Trabajo práctico final: Texto Predictivo

1 Motivación del problema

La predicción de texto es una tarea de aprendizaje automático que consiste en generar el siguiente carácter, palabra o frase de un texto dado. Esta tarea tiene aplicaciones en diversas áreas, como la traducción automática, la generación de texto creativo y la detección de spam.

En el contexto de la escritura, la predicción de texto puede ser utilizada para ayudar a las personas a escribir más eficientemente. Por ejemplo, se puede utilizar para predecir la siguiente palabra que una persona va a escribir, a partir de las palabras que ya ha escrito. Esto puede ayudar a la persona a ahorrar tiempo y esfuerzo, y a evitar cometer errores.

La predicción de texto se basa en el principio de que la probabilidad de que aparezca una palabra dada depende de las palabras que la preceden. Por ejemplo, la probabilidad de que aparezca la palabra "*gato*" es mayor después de la palabra "*el*" que después de la palabra "*la*".

Existen diversos métodos para implementar la predicción de texto. Un método simple consiste en utilizar un modelo probabilístico que asigne una probabilidad a cada palabra posible, en función de las palabras que tiene alrededor.

2 Objetivos del trabajo

En este trabajo se deberá analizar una serie de textos escritos por una misma persona. Estos textos se encontrarán cada uno en un archivo diferente, agrupados dentro de una carpeta que llevará el nombre de la persona que los escribió. El objetivo es utilizar la información obtenida a partir de los textos analizados para luego completar frases escritas por la misma persona que escribió los textos.

3 Detalles de la implementación

Deberán implementarse dos programas: uno en `C` y el otro en `Python`. El primero llamará internamente al segundo para utilizar su funcionalidad, de manera que para ejecutar el trabajo sólo se deberá correr el programa en `C`.

3.1 Programa en C

El programa en `C` tomará como argumento el nombre de la persona cuyos textos se van a analizar. Dicho nombre no deberá contener espacios, en caso de que el nombre contenga más de una palabra, estas deberán estar separadas por guiones bajos. Por ejemplo, si quisiéramos analizar textos escritos por Fito Paez, el argumento de entrada del programa debería ser `Fito_Paez`. En el mismo directorio que el programa se contará con una carpeta llamada `Textos`. Dentro de la misma podremos encontrar una serie

de carpetas, cada una de las cuales tendrá el nombre de una persona respetando el mismo formato que tiene el nombre recibido como argumento.

El programa deberá acceder a la carpeta que corresponde al nombre recibido y procesar todos los archivos que se encuentren dentro. Cada archivo contendrá un texto que puede estar formado por muchas oraciones, puede contener saltos de línea en cualquier lugar y también símbolos de todo tipo (por ejemplo: coma, punto y coma, guión, signos de pregunta, etc.). En los textos a analizar **no** habrá palabras con tilde ni letras especiales como la `ñ`. El objetivo es limpiar todos los textos y unificarlos en un único archivo de salida que deberá tener las siguientes características:

- El archivo deberá tener el mismo nombre que la carpeta inspeccionada, es decir, el nombre recibido como argumento. El mismo tendrá la extensión `.txt`.
- El archivo generado deberá ubicarse dentro de una carpeta llamada `Entradas`, la cual también se encontrará en el mismo directorio que el programa actual.
- Dentro del archivo sólo deberá haber caracteres correspondientes a letras minúsculas y saltos de línea. Deberán quitarse todos los símbolos que se encuentren en los textos analizados y las letras mayúsculas deberán pasarse a minúscula.
- Cada línea del archivo deberá corresponder a una oración de uno de los textos analizados. Se puede identificar el fin de una oración por la presencia del punto. Los saltos de línea que se encuentren en los textos que no estén al final de una oración deberán quitarse también.

Veamos un ejemplo. Supongamos que accedemos a la carpeta con el nombre `Fito_Paez` y dentro encontramos dos archivos: `elamordespuesdelamor.txt` y `unvestidoyunamor.txt`. Dentro de los mismos podemos leer los siguientes textos:

```
El amor despues del amor tal vez
Se parezca a este rayo de sol.
Y ahora que busque
Y ahora que encontre
El perfume que lleva el dolor.
En la esencia de las almas,
En la ausencia del dolor.
Ahora se que ya no puedo
Vivir sin tu amor.
Me hice fuerte ahi,
Donde nunca vi,
Nadie puede decirme quien soy.
Yo lo se muy bien
Que aprendi a querer
El perfume que lleva el dolor.
En la esencia de las almas,
Dice toda religion,
Para mi que es el amor
Despues del amor.
```

```
Te vi,
Juntabas margaritas del mantel.
Ya se que te trate bastante mal.
No se si eras un angel o un rubi,
O simplemente te vi.
Te vi,
Saliste entre la gente a saludar.
Los astros se rieron otra vez.
La llave de mandala se quebro
O simplemente te vi.
Todo lo que diga esta de mas,
Las luces siempre encienden en el alma.
Y cuando me pierdo en la ciudad
Vos ya sabes comprender,
Es solo un rato, no mas.
Tendria que llorar o salir a matar.
Te vi, te vi, te vi.
Yo no buscaba nadie y te vi.
```

El programa en `C` debería generar un archivo llamado `Fito_Paez.txt` dentro de la carpeta `Entradas` con el siguiente contenido:

```
el amor despues del amor tal vez se parezca a este rayo de sol
y ahora que busque y ahora que encontre el perfume que lleva el dolor
en la esencia de las almas en la ausencia del dolor
ahora se que ya no puedo vivir sin tu amor
me hice fuerte ahi donde nunca vi nadie puede decirme quien soy
yo lo se muy bien que aprendi a querer el perfume que lleva el dolor
en la esencia de las almas dice toda religion para mi que es el amor despues del amor
te vi juntabas margaritas del mantel
ya se que te trate bastante mal
no se si eras un angel o un rubi o simplemente te vi
te vi saliste entre la gente a saludar
los astros se rieron otra vez
la llave de mandala se quebro o simplemente te vi
todo lo que diga esta de mas las luces siempre encienden en el alma
y cuando me pierdo en la ciudad vos ya sabes comprender es solo un rato no mas
tendria que llorar o salir a matar
te vi te vi te vi
yo no buscaba nadie y te vi
```

Luego de generar este archivo, el programa deberá llamar al programa escrito en `Python`, pasándole como argumento el nombre recibido.

3.2 Programa en Python

El programa en `Python` tomará como argumento también el nombre de una persona siguiendo el formato establecido. Este deberá acceder a dos archivos. El primero de ellos será el archivo generado por el programa anterior, ubicado en `Entradas/nombre.txt`, reemplazando `nombre` por el argumento recibido. El segundo estará en `Frases/nombre.txt`, la carpeta `Frases` se encontrará en el mismo directorio que las demás. En el segundo archivo se encontrarán frases incompletas (en minúscula y sin símbolos). En cada línea habrá una frase con una palabra faltante indicada por un guión bajo.

El objetivo del programa es reemplazar el guión por una palabra adecuada. Para ello se deberá primero cargar las oraciones del archivo generado por el programa en `C` y almacenarlas de manera que pueda utilizarse esa información para saber qué palabra es más probable que aparezca en el espacio vacío. Siguiendo con el ejemplo anterior, si el argumento recibido es `Fito_Paez`, nuestro programa debería acceder al archivo `Entradas/Fito_Paez.txt` y cargar la información de las oraciones en alguna estructura que sea útil para el propósito establecido. Luego, si dentro del archivo `Frases/Fito_Paez.txt` se encontrara la siguiente frase:

```
te _ fumabas unos chinos en madrid
```

nuestro programa podría reemplazar el espacio vacío por la palabra `vi`. Como salida, el programa deberá generar un nuevo archivo dentro de una carpeta llamada `Salidas` que se encontrará en el mismo directorio. En este caso, se generaría el archivo `Salidas/Fito_Paez.txt` con la siguiente frase dentro:

```
te vi fumabas unos chinos en madrid
```

Todas las frases completas deben ubicarse dentro del mismo archivo de salida, una por línea.

4 Archivos

En resumen, en el mismo directorio que los programas tendremos cuatro carpetas:

- `Textos`: contiene una carpeta por cada persona con archivos escritos por la misma dentro.
- `Entradas`: contiene los archivos creados por el programa en `C`, los cuales agrupan los textos de una persona una vez sanitizados.
- `Frases`: contiene un archivo por cada persona con frases incompletas escritas por la misma.
- `Salidas`: contiene los archivos generados por el programa en `Python`, los cuales contienen las frases de los archivos de la carpeta `Frases` completas.

La cátedra proveerá una carpeta `Textos` y una carpeta `Frases` de ejemplo para que puedan hacer pruebas. Sin embargo, el programa deberá funcionar para cualquier conjunto de archivos que cumpla con las características establecidas.

5 Función system()

La función `system()` es una función de la librería estándar de C (`stdlib.h`) que permite ejecutar un comando del sistema operativo. Esta toma como argumento una cadena de caracteres que representa el comando que se desea ejecutar y devuelve un valor entero que indica el estado de la ejecución del comando. El valor `0` indica que el comando se ejecutó correctamente, mientras que un valor distinto de `0` indica que se produjo un error. La función `system()` es una función peligrosa, ya que permite que los programas ejecuten comandos arbitrarios del sistema operativo. Por esta razón, se recomienda utilizar esta función con precaución.

En este trabajo práctico será necesario utilizar la función `system()` en dos lugares. El primero de ellos es para poder ejecutar el comando `ls` de manera que podamos saber cuántos archivos hay dentro de una carpeta y cuáles son sus nombres. Esto es necesario puesto que en la primera parte del trabajo tenemos únicamente el nombre de la carpeta donde están los archivos que debemos limpiar, pero no sabemos cuántos ni cuáles son. La siguiente llamada a la función `system()` ejecutará el comando `cd Textos/nombre` para movernos a la carpeta que queremos inspeccionar y luego el comando `ls > ../../archivos.txt`. Este segundo comando ejecuta `ls` y vuelca el resultado en el archivo `archivos.txt`, el cual se encontrará en el mismo directorio que nuestro programa. Al hacer esto, dentro de `archivos.txt` encontraremos la lista de los archivos que debemos limpiar, uno por línea.

```
system("cd Textos/nombre && ls > ../../archivos.txt");
```

En el caso de nuestro ejemplo, la sentencia a ejecutar sería:

```
system("cd Textos/Fito_Paez && ls > ../../archivos.txt");.
```

Luego de hacerlo, encontraríamos el siguiente contenido dentro de `archivos.txt`:

```
elamordespuesdelamor.txt
unvestidoyunamor.txt
```

El otro lugar donde será necesario utilizar la función `system()` es para llamar al programa escrito en `Python` desde el programa escrito en `C`. Para esto debemos pasarle como argumento a la función la llamada a realizar. Suponiendo que el programa se llama `programa.py` y que el argumento que debemos pasarle al mismo es `Fito_Paez`, entonces la llamada quedaría como sigue:

```
system("python3 programa.py Fito_Paez");
```

Observación: estos comandos sólo funcionarán en el sistema operativo linux.

6 Evaluación

Para la evaluación del trabajo se tendrán en cuenta los siguientes elementos:

- Estructuras utilizadas para almacenar la información.
- Calidad de código: esto incluye modularización, nombres significativos de funciones y variables, legibilidad, no uso de variables globales, etc.
- Testeo de las funciones que se puedan tanto en `Python` como en `C`.

La entrega del trabajo deberá incluir una descripción general de la solución propuesta en la que se expliquen las decisiones más importantes tomadas.