Clasificación supervisada. Verbal Autopsy

Igor Isasi

Tarea

El objetivo de este trabajo es realizar una clasificación supervisada usando diferentes técnicas de vectorización y diferentes clasificadores. Como técnicas de vectorización he utilizado *Document Embedding* y *TF-IDF*, las cuales se van a explicar más adelante, y como clasificadores, he utilizado *DecisionTree* como Baseline y *MultiLayerPerceptron* (MLP) como clasificador definitivo.

Para la clasificación, son importantes los valores que se utilizan para cada parámetro a la hora de lograr buenos resultados, y es por ello que he hecho un barrido de parámetros para intentar encontrar valores que mejoren los resultados.

He aprovechado parte del trabajo de Clustering para esta tarea, sobre todo el preprocesamiento y el Document embedding, aunque he realizado varias modificaciones.

Dataset: Verbal Autopsy

El conjunto de datos que he elegido es sobre Verbal Autopsy, método utilizado para intentar determinar la causa de muerte de un fallecido. Para intentar averiguar la causa de muerte, se entrevistan a personas cercanas al fallecido recogiendo así la sintomatología o factores importantes que han podido influir en dicha muerte.

Preproceso

El preproceso de esta tarea ha sido muy similar al preproceso de la tarea del Clustering (limpieza de texto, pasarlo todo a minúsculas, quitar las stopwords...) pero hay 2 diferencias.

La primera es que he quitado palabras comunes que no tenían relevancia a la hora de determinar la causa de muerte. A continuación se muestran las palabras más frecuentes de la clase 'Neoplasms' antes y después de la limpieza.

Figure 1. Wordclouds de la clase 'Neoplasms' antes (izquierda) y después (derecha) de la limpieza

Podemos ver como antes de la limpieza hay palabras que no ayudan a la hora de la clasificación como por ejemplo *wa*, *doctor* y *day*. Por otro lado, en el Wordcloud formado después de la limpieza, hay palabras muy significativas que resaltan como *cancer*, *lung* y *chemotherapy*, palabras que ya estaban antes de la limpieza pero que debido a la contaminación que producían algunas palabras(wa, doctor, day, etc) no destacaban tanto.

Vectorización

Para la vectorización he utilizado tanto doc-embedding (utilizado en la tarea anterior de Clustering) como TF-IDF.

- **Doc-embedding**: mediante esta técnica, se asigna un vector por instancia (el tamaño del vector es un parámetro alterable) y aquellos vectores que más cerca se encuentren entre sí serán los que tienen más en común. En mi caso concreto, la data que estamos vectorizando es la información acerca de la muerte de los fallecidos, por lo que cuanto más cerca esté un vector de otro, supondremos que la causa de muerte será más similar.

Para implementar el document embedding los parámetros en los que me he centrado son 3: vector_size, min_count y epochs.

- **TF-IDF**: método que asigna un valor a cada palabra de un documento dependiendo de como de relevante es.
  Solo he utilizado un parámetro, min_df=3, que determina el número de apariciones mínimas de una palabra para que se tenga en cuenta. No he utilizado más parametros porque muchos hacen referencia al preproceso, y nosotros lo hemos hecho previamente.

Clasificador 1: Baseline

Antes de clasificarlos definitivamente, he utilizado el clasificador DecisionTreeClassifier para tener una referencia de que resultados nos proporciona un clasificador ciertamente simple. De esta forma al hacer la clasificación definitiva, podremos compararlos.

Para hacer el Baseline he utilizado Hold-Out con proporción de 70% train y 30% test y lo he hecho con repetición (5 iteraciones con train y test random) para que el resultado sea más fiel a la realidad (más aleatorio).

El Baseline lo he ejecutado con los datos post doc-embedding (doc2vec) y post TF-IDF, y los resultados han sido mejores para la vectorización TF-IDF, aunque también es verdad que ha tardado más.

Vectorización	Media del accuracy
doc2vec	0.234293
TF-IDF	0.415355

Table 1. Los métodos de vectorización y la media del accuracy que ha dado cada uno en el Baseline

La diferencia de accuracy ha sido de 0.181062 a favor de TF-IDF, una diferencia considerable teniendo en cuenta los resultados bajos que logramos con este dataset que tiene tantas clases.

Clasificador 2: MLP

El clasificador definitivo utilizado para esta tarea ha sido MultiLayerPerceptron (MLP), clasificador basado en redes neuronales.

Antes de hacer la clasificación he hecho un barrido de parámetros tanto para los datos post Doc-embedding como para los datos post TF-IDF. Los parámetros en los que me he centrado son: *hidden_layer_sizes* (hls), *activation* (act), *learning_rate* (lr) y *learning_rate_init* (lri). Además, he fijado el parámetro *max_iter* en el valor 500. Esto quiere decir que el clasificador va a iterar hasta que llegue a la convergencia o se llegue al máximo de iteraciones, en este caso, 500. Como *Verbal Autopsy* es un dataset complejo, no se llega a la convergencia y se cumplen las 500 iteraciones.

Después de probar diferentes valores para los parámetros mencionados, he hecho la clasificación final utilizando el modelo del clasificador con nuestros parámetros óptimos.

- **Doc-embedding**: los parámetros que mejor accuracy han logrado han sido los siguientes: **hls=(150, 150), act=logistic, lr=adaptive y lri=0.001000**, consiguiendo un accuracy de **0.396673**.
- **TF-IDF**: los parámetros que mejor accuracy han logrado: **hls=(300,), act=relu, lr=invscalingy lri=0.000500**, logrando un accuracy de **0.525912**.

Como podemos apreciar, en la prueba de parámetros TF-IDF es claramente superior a Doc-embedding.

MLP: resultados finales

Después de guardar nuestros modelos óptimos según nuestro barrido de parámetros, cargamos estos dos modelos y procedemos con la clasificación final. Este proceso no llevará mucho tiempo, ya que nuestros modelos ya están entrenados.

Clasificación final Doc-embedding vs TF-IDF

Vectorización	Media del accuracy
doc2vec	0.405374
TF-IDF	0.858989

Table 2. Los métodos de vectorización y la media del accuracy que ha dado cada uno en la clasificación final mediante MLP

A continuación se encuentran los gráficos ROC de las clases *Birth asphyxia* y *Other cardiovascular diseases* contra las otras clases tanto para Doc-embedding como para TF-IDF. También se muestra el *random-guesser* (TPR=FPR) en los gráficos como referencia. Cuanto más lejos se encuentren las funciones del random-guesser, mejor será el resultado.
En cada gráfico se aprecian 5 funciones; una por cada iteración del hold-out.

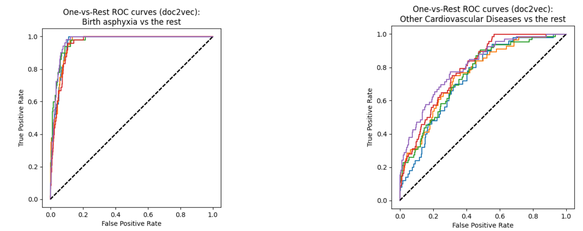

Figure 2. Gráficos ROC para la clase Birth-asphyxia (izquierda) y Other cardiovascular diseases (derecha) contra las demás con vectorización Doc-embedding

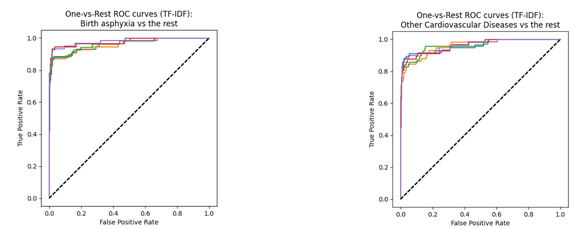

Figure 3. Gráficos ROC para la clase Birth-asphyxia (izquierda) y Other cardiovascular diseases (derecha) contra las demás con vectorización TF-IDF

Conclusiones

Hemos podido ver tanto en el accuracy como en los gráficos ROC de estas dos clases elegidas al azar como claramente TF-IDF consigue mejores resultados que Doc-embedding.

Sin embargo, hay que recordar que en cada ejecución los resultados varían ya que al separar train y test al hacer hold-out repetido, hay un factor de aleatoridad que influye en los resultados. Aún así, parece evidente que la vectorización TF-IDF es más apropiada para este conjunto de datos, ya que la diferencia entre ambos es muy grande.

Para finalizar, como era de esperar, los resultados del clasificador final (MLP) han sido mucho mejores que los del Baseline (DecisionTree), ya que el primero es mucho más complejo que el segundo. Dicha diferencia de complejidad se ha visto reflejada claramente en la duración de cada clasificador.