Clasificación de textos con modelos preentrenados

Toni Blasco Calafat
Universitat Politècnica de València
MIARFID

February 17, 2023

1 Introduction

Este trabajo realizado sobre el marco de la asignatura de Redes Neuronales propone una experimentación mediante modelos preentrenados la realización de la tarea de clasificación de texto. La clasificación de textos, consiste en catalogar textos en función de su contenido, es decir, realizar un análisis de las palabras para decidir qué tipo de texto es el que se está identificando. Por ejemplo, en función del contenido podríamos averiguar si se trata de una noticia deportiva, política o económica.

El conjunto de datos usado para la realización del trabajo es GLUE[1], the General Language Understanding Evaluation benchmark. Es una colección de recursos para formar, evaluar y analizar sistemas de comprensión del lenguaje natural. Concretamente, se ha escogido la parte cola, El Corpus de Aceptabilidad Lingüística, consiste en juicios de aceptabilidad del inglés extraídos de libros y artículos de revistas sobre teoría lingüística. Cada ejemplo es una secuencia de palabras en la que se indica si se trata de una frase gramatical inglesa. Finalmente, se remarca el hecho de que el modelo pre-entrenado se ha descargado de HuggingFace[2] y se ha ejecutado íntegramente en GoogleColab.

2 Carga del dataset y método de evaluación

Para la carga del dataset se ha hecho uso de la función *load_dataset* perteneciente a la librería Datasets[3], dado que GLUE ya forma parte de la plataforma es sencillo el uso de aplicar esta función.

Respecto a la función de evaluación se ha hecho uso de la función *load* de la librería Evaluate[4], esta función permite importar diferentes métricas de evaluación, para el desarrollo de la tarea, sin embargo, al tratarse de GLUE, dataset frecuentado, contiene sus propias métricas y por tanto simplemente hay que marcar el dataset en los parámetros. En este caso al tratarse de GLUE MNLI para un caso de clasificación binario simple, se hace uso del matthews_correlation que mide el grado de asociación entre dos variables binarias.

3 Preprocesamiento de los datos

Para poder realizar el entrenamiento de los diferentes textos, se debe realizar primeramente un preprocesamiento de los datos, es por ello que se debe tokenizar primeramente se instancia el tokenizador con la herramienta *AutoTokenizer.from_pretrained*. Esta permite tokenizar los textos y ponerlo en un formato que el modelo espera, así como generar las otras entradas que el modelo requiere. Para ello, se le pasa como parámetro el modelo preentrenado para que pueda ajustar la tokenización al modelo pasado como parámetro, en este caso se hace uso del modelo DistilBERT[5]. Este modelo es un transformer, más pequeño y rápido que BERT, que se preentrenó en el mismo corpus de forma autosupervisada, utilizando el modelo base de BERT como profesor. Esto significa que se preentrenó sólo con los textos en bruto, sin que los humanos los etiquetaran de ninguna manera (por eso puede utilizar muchos datos disponibles públicamente) con un proceso automático para generar entradas y etiquetas a partir de esos textos utilizando el modelo base de BERT. Posteriormente, se le pasa todo el texto al tokenizador para que este lo trate, para ello se le ha añadido al tokenizador los argumentos

truncation=True y padding='longest. Asegurando que una entrada más larga de lo que el modelo seleccionado puede manejar se truncará a la longitud máxima aceptada por el modelo, y todas las entradas se rellenarán a la longitud máxima de entrada para darnos una sola matriz de entrada.

4 Fine-tunning del modelo

Para la realización del Fine-tunning del DistilBERT se ha hecho uso como en el caso anterior de una herramienta de HuggingFace de la clase Auto. Estas clases sirven para recuperar automáticamente el modelo relevante dado el nombre/ruta a los pesos/configuración/vocabulario preentrenados. Para el modelado, se ha hecho uso de la clase *TFAutoModelForSequenceClassification*, esta clase carga los modelos preentrenados de tareas sequencia a sequencia como en el caso que estamos realizando. En este caso se añade la configuración para DistilBert, es decir, el modelo elegido primeramente, el uso de la cross entropy en segundo lugar para la clasificación de texto.

Posteriormente se debe compilar el modelo para ello, se ha hecho uso de la clase *compile*, indicándole previamente diferentes parámetros iniciales al modelo como un optimizador AdamW, junto un learning-rate decay, un número de epochs igual a 3 y batches per epoch del conjunto de datos entre el entrenamiento entre 16 que es el tamaño de size elegido.

Finalmente lo último que se define es el calculo de métricas a través de las predicciones. Para ello se hace una función callback llamada a cada epoch, donde se toma el argmax de las etiquetas predichas y se hace uso de una función llamada *KerasMetricCallback* que es el que se encargará de calcular las métricas para cada epoch.

5 Experimentación y resultados

En esta parte del trabajo se procede a realizar diferentes ejecuciones sobre el modelo, viendo así que parámetros se pueden variar y intentar maximizar la métrica.

Se han realizado las diferentes experimentaciones con el objetivo de ver como varía la métrica para la tarea.

Experimento	**Batch size**	**epochs**	**optimizador**	**LR**	**matthews correlation**
A	16	3	AdamWeightDecay	0.001	0.5285
B	16	5	AdamW	1e-7	0.5169
C	16	3	AdamW	2e-5	0.5290
D	16	10	AdamW	1e-3	0.5527

Table 1: Resultados

A la vista de estos resultados se ha optado por escoger la experimentación D, ya que se observa como a mayor número de epochs obtiene mejores resultados junto el optimizador y el LR escogido.

6 Conclusión

A modo de conclusión del trabajo se quieren remarcar diferentes aspectos. El primero es sobre la elección del trabajo ya que se ha optado por realizar una tarea completamente diferente a la realizada en prácticas utilizando otras técnicas como el uso de transformer, y el uso de la plataforma HuggingFace. Esta tiene muchos modelos entrenados así como una gran cantidad de datasets todos ellos con diferentes variaciones o especializaciones como en el caso del GLEU que permite realizar diferentes técnicas en un solo dataset. Esta forma de realizar programación de finetuning adaptación de modelos y descarga subida y testeo con la página es novedosa en el ámbito personal y ha resultado agradable. Por otra parte, se remarca que la experimentación realizada no ha sido como en el caso de la práctica donde se buscaba obtener el mayor porcentaje de accuracy posible para el dataset, si no que las diferentes técnicas aplicadas y la observación de las diferentes herramientas que HuggingFace propone entre ellas las diferentes librerías como Dataset, auto clases, transformers. Finalmente, se ha observado

References

[1] https://huggingface.co/datasets/glue

[2] https://huggingface.co

[3]https://huggingface.co/docs/datasets/load_hub

[4] https://huggingface.co/docs/evaluate/choosing_a_metric

[5] https://huggingface.co/distilbert-base-uncased