Logotipo

Descripción generada automáticamente

Sistemas de Información no estructurada

Trabajo del Máster Universitario en Ingeniería Informática

Memoria Practica Final

Victor Fagúndez Poyo

E.T.S de Ingeniería Informática

Curso 2021/2022

|  |  |  |
| --- | --- | --- |
| Historial de Revisiones | | |
| **Versión** | **Fecha** | **Descripción** |
| 0.0.0 | 29.03.22 | Creación del documento |

**Contenido**

[1. Enunciado 1](#_Toc102468550)

[2. Consideraciones iniciales 3](#_Toc102468551)

[3. Algoritmo base 4](#_Toc102468552)

[3.1. Descripción de la implementación realizada 4](#_Toc102468553)

[3.1.1. Problemas encontrados 4](#_Toc102468554)

[3.2. Descripción de los resultados obtenidos 5](#_Toc102468555)

[3.2.1. Evaluación usando 10 cross-fold validation 5](#_Toc102468556)

[4. Limpiando los datos. Todo a minúsculas 6](#_Toc102468557)

[4.1. Descripción de la implementación realizada 6](#_Toc102468558)

[4.2. Descripción de los resultados obtenidos 6](#_Toc102468559)

[4.2.1. Evaluación usando 10 cross-fold validation 6](#_Toc102468560)

[5. Limpiando los datos. Eliminar signos de puntuación 7](#_Toc102468561)

[5.1. Descripción de la implementación realizada 7](#_Toc102468562)

[5.2. Descripción de los resultados obtenidos 8](#_Toc102468563)

[5.2.1. Evaluación usando 10 cross-fold validation 8](#_Toc102468564)

[6. Limpiando los datos. Eliminando Stopwords 9](#_Toc102468565)

[6.1. Descripción de la implementación realizada 9](#_Toc102468566)

[6.1.1. Problemas encontrados 9](#_Toc102468567)

[6.2. Descripción de los resultados obtenidos 10](#_Toc102468568)

[6.2.1. Evaluación usando 10 cross-fold validation 10](#_Toc102468569)

[7. Limpiando los datos. Eliminando nombres de usuarios 12](#_Toc102468570)

[7.1. Descripción de la implementación realizada 12](#_Toc102468571)

[7.2. Descripción de los resultados obtenidos 12](#_Toc102468572)

[7.2.1. Evaluación usando 10 cross-fold validation 12](#_Toc102468573)

[8. Limpiando los datos. Eliminando enlaces 14](#_Toc102468574)

[8.1. Descripción de la implementación realizada 14](#_Toc102468575)

[8.2. Descripción de los resultados obtenidos 14](#_Toc102468576)

[8.2.1. Evaluación usando 10 cross-fold validation 14](#_Toc102468577)

[9. Limpiando los datos. Eliminando tildes 16](#_Toc102468578)

[9.1. Descripción de la implementación realizada 16](#_Toc102468579)

[9.2. Descripción de los resultados obtenidos 16](#_Toc102468580)

[9.2.1. Evaluación usando 10 cross-fold validation 16](#_Toc102468581)

[10. Generación de la matriz tfif a partir de bi-gramas 18](#_Toc102468582)

[10.1. Descripción de la implementación realizada 18](#_Toc102468583)

[10.2. Descripción de los resultados obtenidos 18](#_Toc102468584)

[10.2.1. Evaluación usando 10 cross-fold validation 18](#_Toc102468585)

[11. Generación de la matriz tfif a partir de tri-gramas 20](#_Toc102468586)

[11.1. Descripción de la implementación realizada 20](#_Toc102468587)

[11.2. Descripción de los resultados obtenidos 20](#_Toc102468588)

[11.2.1. Evaluación usando 10 cross-fold validation 20](#_Toc102468589)

[12. Conclusiones 22](#_Toc102468590)

[12.1. Impacto de las variables en los resultados 22](#_Toc102468591)

[12.2. Combinaciones de variables 22](#_Toc102468592)

[12.3. Mejor conjunto de variables 22](#_Toc102468593)

[13. Bibliografía 23](#_Toc102468594)

Índice de Ilustraciones

[Ilustración 1. Error no encuentra el fichero de entrenamiento 4](#_Toc102468595)

[Ilustración 2. Comprobación de la existencia de los datasets 4](#_Toc102468596)

[Ilustración 3. El script indica que el dataset no existe en esa ruta y finaliza 5](#_Toc102468597)

[Ilustración 4. Se añade un punto a la ruta, para acceder a la carpeta anterior a la actual 5](#_Toc102468598)

[Ilustración 5. Resultados del algoritmo base 5](#_Toc102468599)

[Ilustración 6. Función inicial de cleanData 6](#_Toc102468600)

[Ilustración 7. Accuracy primera modificación 6](#_Toc102468601)

[Ilustración 8. Codigo Eliminar signos y puntuación 7](#_Toc102468602)

[Ilustración 9. Accuracy de la segunda modificación 8](#_Toc102468603)

[Ilustración 10. Código para eliminar stopwords 9](#_Toc102468604)

[Ilustración 11. Mensaje de error. Falta la librería NLTK 9](#_Toc102468605)

[Ilustración 12. Instrucción para instalar NLTK en nuestro entorno 10](#_Toc102468606)

[Ilustración 13. Ejecutamos la instrucción en nuestro entorno 10](#_Toc102468607)

[Ilustración 14. Accuracy de la tercera modificación 11](#_Toc102468608)

[Ilustración 15.Llamada a la función de filtrado 12](#_Toc102468609)

[Ilustración 16.Función de filtrado 12](#_Toc102468610)

[Ilustración 17. Accuracy de la cuarta modificación 13](#_Toc102468611)

[Ilustración 18. Restos de enlaces web tras el cleanData 14](#_Toc102468612)

[Ilustración 19. Llamada a la función para filtrar las url 14](#_Toc102468613)

[Ilustración 20. Función para filtrar URL 14](#_Toc102468614)

[Ilustración 21. Accuracy de la quinta modificación 15](#_Toc102468615)

[Ilustración 22. Función para reemplazar los acentos 16](#_Toc102468616)

[Ilustración 23. Accuracy de la sexta modificación 17](#_Toc102468617)

[Ilustración 24. Generación de bi-gramas 18](#_Toc102468618)

[Ilustración 25. Datos ordenados por bigramas 18](#_Toc102468619)

[Ilustración 26. Accuracy de la séptima modificación 19](#_Toc102468620)

[Ilustración 27. Generación de tri-gramas 20](#_Toc102468621)

[Ilustración 28. Datos ordenados por tri-igramas 20](#_Toc102468622)

[Ilustración 29. Accuracy de la séptima modificación 21](#_Toc102468623)

Índice de Tablas

# Enunciado

Se deberán realizar un mínimo de 4 modificaciones. En concreto, el objetivo de la práctica es mejorar el conjunto de variables utilizadas en el clasificador SVM, ya sea mediante la modificación de las variables propuestas o mediante la inclusión de nuevas (generalmente mediante la concatenación para cada vector). La idea de la práctica es jugar con diferentes tipos de variables, analizar su funcionamiento y rendimiento mediante la evaluación, así como compararlas, ya sea por separado o combinadas. Algunas sugerencias de modificaciones son:

* Modificar el tamaño del diccionario para el cálculo de tf-idf, ya sea mediante la eliminación de Stop Words, mediante la selección de las XX palabras más usadas, 9 o la elección de ciertos conjuntos gramaticales (verbos y adjetivos, o nombres y adjetivos).
* Generación de la matriz de tf-idf mediante otros elementos que no sean palabras, como pueden ser bi-gramas, tri-gramas o n-gramas, así como el uso de lemas en lugar de palabras, o mediante concatenación de pares como puede ser palabra/categoría-Gramatical.
* Uso de conceptos en lugar de palabras mediante algoritmos de desambiguación semántica.
* Inclusión de elementos como los vectores de Word Embeddings.
* Inclusión de categorías gramaticales más frecuentes.
* Uso de léxicos afectivos o de palabras agresivas para identificar palabras polares (SentiSense, Spanish Emotion Lexicon, HurtLex, etc.)

Otra opción de mejora es la modificación del algoritmo de entrenamiento, pudiendo utilizar otras familias de algoritmos como pueden ser los árboles de decisión, los sistemas basados en regla, o la combinación de varios mediante sistemas de votación.

Como ya se ha expuesto, los puntos aquí planteados son sugerencias de posibles modificaciones, pero no es necesario restringirse a estos. Se pueden incluir cualesquiera originales, o ya vistos en la bibliografía o estado del arte de otros artículos científicos en el área de procesamiento del lenguaje o clasificación de texto.

Por último, se deberá realizar una memoria en la que se detalle el diseño de las modificaciones realizadas sobre el clasificador básico propuesto, su evaluación, análisis de resultados y conclusiones extraídas. En concreto la memoria deberá contener:

* Descripción de la implementación realizada, problemas encontrados, soluciones aportadas, etc.
* Descripción de resultados obtenidos para diferentes variables, y de todas las alternativas/combinaciones realizadas, con un análisis exhaustivo de los mismos en base a las distintas métricas utilizadas.
* Conclusiones que se puedan obtener a partir del análisis de resultados: cómo afectan las variables a los resultados, sus combinaciones, mejor conjunto de variables y por qué, etc.

# Consideraciones iniciales

Dado que los datasets de entrenamiento y test proporcionados para la práctica superaban con las 6000 y 4000 líneas respectivamente, se ha decidido, con el fin de acelerar las pruebas intermedias durante el desarrollo de las modificaciones del algoritmo, que se reducirán estos datasets a un tamaño de 100 líneas cada uno de las secciones en castellano. Una vez se hayan finalizado las modificaciones, y se tenga un prototipo de algoritmo viable, se realizará una ejecución con los datasets originales.

# Algoritmo base

## Descripción de la implementación realizada

Se realizará en primer lugar una ejecución del algoritmo base sin modificaciones a modo de referencia para el resto de las modificaciones que se implementen.

### Problemas encontrados

#### Problema 1

##### Descripción del problema

Cuando tratamos de ejecutar el script por primera vez, nos devuelve un mensaje de error, el cual se muestra en la ilustración 1.

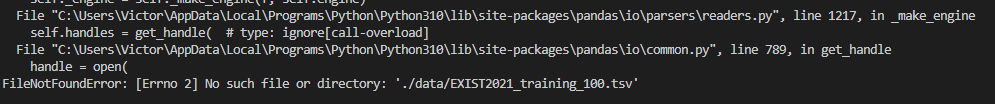


Ilustración 1. Error no encuentra el fichero de entrenamiento

##### Solución

Con el fin de realizar una comprobación de la existencia de los archivos de entrenamiento antes entrar a la función processEXISTTraining(), se ha optado por añadir una condición if que compruebe la existencia de los ficheros en la ruta proporcionada.

Texto

Descripción generada automáticamente

Ilustración 2. Comprobación de la existencia de los datasets

Si volvemos a ejecutar el script nos devuelve los siguiente:

Texto

Descripción generada automáticamente

Ilustración 3. El script indica que el dataset no existe en esa ruta y finaliza

En base a esto se interpreta que el script no esta consiguiendo acceder a la carpeta en la que se encuentran los datasets en la carpeta *data* del nivel anterior. Por ello, en la ruta descrita para pathTraining y pathTest se añade un punto en la parte anterior a la ruta ya descrita para que retroceda al nivel de jerarquía anterior. Por lo tanto, el código quedaría de la forma siguiente.

Texto

Descripción generada automáticamente

Ilustración 4. Se añade un punto a la ruta, para acceder a la carpeta anterior a la actual

## Descripción de los resultados obtenidos

### Evaluación usando 10 cross-fold validation

Tras ejecutar el script obtenemos los resultados mostrados en la ilustración 5.

Calendario

Descripción generada automáticamente

Ilustración 5. Resultados del algoritmo base

Si observamos los resultados del algoritmo base tenemos que tiene una precisión o accuracy del 0.66.

# Limpiando los datos. Todo a minúsculas

Con el fin de homogeneizar los datos obtenidos a la hora de tokenizar las palabras obtenidas de los diferentes tweets, en esta primera aproximación nos centraremos en poner en minúsculas todas las palabras obtenidas tras el proceso de tokenización.

## Descripción de la implementación realizada

Para realizar esto, creamos una nueva función dentro del script a la que llamaremos cleanData(). Esta función podrá servirnos para futuras modificaciones del script a la hora de limpiar los datos obtenidos del algoritmo. De momento, para reducir a minúsculas se utilizará el siguiente código.

Texto

Descripción generada automáticamente

Ilustración 6. Función inicial de cleanData

## Descripción de los resultados obtenidos

### Evaluación usando 10 cross-fold validation

Si ejecutamos nuevamente el script con la modificación, obtenemos una leve mejoría en el accuaracy del script, pasando de una precisión del 0.66 al 0.68, como podemos observar en la ilustración siguiente.

Calendario

Descripción generada automáticamente

Ilustración 7. Accuracy primera modificación

# Limpiando los datos. Eliminar signos de puntuación

El siguiente paso para limpiar los datos obtenidos al tokenizar los mensajes del dataset, es eliminar los signos de puntuación y espacios en blanco.

## Descripción de la implementación realizada

Para realizar esto, dentro de la función que creamos anteriormente llamada cleanData() añadimos una nueva sección destinada a la eliminación de símbolos de puntuación y espacios en blanco. Como se muestra en la imagen a continuación.

Texto

Descripción generada automáticamente con confianza media

Ilustración . Codigo Eliminar signos y puntuación

## Descripción de los resultados obtenidos

### Evaluación usando 10 cross-fold validation

Si ejecutamos nuevamente el script con la segunda modificación, obtenemos una leve mejoría en el accuracy del script, pasando de una precisión del 0.68 al 0.70, como podemos observar en la ilustración siguiente.

Calendario

Descripción generada automáticamente con confianza media

Ilustración . Accuracy de la segunda modificación

# Limpiando los datos. Eliminando Stopwords

El siguiente paso para limpiar los datos obtenidos al tokenizar los mensajes del dataset, es eliminar las stopwords o palabras vacías, estas son palabras que carecen de sentido propio cuando se escriben solas o sin la palabra clave o keyword.[1]

## Descripción de la implementación realizada

Para realizar esto, dentro de la función que creamos anteriormente llamada cleanData() añadimos una nueva sección destinada a la eliminación de stopwords. Para ello necesitamos importar la librería nltk con herramientas de gestión del lenguaje.

Para ello utilizamos el siguiente código de la ilustración siguiente.

Texto

Descripción generada automáticamente

Ilustración . Código para eliminar stopwords

Esto permitirá en primer lugar filtrar todas las palabras que no aparten información que están en la lista de stopwords de la librería nltk. Y en segundo lugar eliminamos todos los palabras vacías o ‘’ que se hayan quedado en la lista despues de la limpieza y normalización de los datos.

### Problemas encontrados

#### NLTK no disponible

##### Descripción del problema

Dado que usaremos la librería NLTK (Natural Lenguage Toolkit) [2] necesitamos no solo importarla en nuestro script, si no que también debemos instalarla en nuestro entorno de desarrollo.

Es por ello que cuando tratamos de ejecutar el script nos devuelve el siguiente mensaje de error que se muestra en la ilustración 10.

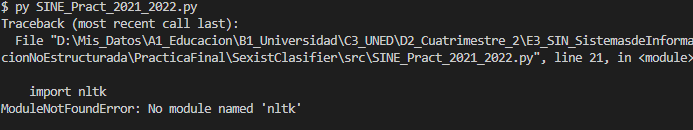


Ilustración . Mensaje de error. Falta la librería NLTK

##### Solución

Para ello si acudimos a la documentación oficial de la librería en su parte de instalación [2]encontraremos las instrucciones para instalar la librería en nuestro entorno[3].

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración . Instrucción para instalar NLTK en nuestro entorno

Ejecutamos la instrucción en nuestra consola, y ya la tendríamos instalada.

Texto

Descripción generada automáticamente

Ilustración . Ejecutamos la instrucción en nuestro entorno

## Descripción de los resultados obtenidos

### Evaluación usando 10 cross-fold validation

Si ejecutamos nuevamente el script con la segunda modificación, obtenemos un leve empeoramiento en el accuracy del script, pasando de una precisión del 0.7 al 0.66, como podemos observar en la ilustración siguiente. Sin embargo, la desviación estándar se ha reducido de un 0.10 a un 0.06.

Calendario

Descripción generada automáticamente

Ilustración 14. Accuracy de la tercera modificación

# Limpiando los datos. Eliminando nombres de usuarios

Con el fin de evitar sesgos a determinados usuarios, se ha obtado por eliminar toda referencia a nombres de usuarios de los comentarios utilizados.

## Descripción de la implementación realizada

Para realizar esto, buscaremos todas las palabras que empiecen por un @ y las eliminaremos de la lista dentro de nuestra función cleanData(). Para ello, usamos el código que mostramos a continuación.



Ilustración .Llamada a la función de filtrado

Texto

Descripción generada automáticamente

Ilustración .Función de filtrado

Para determinar si una palabra es un nombre de usuario buscamos si contiene el símbolo @.

## Descripción de los resultados obtenidos

### Evaluación usando 10 cross-fold validation

Si ejecutamos nuevamente el script con la segunda modificación, obtenemos un leve empeoramiento en el accuracy del script, pasando de una precisión del 0.66 al 0.65, como podemos observar en la ilustración siguiente. Sin embargo, la desviación estándar se ha mantenido en un 0.06.

Calendario

Descripción generada automáticamente

Ilustración . Accuracy de la cuarta modificación

# Limpiando los datos. Eliminando enlaces

Si observamos los datos despues del proceso de tokenización y despues de pasar por cleanData() como se muestra en la ilustración siguiente, observamos que hay enlaces que no aportan información relevante.

Imagen en blanco y negro

Descripción generada automáticamente con confianza media

Ilustración . Restos de enlaces web tras el cleanData

Por ello, se procederá en este apartado a eliminar estos enlaces.

## Descripción de la implementación realizada

Para ello, buscaremos todas las palabras que empiecen o contengan https. A continuación se muestra el código utilizado para filtrar las URL.



Ilustración . Llamada a la función para filtrar las url

Texto

Descripción generada automáticamente

Ilustración . Función para filtrar URL

## Descripción de los resultados obtenidos

### Evaluación usando 10 cross-fold validation

Si ejecutamos nuevamente el script con la quinta modificación, obtenemos que el accuracy del script no vario, permanece a la precisión de 0.65, como podemos observar en la ilustración siguiente. Y la desviación estándar se ha mantenido en un 0.06.

Calendario

Descripción generada automáticamente

Ilustración 21. Accuracy de la quinta modificación

# Limpiando los datos. Eliminando tildes

Para normalizar aun mas los datos obtenidos, vamos a eliminar todas las tildes, sustituyendo esos caracteres por sus equivalentes sin tildes.

## Descripción de la implementación realizada

Para ello, buscaremos todas los caracteres con acentos sustituyéndolos por sus equivalentes sin acentos como se muestra en el código mostrado en la siguiente ilustración.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Ilustración . Función para reemplazar los acentos

## Descripción de los resultados obtenidos

### Evaluación usando 10 cross-fold validation

Si ejecutamos nuevamente el script con la sexta modificación, obtenemos que el accuracy del script se incrementó del 0.65 anterior a un 0.66, como podemos observar en la ilustración siguiente. Y la desviación estándar se ha reducido de un 0.06 a un 0.05.

Calendario

Descripción generada automáticamente

Ilustración 23. Accuracy de la sexta modificación

# Generación de la matriz tfif a partir de bi-gramas

En este apartado probaremos a generar la matriz de tf-idf mediante otros elementos que no sean palabras, como pueden ser los bi-gramas, es decir, pares de palabras.

## Descripción de la implementación realizada

Para ello, despues de haber limpiado los datos con la función que creamos en apartados anteriores cleanData() usaremos la función bigrams() de la librería nltk para formar los bigramas como se muestra en el código de la ilustración siguiente.



Ilustración . Generación de bi-gramas

De esta forma los datos quedarían colocados por pares de la siguiente forma:

Texto

Descripción generada automáticamente

Ilustración . Datos ordenados por bigramas

## Descripción de los resultados obtenidos

### Evaluación usando 10 cross-fold validation

Si ejecutamos nuevamente el script con esta modificación, obtenemos que el accuracy del script se reduce del 0.66 anterior a un 0.57, como podemos observar en la ilustración siguiente. Y la desviación estándar se ha reducido de un 0.05 a un 0.03.

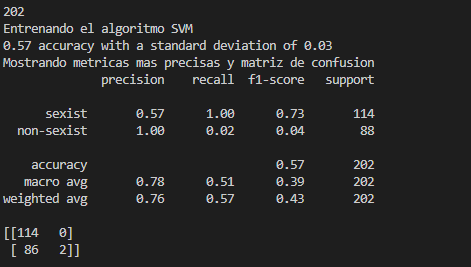


Ilustración 26. Accuracy de la séptima modificación

# Generación de la matriz tfif a partir de tri-gramas

En este apartado probaremos a generar la matriz de tf-idf mediante tri-gramas, es decir, conjuntos de tres palabras.

## Descripción de la implementación realizada

Para ello, despues de haber limpiado los datos con la función que creamos en apartados anteriores cleanData() usaremos la función trigrams() de la librería nltk para formar los trigramas como se muestra en el código de la ilustración siguiente.



Ilustración 27. Generación de tri-gramas

De esta forma los datos quedarían colocados por conjuntos de tres palabras de la siguiente forma:

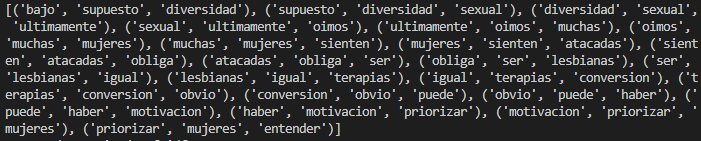


Ilustración 28. Datos ordenados por tri-igramas

## Descripción de los resultados obtenidos

### Evaluación usando 10 cross-fold validation

Si ejecutamos nuevamente el script con esta modificación, obtenemos que el accuracy del script no cambia con respecto al uso de bigramas, como podemos observar en la ilustración siguiente. Y la desviación estándar también se mantiene a un 0.03.

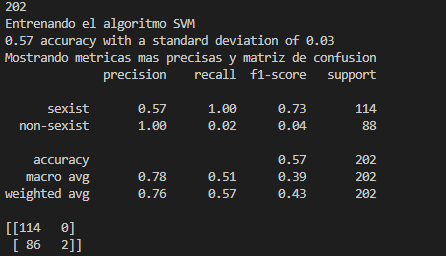


Ilustración 29. Accuracy de la séptima modificación

# Conclusiones

## Impacto de las variables en los resultados

## Combinaciones de variables

## Mejor conjunto de variables

# Bibliografía

[1] “¿Qué son las Stop Words y para qué sirven? – BlackBeast.” https://blackbeast.pro/diccionario/stop-words/ (accessed May 02, 2022).

[2] “NLTK :: Natural Language Toolkit.” https://www.nltk.org/index.html (accessed May 02, 2022).

[3] “nltk · PyPI.” https://pypi.org/project/nltk/ (accessed May 02, 2022).