$  el total de palabras de cada texto.

$$Sim_D(\vec{D}_1, \vec{D}_2) = \frac{2m_c}{m_1 + m_2}$$

### 3.3 Distribución de las categorías gramaticales

Las etiquetas POS (Parts of Speech) o de categorías gramaticales son útiles en la atribución de autoría. El uso constante de una sola categoría gramatical (ya sea sustantivo, adjetivo, etc.) refleja un estilo característico de un individuo y este se repite a lo largo de todos los textos que produzca la misma persona.

En este proyecto, se toman las cinco categorías gramaticales del español que son los sustantivos, los adjetivos, los verbos, los adverbios y los determinantes. La frecuencia de uso de cada uno de ellos contribuye a crear un perfil del autor, saber cómo escribe.

## 4. Desarrollo

El proyecto está elaborado en Python debido a su versatilidad para trabajar con el lenguaje. La librería más importante para este trabajo es NLTK (Natural Language Toolkit) y la herramienta principal es Freeling,<sup>2</sup> cuya función es realizar el etiquetado de categorías gramaticales de un texto, la cual se ocupa en la mayor parte del trabajo.

Para comenzar, se realizó una función que obtiene los nombres de los archivos que ingresa el usuario, estos son los que el sistema va a comparar hasta que sea cerrado.

---

<sup>2</sup> Para consultar el sistema: <http://www.corpus.unam.mx/servicio-freeling/>

```

1 #bibliotecas
2 import requests
3 import nltk
4 import numpy
5
6 valores = [0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3]
7 T = "Textos/"
8 t = ".txt"
9
10 #Abre Los textos y Los almacena en distintas variables
11 def abrir_archivos():
12     print("Introduzca el nombre de los textos a analizar")
13     print("Los archivos deben estar en archivos de extensión .txt")
14     print("No es necesario anotar la extensión en este espacio")
15     documento01 = input("¿Cuál es su primer documento?: ")
16     documento02 = input("¿Cuál es su segundo documento?: ")
17
18     with open(T + documento01 + t, "r", encoding = 'utf8') as doc:
19         texto1 = doc.read()
20     with open(T + documento02 + t, "r", encoding = 'utf8') as doc:
21         texto2 = doc.read()
22     return documento01, documento02, texto1, texto2
23
24 documento01, documento02, texto1, texto2 = abrir_archivos()

```

**Imagen 1.** Bibliotecas empleadas y definición de textos

Por medio de Freeling, como se mencionó al principio, se obtienen las etiquetas de las categorías gramaticales. Éstas se almacenan en diccionarios que, posteriormente se convertirán en listas para efectos del programa.

```

54 #Conseguir el etiquetado de las categorías gramaticales de ambos textos
55 def Etiquetado_POS():
56     #Código copiado del servicio Freeling http://www.corpus.unam.mx/servicio-freeling/
57     files = {'file': open(T + documento01 + t, 'rb')}
58     params = {'outf': 'tagged', 'format': 'json'}
59     url = "http://www.corpus.unam.mx/servicio-freeling/analyze.php"
60     r = requests.post(url, files=files, params=params)
61     obj1 = r.json()
62
63     files = {'file': open(T + documento02 + t, 'rb')}
64     params = {'outf': 'tagged', 'format': 'json'}
65     url = "http://www.corpus.unam.mx/servicio-freeling/analyze.php"
66     r = requests.post(url, files=files, params=params)
67     obj2 = r.json()
68
69     return obj1, obj2
70
71 POS_TEXT01, POS_TEXT02 = Etiquetado_POS()
72

```

**Imagen 2.** Código de Freeling

Las siguientes funciones se refieren a los métodos mencionados en la Introducción, los cuales emplean distintos tipos de listas para almacenar toda la información, ya sea para emplearla después o en la misma función. Por ejemplo, en la función de *Dice\_similarity*, se crean dos listas llamadas *WORDS1* y *WORDS2*, cada una guarda las palabras que se encuentran en el texto 1 y en el texto 2, respectivamente. Posteriormente, la tercera lista *COMMON\_WORDS*, contiene las palabras que tienen en común ambos documentos sin que ninguna sea repetida.

```

87 #Esta función obtiene las palabras en común que tienen ambos textos y las divide entre la suma de las palabras totales de am
88 def Dice_similarity():
89     print("Dice Similarity")
90     cont = 0
91     WORDS1 = []
92     WORDS2 = []
93     COMMON_WORDS = []
94     token = 'token'
95     lemma = 'lemma'
96     i = 50
97
98     for oracion in POS_TEXT01:
99         for palabra in oracion:
100             WORDS1.append(palabra[lemma])
101
102     for oracion in POS_TEXT02:
103         for palabra in oracion:
104             WORDS2.append(palabra[lemma])
105
106     for palabra in WORDS1:
107         if palabra in WORDS2:
108             if palabra not in COMMON_WORDS:
109                 cont = cont + 1
110                 COMMON_WORDS.append(palabra)
111
112     print("Palabras comunes |", len(COMMON_WORDS))
113     print("Palabras texto 1 |", len(WORDS1))
114     print("Palabras texto 2 |", len(WORDS2))
115     print("Acercamiento | "+str((len(COMMON_WORDS))/(len(WORDS1) + len(WORDS2) - len(COMMON_WORDS))))
116     return (2*len(COMMON_WORDS))/(len(WORDS1) + len(WORDS2))

```

**Imagen 3.** Definición de la función *Dice\_similarity*

Por último, en el cuerpo principal del código, se llaman a todas las funciones descritas anteriormente por medio de la estructura de control *switch*, aunque, por trabajar en Python, se encuentra como estructura de condición *if, elif, else*.

```

1 POS_TEXT01, POS_TEXT02 = Etiquetado_POS()
2 lista = {}
3
4 print("Bienvenido al SAE (Sistema de Análisis Estilométrico)")
5 print("En este programa podrás encontrar varias opciones para analizar y comparar dos textos distintos")
6 print("")
7
8 while True:
9
10     print("")
11     opciones(1)
12     opcion = int(input("¿Qué desea hacer?: "))
13     print("")
14
15     if opcion is 1:
16         #Llama a la función Riqueza_lexica y guarda el resultado en la variable lista
17         lista["Riqueza léxica"] = Riqueza_lexica()
18     elif opcion is 2:
19         #Llama a la función Dice_similarity y guarda el resultado en la variable lista
20         lista["Dice similarity"] = Dice_similarity()
21     elif opcion is 3:
22         opciones(2)
23         POS = int(input("¿Qué categoría gramatical necesitas?: "))
24         print("")
25         #Llama a la función Contar_pos y guarda el resultado en la variable lista
26         texto, a = Contar_pos(POS)

```

**Imagen 4.** Cuerpo principal del código

La opción cuatro del programa es, posiblemente, la herramienta más importante de todo el programa, ya que, por medio de una lista, almacena toda la información que se va recopilando durante su ejecución. Una vez seleccionada esta opción, el programa revisa cada uno de los elementos que contiene; si la lista está vacía, devuelve una advertencia "Se necesitan más datos para continuar / Por favor, haga otra prueba". De lo contrario, si la lista tiene, al menos, un elemento, se obtendrá el promedio y regresará un valor aproximado, o no, a 1, lo cual indica si el análisis ha sido exitoso (ambos textos fueron escritos por el mismo autor) o no (cada texto fue escrito por un autor diferente).

```

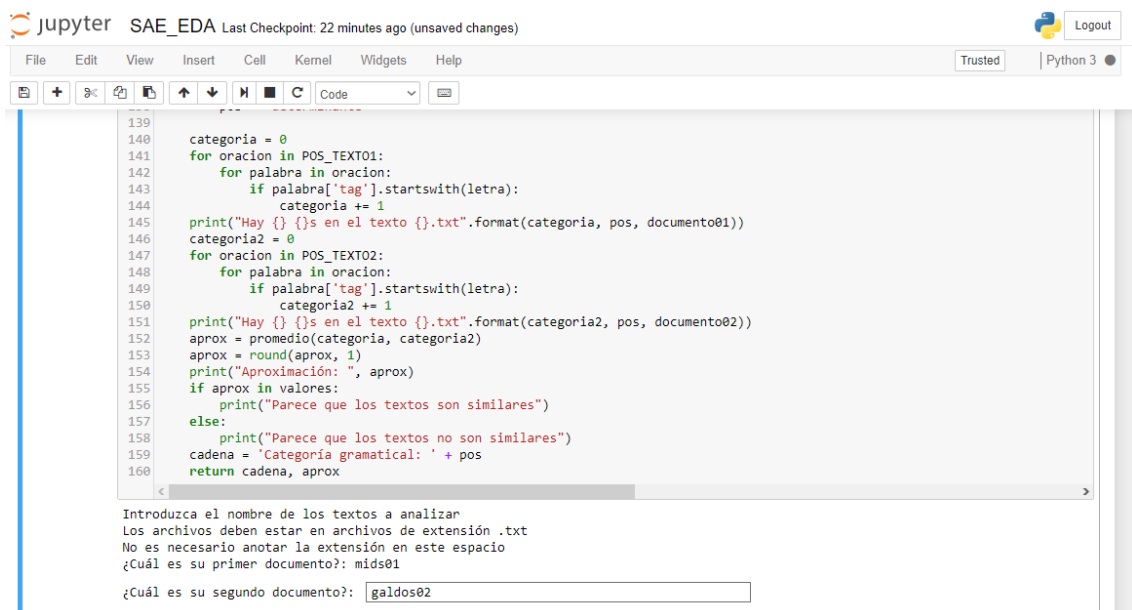
elif opcion is 4:
    print("Conteo general")
    datos = 0
    if len(list(lista)) <= 1:
        print("Se necesitan más datos para continuar")
        print("Por favor, haga otra prueba.")
    else:
        #Obtiene el promedio de todos los datos almacenados en la lista
        datos = sum(lista.values())
        resultado = promedio(datos, len(list(lista)))
        for elemento in list(lista):
            print("En {}, se obtuvo un resultado de {}".format(elemento, lista.get(elemento)))
        print("")
        print("El promedio de información analizada hasta el momento es igual a {}".format(resultado))
        print("Mientras más próximo esté el resultado a 1, más similitud existe entre los textos")

```

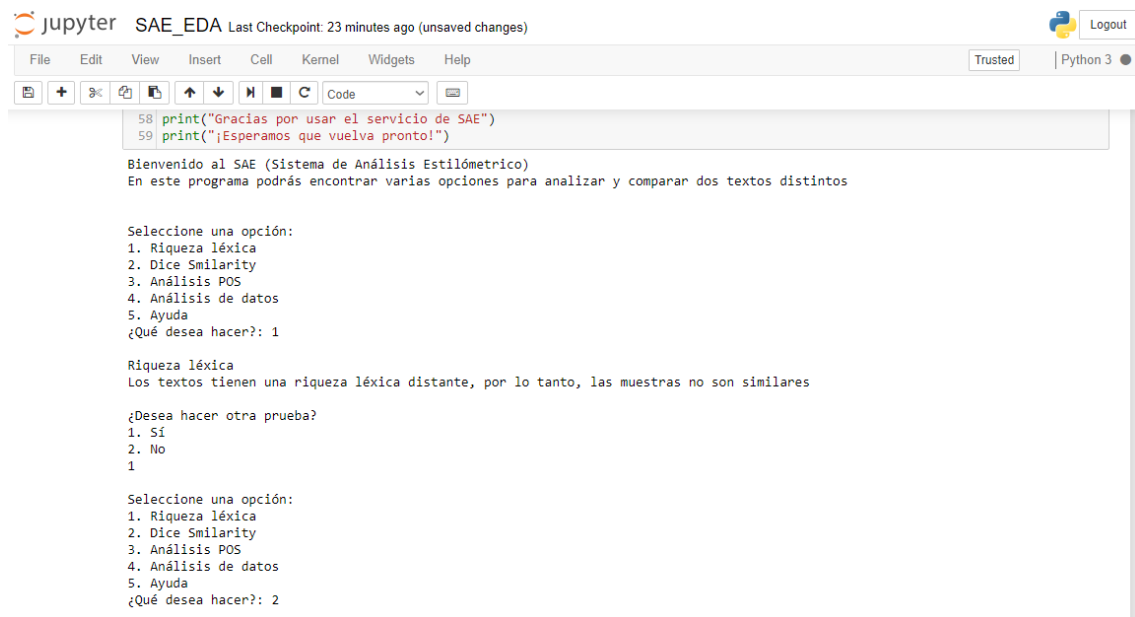
**Imagen 5.** Opción cuatro, la cual obtiene el resultado fundamental del programa

## 5. Resultados

Con todo lo anterior, se consiguió un programa totalmente funcional y que devuelve resultados bastante confiables, como se muestra en las siguientes imágenes.



**Imagen 6.** Menú en donde se introducen los textos



```
58 print("Gracias por usar el servicio de SAE")
59 print("¡Esperamos que vuelva pronto!")

Bienvenido al SAE (Sistema de Análisis Estilométrico)
En este programa podrás encontrar varias opciones para analizar y comparar dos textos distintos

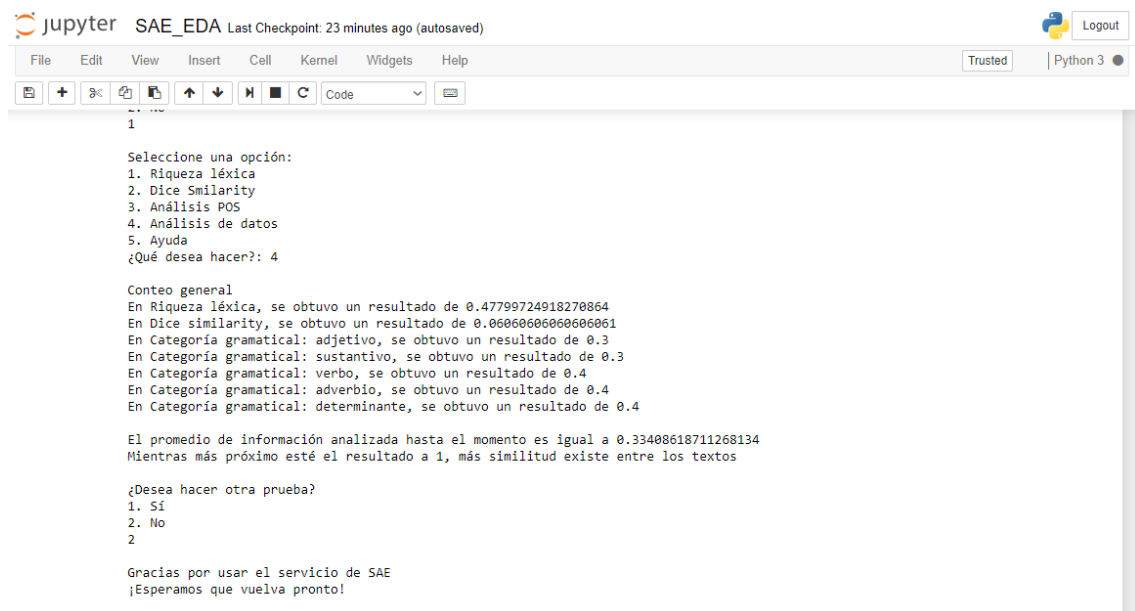
Seleccione una opción:
1. Riqueza léxica
2. Dice Smilarity
3. Análisis POS
4. Análisis de datos
5. Ayuda
¿Qué desea hacer?: 1

Riqueza léxica
Los textos tienen una riqueza léxica distante, por lo tanto, las muestras no son similares

¿Desea hacer otra prueba?
1. Si
2. No
1

Seleccione una opción:
1. Riqueza léxica
2. Dice Smilarity
3. Análisis POS
4. Análisis de datos
5. Ayuda
¿Qué desea hacer?: 2
```

**Imagen 7.** Menú de análisis y pruebas realizadas



```
1

Seleccione una opción:
1. Riqueza léxica
2. Dice Smilarity
3. Análisis POS
4. Análisis de datos
5. Ayuda
¿Qué desea hacer?: 4

Cuento general
En Riqueza léxica, se obtuvo un resultado de 0.47799724918270864
En Dice similarity, se obtuvo un resultado de 0.06060606060606061
En Categoría gramatical: adjetivo, se obtuvo un resultado de 0.3
En Categoría gramatical: sustantivo, se obtuvo un resultado de 0.3
En Categoría gramatical: verbo, se obtuvo un resultado de 0.4
En Categoría gramatical: adverbio, se obtuvo un resultado de 0.4
En Categoría gramatical: determinante, se obtuvo un resultado de 0.4

El promedio de información analizada hasta el momento es igual a 0.33408618711268134
Mientras más próximo esté el resultado a 1, más similitud existe entre los textos

¿Desea hacer otra prueba?
1. Si
2. No
2

Gracias por usar el servicio de SAE
¡Esperamos que vuelva pronto!
```

**Imagen 8.** Resultados generales del programa

## 6. Conclusiones

Aún queda mucho camino por delante debido a que la implementación de programas que contribuyan al análisis estilométrico es un campo bastante reciente. Sin embargo, varios autores ya han contribuido al presentar pruebas que ayuden con este tipo de pruebas. El SAE ha tomado algunas de esas pruebas, con un implemento original y con

un código abierto al público, para que también pueda ayudar al desarrollo de nuevos programas.

## Referencias

- Burrows, J. (2002). «Delta»: A Measure of Stylistic Difference and a Guide to Likely Authorship. *Literary and Linguistic Computing*, 17, 267-287.  
<https://doi.org/10.1093/lc/17.3.267>
- Burrows, J. (2003). Questions of Authorship: Attribution and Beyond: A Lecture Delivered on the Occasion of the Roberto Busa Award ACH-ALLC 2001, New York. *Computers and the Humanities*, 37(1), 5–32.
- Holmes, D. I. (1994). Authorship Attribution. *Computers and the Humanities*, 28(2), 87–106.