UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

Departamento de sistemas informáticos y computación

Author profiling

Trabajo Aplicaciones de la Lingüística Computacional

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autores
Andrés Marín Galán
Francisco Javier Gil-Terrón Rodríguez

2021 - 2022

Resumen

La tarea de *author profiling* o perfil de autor consiste en analizar una serie de textos dados de un autor concreto con el fin de descubrir ciertas características de este mismo basadas en sus características estilísticas, contenido del mensaje, etc.

En este proyecto se llevará a cabo experimentación de *author profiling* para determinar si un autor es o no irónico a partir de un conjunto de tweets publicados por el mismo autor.

En este trabajo se expondrán qué técnicas se han utilizado para llevar a cabo el perfil de autor, desde los mecanismos de pre-procesado y codificación de la información, hasta los modelos y clasificadores empleados para determinar si los autores están siendo irónicos o no, apoyados con materiales gráficos en forma de tablas e imágenes con los resultados obtenidos.

Tabla de contenidos

1. Estructura y librerías

En primer lugar, se expondrá la estructura que se seguirá en los siguientes capítulos a este mismo, así como cuáles han sido las tecnologías y paquetes empleados para la realización del trabajo.

En el siguiente capítulo, experimentos y modelos, se explicará con detalle cuáles han sido las técnicas de preprocesado aplicadas a los conjuntos de datos previamente al comienzo de la propia experimentación y que han sido aplicadas de manera generalizada para la realización de todas las pruebas. Tras esto, se profundizará en las estrategias de entrenamiento que han sido probadas y se comentarán aquellas que fueron planteadas pero desestimadas antes de llegar a experimentar con ellas.

La siguiente sección, discusión de resultados, presentará en formato de gráficas y tablas los resultados obtenidos a través de los modelos previamente citados y se comentará cómo se ha llegado a estos mismos,

Finalmente se concluirá con un breve capítulo en el que se inferirá de manera resumida todo aquello que se ha aprendido y trabajado en el presente trabajo.

Por lo que respecta a las tecnologías utilizadas, el trabajo ha sido desarrollado sobre la plataforma de Google Colaboratory en un entorno equipado con GPU, siendo las librerías con las que se ha desarrollado el proyecto, principalmente, Numpy y Pytorch para las estructuras de datos y transformaciones de datos sobre GPU, Scikit-learn para la importación de clasificadores y la librería transformes para la inclusión de modelos pre-entrenados.

2. Experimentos y modelos

Antes de comenzar a hablar de la propia experimentación se comentará qué preprocesado de ha utilizado adicionalmente al que ya existe de base en el conjunto de datos. Pese a que los datos ya se encontraran tokenizados en un momento inicial, decidimos aplicar un segundo proceso de tokenización a los tweets probando con distintos tokenizadores inspirados en la segunda práctica de esta misma asignatura.

En cualquier caso, los datos que se obtenían tras este proceso diferían muy ligeramente respecto del conjunto inicial, independientemente del tokenizador utilizado, por lo que no se dedicó un esfuerzo excesivo en esta etapa.

El primero de los experimentos que se tuvo en consideración fue el de utilizar un modelo de lenguaje pre-entrenado con el fin de transformar en vectores los tweets de cada usuario para posteriormente emplear un clasificador con los tweets vectorizados.

En un primer caso, se optó por probar con un modelo Bert pre-entrenado para esta tarea en concreto, la clasificación de ironía (o no) de usuarios de Twitter a partir de sus tweets. Este modelo se obtuvo del repositorio de modelos "*Hugging Face*" pero se desestimó esta opción debido a que, por la complejidad de este modelo concreto, la vectorización era demasiado lenta como para llevar a cabo las pruebas que se querían realizar.

Tras esto nos propusimos emplear una versión más ligera de un modelo Bert pre-entrenado por lo que recurrimos a DistilBert. Puesto que, al contrario de lo que ocurría en el caso anterior, DistilBert vectorizó los tweets en un tiempo bastante aceptable, decidimos emplear también el modelo Bert estándar además de su versión ligera que acabábamos de probar.

En ambos casos, el resultado de la vectorización de los tweets con Bert y DistilBert daba, para cada usuario, doscientos vectores de características (uno por cada tweet), que concatenamos antes de pasar los datos a los clasificadores de manera que finalmente, cada usuario tuviera un vector de características.

En otro orden de ideas, más allá de emplear modelos pre-entrenados para vectorizar los tweets, tratamos de emplear la tecnología Word2Vec para transformar los tweets en vectores, de manera similar a cómo habíamos hecho con Bert y DistilBert.

Una vez transformados los datos en vectores, ya fuera con Bert, DistilBert o Word2Vec, se recurrió a distintos clasificadores presentes en Scikit-learn.

Los clasificadores que fueron empleados son: clasificación con vectores soporte (SVC) y clasificación lineal con vectores soporte (LinearSVC), clasificación lineal con descenso por gradiente estocástico (SGD) y clasificación por refuerzo de gradiente (GradientBoosting). Otros clasificadores fueron desestimados debido a que sus resultados eran considerablemente inferiores que el resto, como el algoritmo de K-vecinos.

3. Discusión de resultados

Con el fin de evaluar los modelos expuestos en el punto anterior ha sido empleada la técnica de validación cruzada con un tamaño de diez particiones. A continuación, se mostrarán los resultados obtenidos con los distintos clasificadores empleados, pero únicamente con el modelo que usaba Bert, ya que este modelo obtenía los mejores resultados y superaba al resto de modelos (DistilBert y Word2Vec).

En los resultados se mostrará, para cada una de las particiones realizadas durante la validación cruzada, su precisión y su intervalo de confianza con un 95% de precisión (como se podrá ver, estos intervalos de confianza serán bastante holgados debido a la cantidad de datos de que se disponía).

3.1 SVC

Partición	Precisión	Intervalo de confianza
0	0.881	[0.784 - 0.978]
1	0.929	[0.851 - 1]
2	0.857	[0.752 - 0.962]
3	0.881	[0.784 - 0.978]
4	0.905	[0.816 - 0.993]
5	0.929	[0.851 - 1]
6	0.905	[0.816 - 0.993]
7	0.881	[0.784 - 0.978]
8	0.952	[0.888 - 1]
9	0.905	[0.816 - 0.993]

Tabla 1 – Tabla resultados SVC

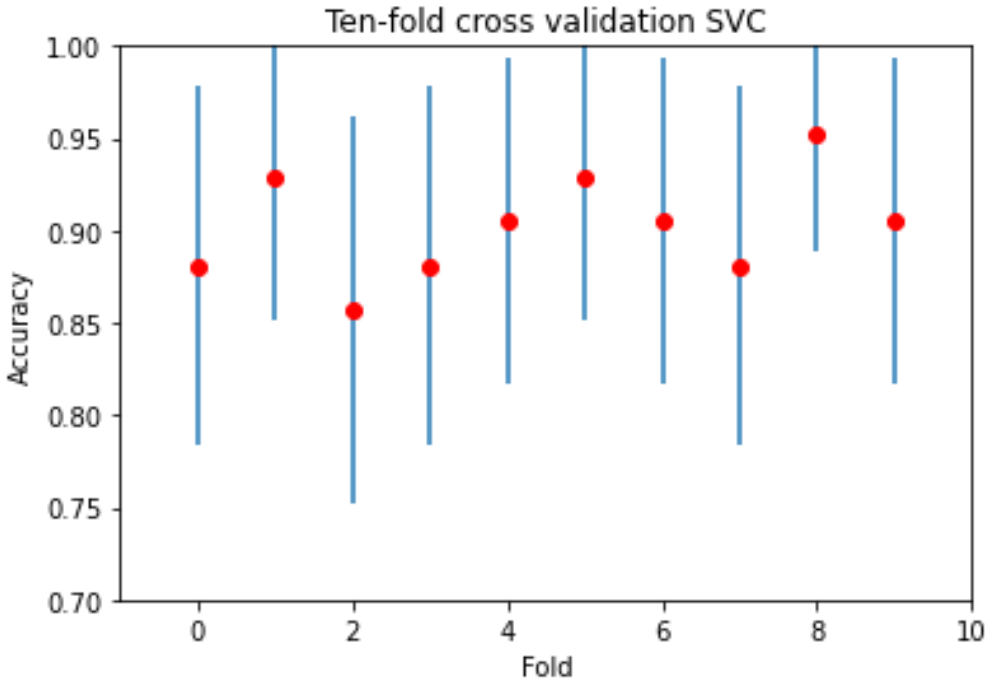


Figura 2 – Gráfica resultados SVC

3.2 LinearSVC

Partición	Precisión	Intervalo de confianza
0	0.881	[0.784 - 0.978]
1	0.929	[0.851 - 1]
2	0.810	[0.691 - 0.928]
3	0.857	[0.752 - 0.962]
4	0.929	[0.851 - 1]
5	0.929	[0.851 - 1]
6	0.929	[0.851 - 1]
7	0.881	[0.784 - 0.978]
8	0.952	[0.888 - 1]
9	0.905	[0.816 - 0.993]

Tabla 2 – Tabla resultados SVC-Linear

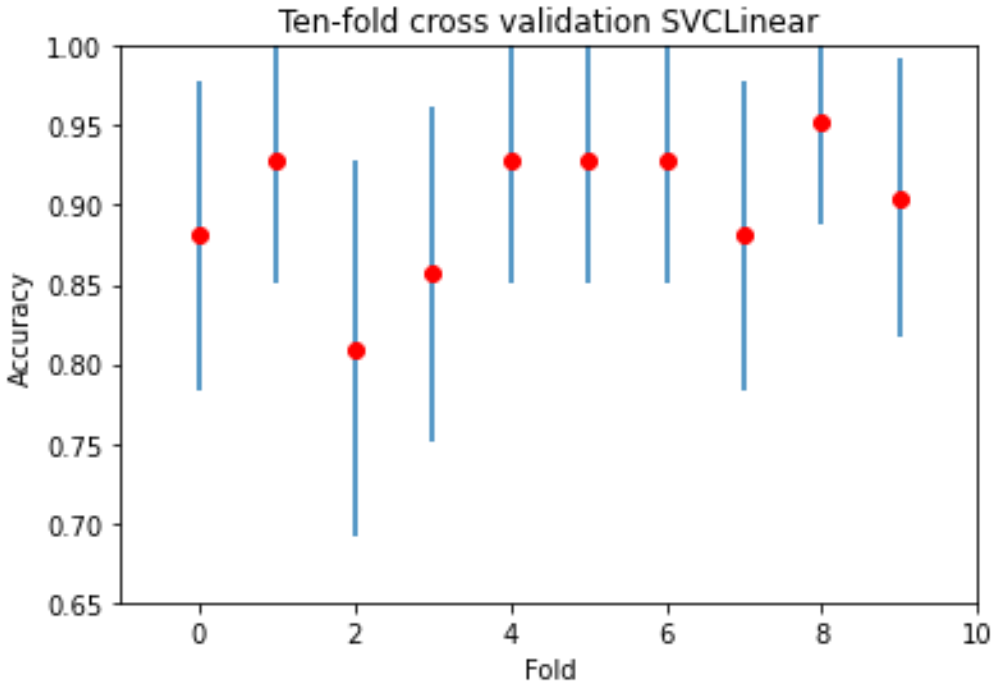
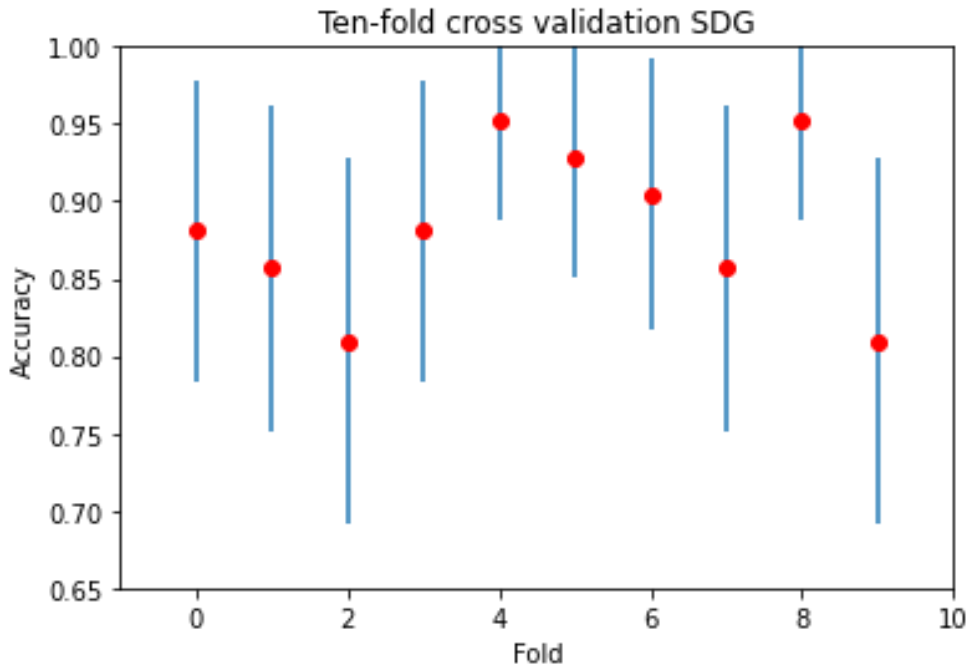


Figura 2 – Gráfica resultados SVC-Linear

3.3 SDG

Partición	Precisión	Intervalo de confianza
0	0.881	[0.784 - 0.978]
1	0.857	[0.752 - 0.962]
2	0.810	[0.691 - 0.928]
3	0.881	[0.784 - 0.978]
4	0.952	[0.888 - 1]
5	0.929	[0.851 - 1]
6	0.905	[0.816 - 0.993]
7	0.857	[0.752 - 0.962]
8	0.952	[0.888 - 1]
9	0.810	[0.691 - 0.928]



Así podemos estimar que:

SVC: es el mejor clasificador pues obtiene muy buenos valores de accuracy aun en la configuración por defecto.

LinearSVC: obtiene valores altos de accuracy cercano al SVC, pero peores.

Regresión Logística: el tiempo para su entrenamiento era desproporcionado por lo que fue descartado.

GradientBoosting: el tiempo para su entrenamiento era desproporcionado por lo que fue descartado.

SGD: se obtienen buenos valores de accuracy, pero son bajos en comparación a otros modelos

K-vecinos: se han obtenido resultados pésimos aun variando el número de vecinos por lo que se ha descartado este modelo.

4. Conclusión

Como hemos visto el modelo SVC con vectores soporte funciona francamente bien para la tarea de detección de ironía con un preprocesado usando BERT, para convertir cada tweet en un vector. Se han probado los modelos preentrenados como BERT o Word2vect y se han ajustado los parámetros de los modelos que mejor funcionaban.

Obteniendo valores de error menores a un 10%, por lo que se ha construido un buen modelo de detección de ironía.

Podemos concluir que el empleo de modelos preentrenados es sustancialmente mejor, además del uso de los vectores soporte al menos para el conjunto de datos sobre el que se ha trabajado.