Informe Final de Resultados

Presentado por: Luz Andrea Quiceno L. y Maribel Alvarez L.

Introducción

Este informe presenta los resultados finales del proyecto, cuyo objetivo es la construcción de una solución que clasifique artículos médicos en determinados grupos. Se detallará la metodología utilizada para implementar un sistema capaz de etiquetar los artículos, utilizando únicamente el título y el resumen, además de las evidencias recopiladas y las conclusiones obtenidas. El propósito es ofrecer una visión del trabajo realizado con las diferentes opciones exploradas y la justificación de la escogencia de la solución final.

Análisis y exploración del dataset

El dataset contiene títulos y resúmenes con muchas variantes morfológicas de palabras técnicas.

Se realizó la exploración de la frecuencia de cada grupo en el dataset:

- Neurological: 1785 artículos
- Cardiovascular: 1268 artículos
- Hepatorenal: 1091 artículos
- Oncological: 601 artículos

Con esta frecuencia, observamos que:

- El grupo neurológico es el más común, seguido por el cardiovascular y el hepatorenal.
- El grupo oncológico es el menos representado, lo que podría impactar la predicción y el balance del modelo.
- Existe un desequilibrio entre la frecuencia de los grupos. Esto es un indicador para monitorear las métricas por cada uno y considerar técnicas de balanceo.
- El modelo debe ser robusto para el etiquetado múltiple, ya que muchos artículos pertenecen a más de un grupo.

Justificación

Después de realizar el análisis exploratorio de los datos, nos dimos a la tarea de enlistar diferentes métodos que nos ayudaban a entrenar nuestro modelo, tales como:

- Modelo híbrido BERT + Logistic Regression (LR)
- Modelo clásico LinearSVC
- Modelo clásico Logistic Regression
- Modelo clásico Random Forest.
- Modelo clásico Gradient Boosting

Se exploró el modelo Experimental BERT + Logistic Regression (LR) utilizando embeddings de BERT para capturar de manera más profunda el contexto semántico de los textos. Este modelo ofreció un buen rendimiento, destacándose en la clasificación de las clases más frecuentes del dataset, sin embargo, su F1-Score ponderado global no superó al del pipeline TF-IDF + HistGradientBoosting.

Adicionalmente, el costo computacional fue notablemente mayor, tanto en tiempo de procesamiento para la generación de embeddings como en el consumo de memoria. Aunque es un enfoque potente, se descartó para la implementación final debido a lo anteriormente mencionado, ya que el beneficio marginal en rendimiento no justificó el incremento significativo en la complejidad y los recursos computacionales requeridos.

También se evaluaron varios algoritmos de clasificación tradicionales utilizando la misma representación de texto basada en TF-IDF. Los resultados de precisión (subset accuracy) fueron los siguientes:

LinearSVC: 0.71
Logistic Regression: 0.67
Random Forest: 0.54

Todos los anteriores modelos generaron una exactitud mucho más baja en comparación con HistGradientBoosting.

Por lo anterior, se optó por un método de **Machine Learning tradicional**, utilizando un modelo de **Gradient Boosting** en combinación con el esquema **OneVsRestClassifier** para abordar la clasificación **multietiqueta**.

La elección de este enfoque se justifica por varias razones:

Adaptación al problema multi-label:
El uso de MultiLabelBinarizer permite transformar las etiquetas múltiples de cada documento en un formato binario, facilitando el entrenamiento de modelos que pueden predecir varias clases para una misma instancia. El esquema

OneVsRestClassifier entrena un clasificador independiente para cada etiqueta, lo que es especialmente adecuado para problemas donde los documentos pueden pertenecer a más de una categoría.

Eficiencia y robustez:
El modelo HistGradientBoostingClassifier es eficiente y capaz de capturar relaciones complejas en los datos, lo que resulta útil en contextos donde las clases pueden estar correlacionadas o los textos presentan alta variabilidad.

Preprocesamiento especializado:
El uso de técnicas de procesamiento de lenguaje natural (NLP), como el stemming, asegura que el modelo trabaje con representaciones textuales limpias y relevantes, mejorando la calidad de las características extraídas mediante TF-IDF.

Escalabilidad y reproducibilidad:
El flujo implementado permite fácilmente probar otros modelos, ajustar hiperparámetros o incorporar nuevas técnicas de representación de texto, manteniendo la reproducibilidad y transparencia del proceso.

En resumen, el enfoque de **Machine Learning tradicional**, adaptado con técnicas específicas para clasificación **multietiqueta** y procesamiento de texto, es coherente y adecuado para resolver el problema planteado, permitiendo obtener resultados interpretables y comparables en el dominio biomédico.

Metodología

1. Carga y Preparación de Datos:

Se utilizó pandas para leer el archivo CSV que contiene los títulos, resúmenes y etiquetas de los artículos. Los campos de título y resumen se combinaron en una sola columna de texto para maximizar la información disponible para el modelo.

2. Preprocesamiento de Texto:

Aunque se definieron funciones para lematización y stemming, el preprocesamiento final aplicado antes de la vectorización utiliza únicamente stemming. El texto se normalizó convirtiéndolo a minúsculas, eliminando caracteres especiales y acentos, tokenizando con nltk, eliminando palabras vacías (stopwords) y aplicando stemming con SnowballStemmer para reducir las palabras a su raíz.

Esto ayuda a:
- Unificar variantes de palabras (mejorando la generalización del modelo).
- Reducir el número de características (menos palabras distintas).
- Hacer el modelo menos sensible a la forma gramatical de las palabras.

Se optó por el steeming debido a que el preprocesamiento con ésta técnica mostró ser el más eficaz, resultando en una mejor precisión (0.77) y puntuación F1 (0.90). Estos valores superaron ligeramente los obtenidos únicamente con lematización. Cabe destacar que la

combinación con steeming no presentó diferencias significativas en comparación con el steeming individual.

3. Vectorización:

Los textos procesados con stemming se transformaron en vectores numéricos utilizando TfidfVectorizer, limitando el número de características a las 5000 más relevantes para reducir la dimensionalidad y mejorar la eficiencia computacional.

4. Binarización de Etiquetas:

Las etiquetas múltiples de cada artículo se transformaron a formato binario mediante MultiLabelBinarizer, permitiendo abordar el problema como una clasificación multietiqueta.

5. División de Datos:

Los datos se dividieron en conjuntos de entrenamiento, validación y prueba (70%, 15%, 15%) usando train_test_split, asegurando una evaluación objetiva del modelo.

6. Entrenamiento del Modelo:

Se empleó un clasificador multietiqueta OneVsRestClassifier con un modelo base de HistGradientBoostingClassifier. El modelo fue ajustado con los datos de entrenamiento.

7. Evaluación y Métricas:

El desempeño del modelo se evaluó en los conjuntos de validación y prueba utilizando métricas de exactitud (accuracy) y F1-score ponderado. Además, se generó la matriz de confusión multietiqueta para cada clase, permitiendo analizar el comportamiento del modelo en detalle.

8. Exportación de Resultados:

Las métricas y la matriz de confusión se exportaron a archivos CSV para facilitar el análisis y la presentación de resultados.

Funciones utilizadas en cada paso:

1. Carga de datos:
- Lectura CSV:
  - pd.read_csv()
- Combinación de columnas:
  - df['title'].astype(str) + " " + df['abstract'].astype(str)

2. Preprocesamiento de texto
- Tokeniza el texto en palabras.
  - nltk.word_tokenize()
- Stemming:
  - SnowballStemmer().stem()

3. Vectorización
- TF-IDF:
  - TfidfVectorizer(max_features=5000)

- Transformación:
    - vectorizer.fit_transform()

4. Binarización de etiquetas
    - MultiLabelBinarizer:
        - MultiLabelBinarizer(), mlb.fit_transform()

5. División de datos
    - train_test_split:
        - train_test_split()

6. Entrenamiento del modelo
    - Definición del modelo:
        - HistGradientBoostingClassifier()
    - Clasificador multietiqueta:
        - OneVsRestClassifier()
    - Ajuste:
        - clasificador_multietiqueta.fit()

7. Evaluación
    - Predicción:
        - clasificador_multietiqueta.predict()
    - Accuracy:
        - accuracy_score()
    - F1-Score:
        - f1_score()

8. Análisis por clase y exportación
    - Matriz de confusión:
        - multilabel_confusion_matrix()
    - Exportar a CSV:
        - csv.writer(), writer.writerow()

Evidencias

Como resultado del desarrollo del proyecto, se generaron las siguientes evidencias que demuestran el correcto funcionamiento y la validez de la solución propuesta:

- Código fuente: El archivo train.py contiene todo el flujo de procesamiento, preprocesamiento, entrenamiento y evaluación del modelo de clasificación multietiqueta.
- Modelo entrenado y vectorizador: Los archivos classifier_gb_multilabel.pkl y vectorizer_gb_multilabel.pkl almacenan el clasificador y el vectorizador TF-IDF entrenados, permitiendo su reutilización sin necesidad de reentrenar.

- Resultados de métricas: Se presentan en consola las métricas de exactitud (accuracy), F1-score ponderado y los valores por clase, facilitando el análisis del desempeño del modelo.
- Matriz de confusión: El archivo confusion_matrix.csv contiene la matriz de confusión multietiqueta y las métricas de precisión, recall y F1-score para cada clase.
- Exportación de resultados: Todos los resultados relevantes se exportan a archivos CSV en la carpeta models/, permitiendo su consulta y análisis externo.
- Reporte metodológico: Se documenta el proceso seguido, las decisiones tomadas y las conclusiones obtenidas, asegurando la transparencia y reproducibilidad del proyecto.

Resultados

Los resultados obtenidos se pueden observar en la siguiente matriz de confusión, Esta matriz permite analizar el número de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) obtenidos por el modelo en cada categoría:

Clase	**TN**	**FP**	**FN**	**TP**	**Precisión**	**Recall**	**F1**	**Support**	**Accuracy**	**F1 Ponderado Global**
Cardiovascular	340	7	12	176	0.9617	0.9362	0.9488	188		
Hepatorenal	375	1	36	123	0.9919	0.7736	0.8693	159		
Neurological	250	24	29	232	0.9062	0.8889	0.8975	261		
Oncological	435	1	24	75	0.9868	0.7576	0.8571	99		
F1 ponderado					0.9516	0.8571	0.8991	707	0.7682	0.8991

A partir de la matriz de confusión, se calcularon las métricas de precisión, recall y F1-score para cada clase, lo que facilita identificar el comportamiento del modelo en cada etiqueta y detectar posibles áreas de mejora.

Conclusiones

En la realización de este proyecto, el cual es una significativa introducción al análisis de datos predictivo, fue importante comprender lo siguiente durante las diferentes etapas del proceso:

Evaluación General del Modelo:

- La división de datos en conjuntos de entrenamiento, validación y prueba es crucial para una evaluación sólida y para prevenir el sobreajuste del modelo.
- Se evaluaron tres estrategias de preprocesamiento (stem, lemma, combined). Stem fue elegido como estrategia final por ofrecer el mejor rendimiento en las métricas.
- Para el caso particular de etiquetado de artículos médicos, con el dataset proporcionado, El modelo de Gradient Boosting multietiqueta, combinado con TF-IDF y preprocesamiento de texto, logra una exactitud y F1-score ponderado que reflejan un buen desempeño en la clasificación de literatura médica y un buen rendimiento en el tiempo de entrenamiento. Es decir, ofreció un balance sólido entre rendimiento y eficiencia en el conjunto de datos multietiqueta.

Desempeño por Clase:

- Las métricas de precisión, recall y F1-score por clase ayudan a entender qué etiquetas son más difíciles de predecir y dónde el modelo podría mejorar.
- El F1 ponderado global con la técnica seleccionada fue de 0.8991. La matriz de confusión por clase muestra cómo el modelo distingue entre las diferentes categorías, permitiendo identificar clases con mayor cantidad de falsos positivos o falsos negativos.

Exportación y Reproducibilidad:

- Guardar el modelo y el vectorizador permite reutilizar la solución sin necesidad de reentrenar.
- Exportar la matriz de confusión y métricas a CSV facilita el análisis externo y la presentación de resultados.

Recomendaciones:

- Si algún grupo muestra baja precisión o recall, se recomienda revisar el balance de datos y considerar técnicas de remuestreo o ajuste de hiper parámetros.
- El flujo implementado es fácilmente escalable para probar otros modelos o técnicas de representación de texto.