Clasificación de textos médicos

Documentación

PRESENTADO POR:

Stellar Health Analytics

INTEGRANTES:

Eliana María Brand Agudelo

Valentina Hoyos Castrillón

Leidy Dahiana Hoyos Cardona

PRESENTADO A:

Techsphere Colombia

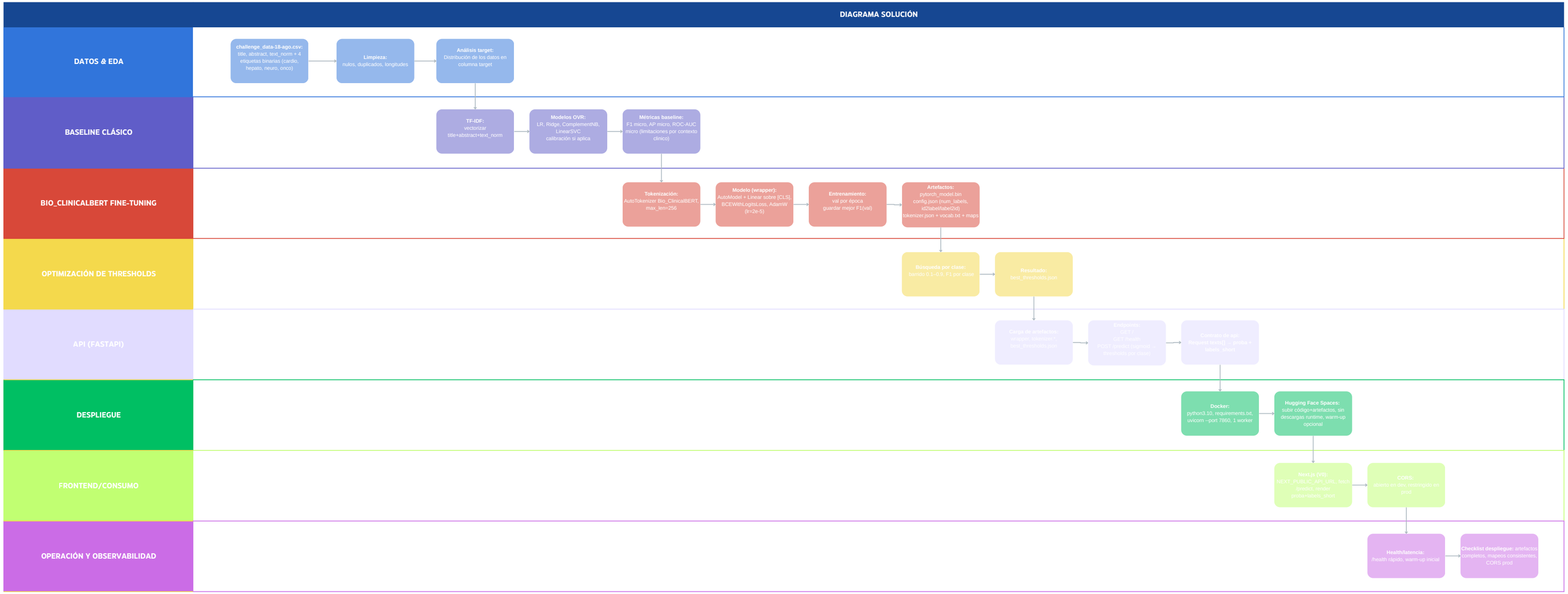


DIAGRAMA SOLUCIÓN
DATOS & EDA
challenge_data-18-ago.csv: title, abstract, text_norm + 4 etiquetas binarias (cardio, hepato, neuro, onco)
Limpieza: nulos, duplicados, longitudes
Análisis target: Distribución de los datos en columna target
BASELINE CLÁSICO
TF-IDF: vectorizar title+abstract+text_norm
Modelos OVR: LR, Ridge, ComplementNB, LinearSVC calibración si aplica
Métricas baseline: F1 micro, AP micro, ROC-AUC micro (limitaciones por contexto clínico)
BIO_CLINICALBERT FINE-TUNING
Tokenización: AutoTokenizer Bio_ClinicalBERT, max_len=256
Modelo (wrapper): AutoModel + Linear sobre [CLS], BCEWithLogitsLoss, AdamW (lr=2e-5)
Entrenamiento: val por época guardar mejor F1(val)
Artefactos: pytorch_model.bin config.json (num_labels, id2label/label2id) tokenizer.json + vocab.txt + maps
OPTIMIZACIÓN DE THRESHOLDS
Búsqueda por clase: barrido 0.1–0.9, F1 por clase
Resultado: best_thresholds.json
API (FASTAPI)
Carga de artefactos: wrapper, tokenizer, best_thresholds.json
Endpoints: GET / GET /health POST /predict {"texts":[...]} Parámetros por clase
Contrato de api: Request texts[] → probs + labels_short
DESPLIEGUE
Docker: python3.10, requirements.txt, uvicorn --port 7860, 1 worker
Hugging Face Spaces: subir código+artefactos, sin descargas runtime, warm-up opcional
FRONTEND/CONSUMO
Next.js (V0): NEXT_PUBLIC_API_URL, fetch /predict, render probs+labels_short
CORS: abierto en dev, restringido en prod
OPERACIÓN Y OBSERVABILIDAD
Health/latencia: /health rápido, warm-up inicial
Checklist despliegue: artefactos completos, mapeos consistentes, CORS prod

DESCRIPCIÓN DEL PROBLEMA

La clasificación automática de documentos científicos ha tenido un desarrollo importante en los últimos años, impulsado por el crecimiento acelerado de la literatura en el campo de la salud. Sin embargo, para lograr organizar y categorizar de manera precisa estos artículos, es necesario enfrentarse a la complejidad inherente del lenguaje biomédico. Este tipo de lenguaje combina tecnicismos, términos especializados y expresiones que pueden variar según el área de estudio, lo cual dificulta la identificación clara de la temática principal de cada publicación.

El presente proyecto tiene como propósito el desarrollo de un sistema de inteligencia artificial orientado a la clasificación automática de artículos de investigación médica a partir de la información contenida en su título y resumen. La funcionalidad central del sistema consiste en asignar cada artículo a una o varias categorías biomédicas predefinidas, específicamente: Cardiovascular, Neurológica, Hepatorrenal u Oncológica.

EDA

Se utiliza el método EDA para identificar patrones, valores atípicos y datos faltantes que afectan la calidad del dataset. Además, se asocia la visualización de la distribución y relaciones entre variables, lo cual orienta las decisiones de preprocesamiento.

1.1 Información general del dataset

- Visualización de los primeros 5 datos del dataset

	title	abstract	group
0	Adrenoleukodystrophy: survey of 303 cases: biochemistry, diagnosis, and therapy.	Adrenoleukodystrophy (ALD) is a genetically determined disorder associated with progressive central demyelination and adrenal cortical insufficiency . All...	neurological|hepatorenal
1	endoscopy reveals ventricular tachycardia secrets	Research question: How does metformin affect cancer through pituitary adenoma mechanisms? Methods: randomized controlled study with 53 elderly patients, ass...	neurological
2	dementia and cholecystitis: organ interplay	Purpose: This randomized controlled study examined statins for diabetes in adult population. The investigation included analysis of bun, cholelithiasis, and...	hepatorenal
3	The interpeduncular nucleus regulates nicotine's effects on free-field activity.	Partial lesions were made with kainic acid in the interpeduncular nucleus of the ventral midbrain of the rat. Compared with sham-operated controls, lesions ...	neurological
4	guillain-barre syndrome pathways in leukemia	Hypothesis: statins improves stroke outcomes via migraine pathways. Methods: cross-sectional trial with 285 adult population, measuring astrocytoma and pals...	neurological

- La base de datos cuenta con la siguiente información

Filas	3565
Número de Columnas	3
Nombre de columnas	"title", "abstract", "group"
Valores faltantes	"title": 0, "abstract": 0, "group": 0
Titulos duplicados	2
Abstract duplicados	0

El dataset cuenta con 3565 registros organizados en 3 columnas: *title*, *abstract* y *group*, donde se presenta el título del artículo, su resumen y la categoría biomédica correspondiente. No se

encontraron valores faltantes en ninguna de las variables. Sin embargo, se identificaron 2 títulos duplicados, aunque no existen duplicados en los *abstracts*, lo que indica que la base de datos se encuentra en general limpia, completa y adecuada para iniciar procesos de análisis y modelado.

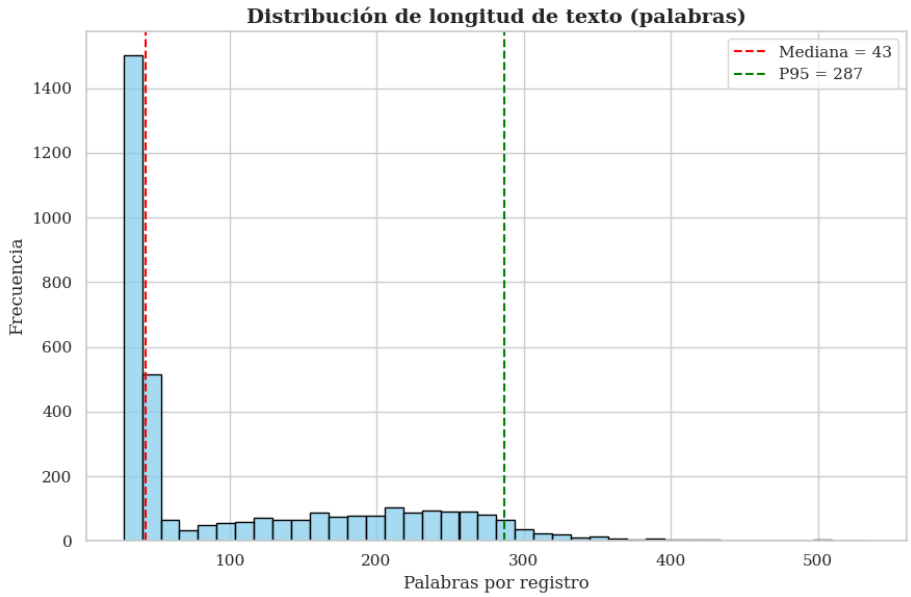
1.2 Limpieza del dataset

Se encuentra que los títulos *"Long term hormone therapy for perimenopausal and postmenopausal women"* y *"State-of-the-art thiazide diuretics for prostate cancer"* se encuentran duplicados.

	title	abstract	group
2246	Long term hormone therapy for perimenopausal and postmenopausal women.	BACKGROUND: Hormone therapy (HT) is widely used for controlling menopausal symptoms. It has also been used for the management and prevention of cardiovascul...	neurological|cardiovascular|oncological
2261	Long term hormone therapy for perimenopausal and postmenopausal women.	BACKGROUND: Hormone therapy (HT) is widely used for controlling menopausal symptoms and has also been used for the management and prevention of cardiovascul...	neurological|cardiovascular|oncological
1433	State-of-the-art thiazide diuretics for prostate cancer	Research question: How does beta-blockers affect cancer through cholestasis mechanisms? Methods: prospective study with 369 cardiac patients, assessing rena...	hepatorenal
3096	State-of-the-art thiazide diuretics for prostate cancer	Background: cancer affects diabetic patients worldwide, particularly involving arrhythmia and stenosis. Methods: We conducted a randomized controlled study ...	cardiovascular

Se optó por unificar los registros con títulos duplicados, conservando un único título y combinando los respectivos resúmenes. En este proceso, los artículos *"Long term hormone therapy for perimenopausal and postmenopausal women"* se mantuvo con su mismo título, abstarct y etiqueta, ya que este era el mismo, mientras que con *"State-of-the-art thiazide diuretics for prostate cancer"*, en el campo *abstract* se seleccionó la versión más extensa y se combinaron las etiquetas, garantizando así la preservación de la mayor cantidad de información posible. Al final de este proceso quedan 3563 datos

1.3 Estadísticas descriptivas de longitudes de los textos



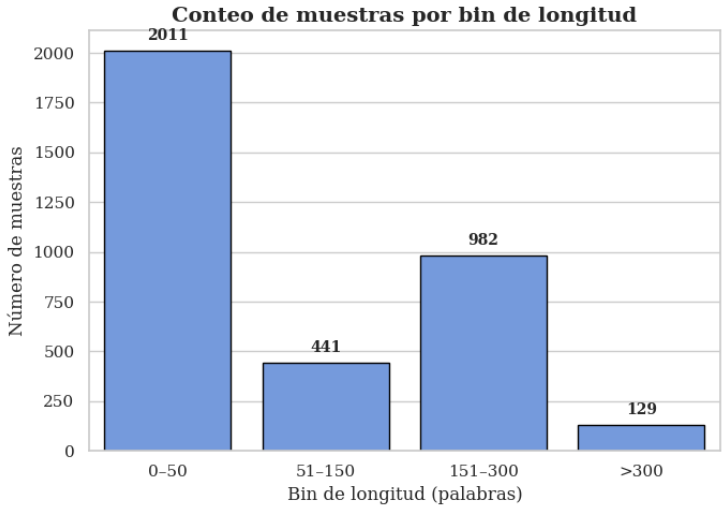
Promedio	108.72
Desviación estándar	96.51
Mediana	43
Percentil 90	261
Percentil 95	287
Máximo	535

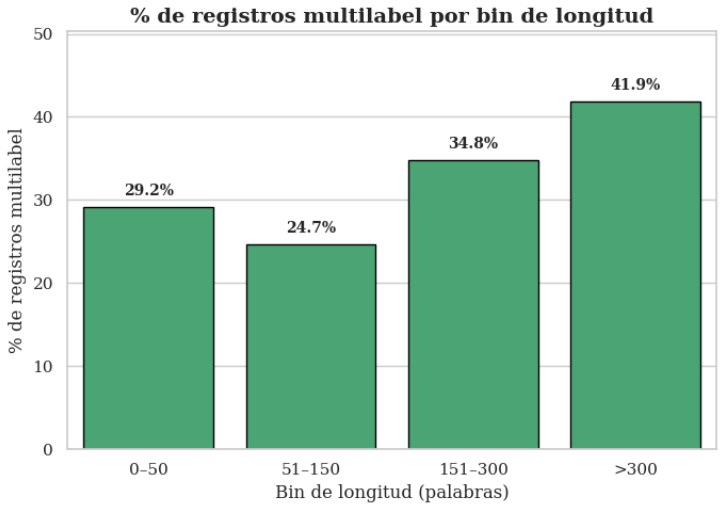
- Promedio: cada registro tiene en promedio unas 109 palabras.
- Desviación estándar: hay mucha variabilidad en la cantidad de palabras.

- Mediana: el 50% de los textos tiene 43 palabras o menos (indica que la mayoría de textos son cortos).
- Percentil 90: el 10% de los textos más largos tiene 261 palabras o más.
- Percentil 95: el 5% más largo tiene 287 palabras o más.
- Máximo: el texto más largo tiene 535 palabras.

1.4 Resumen estadístico de los textos del dataset agrupados por rangos de longitud

Intervalos de longitud de texto	Número de registros en cada rango	Longitud promedio de los textos	Percentil 95 de la longitud	Número promedio de categorías asignadas por registro	multilabel ratio %
0–50	2011	38.04	44	1.2919	29.2
51–150	441	104.01	146	1.2857	24.7
151–300	982	223.25	287	1.4084	34.8
300	129	354.76	474	1.5039	41.9



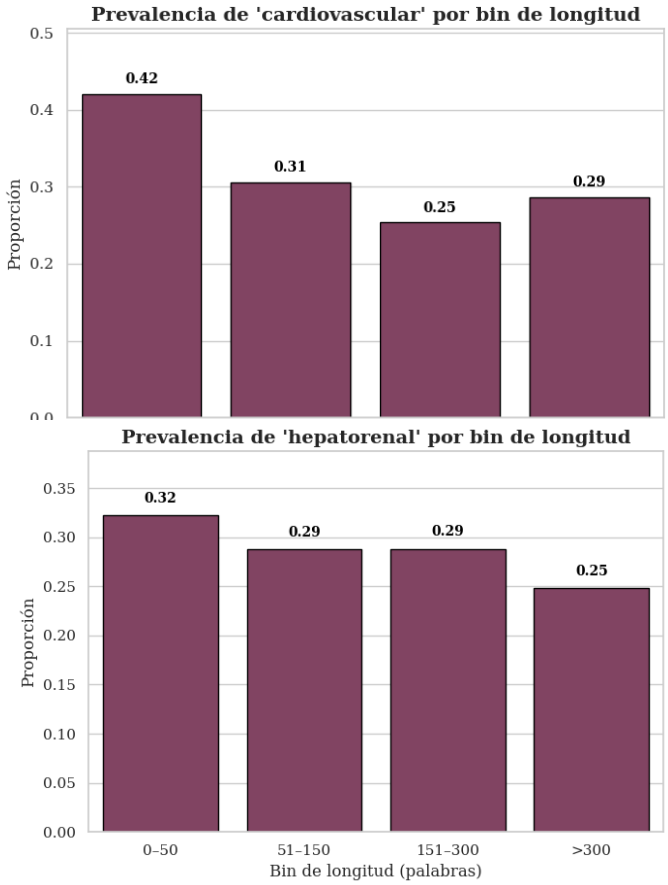


- Textos cortos (0–50 palabras): son la mayoría (2011 registros), con un promedio de 38 palabras y baja proporción de múltiples categorías (29.2%).

- Textos medianos (51–150 palabras): 441 registros, con longitud promedio de 104 palabras y baja proporción múltiples categorías (24.7%).
- Textos largos (151–300 palabras): 982 registros, con promedio de 223 palabras; aquí aumenta la tendencia a tener múltiples categorías (34.8%).
- Textos muy largos (>300 palabras): solo 129 registros, pero con promedio de 355 palabras y un alto porcentaje de múltiples categorías (41.9%). Además, tienden a tener más de una etiqueta; conviene usar umbrales por clase (no uno global).

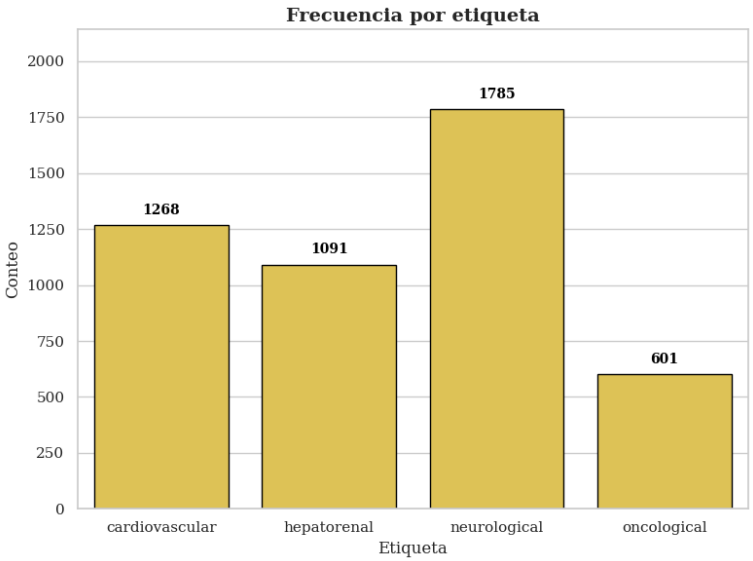
1.5 Cómo se distribuyen las categorías biomédicas según la longitud de los textos, expresado en proporciones

Intervalos de longitud de texto	cardiovascular	hepatorenal	neurological	oncological
0–50	0.421	0.323	0.402	0.146
51–150	0.306	0.288	0.546	0.145
151–300	0.254	0.288	0.646	0.221
300	0.287	0.248	0.775	0.194



- 0–50 palabras: predominan los artículos cardiovasculares (0.421) y neurológicos (0.402), mientras que los oncológicos son menos frecuentes (0.146).
- 51–150 palabras: la proporción más alta corresponde a neurológicos (0.546), mostrando que los resúmenes medianos suelen ser de esta categoría.
- 151–300 palabras: aquí la categoría neurológica (0.646) domina aún más, seguida por cardiovascular (0.254) y oncológica (0.221).
- >300 palabras: los textos largos tienen una fuerte concentración en la categoría neurológica (0.775), mientras que las demás disminuyen, especialmente hepatorenal (0.248).

1.6 Análisis de las categorías biomédicas según su frecuencia en la aparición de artículos



classes	label_counts
neurological	1785
hepatorenal	1091
cardiovascul	1268
oncological	601

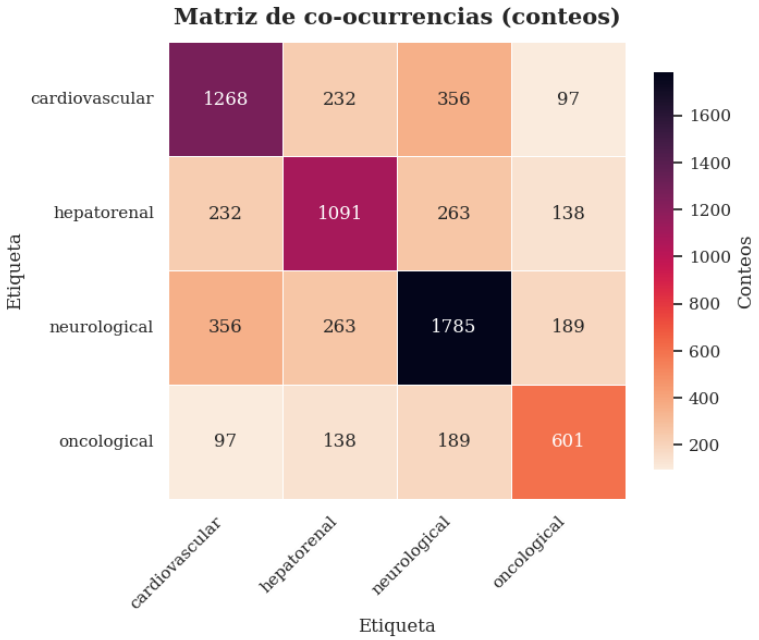
Número de etiquetas únicas: 4
Promedio de etiquetas por muestra: 1.331
Proporción de muestras con múltiples etiquetas: 0.306

El dataset está compuesto por 4 categorías, con una distribución desigual (predominan los artículos neurológicos) y alrededor de un tercio de los registros son multietiqueta, lo que lo convierte en un reto de clasificación multilabel.

1.7 Matriz de correlación

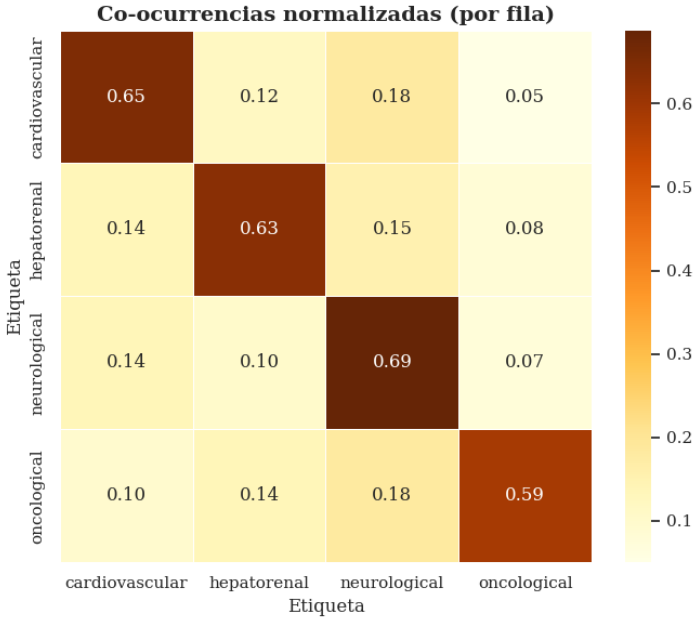
- Por conteo

	cardiovascular	hepatorenal	neurological	oncological
cardiovascular	1268	232	356	97
hepatorenal	232	1091	263	138
neurological	356	263	1785	189
oncological	97	138	189	601



- Por fila

	cardiovascular	hepatorenal	neurological	oncological
cardiovascular	**0.649**	**0.119**	**0.182**	**0.05**
hepatorenal	**0.135**	**0.633**	**0.153**	**0.08**
neurological	**0.137**	**0.101**	**0.688**	**0.073**
oncological	**0.095**	**0.135**	**0.184**	**0.586**



El dataset presenta un 30–40% de ejemplos con múltiples etiquetas, lo que confirma que no es multiclase puro sino multietiqueta real, por lo que resulta clave usar modelos que capturen correlaciones entre etiquetas para mejorar el rendimiento.

1.8 Top 10 términos repetidos por cada clase

- Cardiovascular

term	score
patients	0.042
cardiac	0.036
heart	0.036
results	0.033
methods	0.032
conclusion	0.031
disease	0.031
vascular insi	0.03
vascular	0.03
cancer	0.029

- Hepatorenal

term	score
patients	0.04
renal	0.035
methods	0.03
results	0.03
conclusion	0.028
cancer	0.027
disease	0.026
liver	0.026
organ	0.026
study	0.024

- Neurological

term	score
patients	0.037
results	0.024
disease	0.024
methods	0.023
cancer	0.023
study	0.022
conclusion	0.021
brain	0.02
pathways	0.018
induced	0.017

- Oncological

term	score
cancer	0.084
patients	0.04
brca1	0.033
mutations	0.029
breast	0.028
tumor	0.026
results	0.025
methods	0.025
conclusion	0.022
ovarian	0.022

PREPROCESAMIENTO

1. Unificamos 'title' + 'abstract' en 'text' (hecho en el EDA) para simplificar la entrada del modelo.
   Hay 4 columnas, incluida la columna Text que cuenta con el título del artículo más el abstract, cuenta con 3563 datos.

	title	abstract	group	text
0	"Real-world" data on the efficacy and safety o...	Lenalidomide and dexamethasone (RD) is a stand...	neurological	"Real-world" data on the efficacy and safety o...
1	22-oxacalcitriol suppresses secondary hyperpar...	BACKGROUND: Calcitriol therapy suppresses seru...	hepatorenal	22-oxacalcitriol suppresses secondary hyperpar...

2. Normalizamos texto de forma mínima y segura para datos biomédicos:

   - Minusculización y limpieza de espacios.
   - Conservamos dígitos, guiones y términos clínicos (no removemos agresivamente).
   - Evitamos stemming/lemmatization fuerte para no romper terminología médica.

```python
def normalize_text(s: str) -> str:
    if not isinstance(s, str):
        s = "" if pd.isna(s) else str(s)
    # Unicode NFKC: normaliza formas de caracteres
    s = unicodedata.normalize("NFKC", s)
    # Minúsculas
    s = s.lower()
    # Reemplazar múltiples espacios y saltos por un espacio
    s = re.sub(r"\s+", " ", s).strip()
    return s

df["text_norm"] = df["text"].apply(normalize_text)

# Comprobación rápida
print(df[["text","text_norm"]].head(1).to_string(index=False, max_colwidth=90))
```

   Normalización minimalista—evita destruir términos como "BRCA1", "TNF-α", etc. Convertimos a minúsculas, colapsamos espacios, removemos caracteres invisibles raros. Preservamos dígitos y signos relevantes (%, -, /) que aparecen en resúmenes.

3. Binaríamos etiquetas multilabel con un listado determinístico de clases.

```python
# Listas de etiquetas por fila
labels_series = df["group"].astype(str).apply(lambda s: [t.strip() for t in s.split("|") if t.strip()!=""])

# Conjunto de clases (orden determinista)
classes = sorted({lab for L in labels_series for lab in L})
print("Clases:", classes)

# Matriz binaria Y (n x C)
Y = pd.DataFrame([{c: int(c in L) for c in classes} for L in labels_series], dtype=int)
print(Y.sum().to_dict())  # positivos por clase

# Guardar lista de clases
with open(OUT_CLASSES, "w", encoding="utf-8") as f:
    f.write("\n".join(classes))
print("Guardado:", OUT_CLASSES)
```

	title	abstract	group	text	text_norm	cardiovascular	hepatorenal	neurological	oncological
0	"Real-world" data on the efficacy and safety o...	Lenalidomide and dexamethasone (RD) is a stand...	neurological	"Real-world" data on the efficacy and safety o...	"real-world" data on the efficacy and safety o...	0	0	1	0
1	22-oxacalcitriol suppresses secondary hyperpar...	BACKGROUND: Calcitriol therapy suppresses seru...	hepatorenal	22-oxacalcitriol suppresses secondary hyperpar...	22-oxacalcitriol suppresses secondary hyperpar...	0	1	0	0

4. Dividimos train/val/test con estratificación multilabel (iterative-stratification) y semilla fija.

```
Train+Val: 2963  | Test: 600
Guardado: ..\data\splits.json
(600, [2368, 2373, 2368, 2369, 2374], [595, 590, 595, 594, 589])
```

Usamos MultilabelStratifiedKFold para mantener las proporciones por clase en cada split. Primero separamos una prueba estable (evaluación final), y luego hacemos 5 folds para ajustar el modelo y umbrales.

5. Generamos TF-IDF aquí si el baseline clásico lo requiere, para ahorrar tiempo en el entrenamiento.

```
TF-IDF shape: (3563, 23621)
Guardado vectorizador: ..\models\baseline\tfidf.joblib
```

TF-IDF (1–2-gramas, min_df=3, sublinear_tf=True) rinde fuerte como baseline en textos clínicos. Guardamos el vectorizador para reutilizar.

6. Resumen de la información del dataset después de su procesamiento.

	rows'	3563
	'cols'	9
	'classes'	'cardiovascular', 'hepatorenal', 'neurological', 'oncological'
'positives per class'	'cardiovascular'	1267
	'hepatorenal'	1091
	'neurological'	1784
	'oncological'	600
	'has_text_norm'	True
'files'	'processed_csv'	'True'
	'classes_txt'	'True'
	'splits_json'	'True'
	'tfidf_vectorizer'	'True'

El conjunto de datos procesado consta de 3.563 registros y 9 columnas, con cuatro clases de salida: cardiovascular, hepatorenal, neurological y oncological. La distribución de ejemplos positivos por clase evidencia un desbalance, con 1.267 casos cardiovasculares, 1.091 hepatorenales, 1.784 neurológicos y solo 600 oncológicos. Además, el texto fue sometido a un proceso de normalización, y se generaron archivos auxiliares como el dataset procesado en formato CSV, la lista de clases, la división en subconjuntos (train, test y validación) y el vectorizador TF-IDF. Esta información resume tanto la estructura como el preprocesamiento aplicado, lo cual garantiza que los datos estén listos para su uso en modelos de aprendizaje automático.

1. Ensayo inicial con métodos tradicionales

- **Notebook:** aquí

1.1 Configuración Experimental

División de Datos

- **Random seed**: 42
- **Splits**: Los datos se dividen en train/validation/test usando `splits.json`
- **Estrategia**: Estratificación por combinación de etiquetas para mantener distribución multilabel
- **Dimensiones finales**:
  - Train: 2,368 muestras
  - Validation: 595 muestras
  - Test: 600 muestras

1.2 Preprocesamiento de Texto

TF-IDF Vectorizer:

- **N-gramas**: (1,2) - unigrams y bigrams
- **Frecuencia mínima**: 2 documentos
- **Frecuencia máxima**: 90% del corpus
- **Máximo features**: 120,000
- **Transformaciones**: lowercase, strip_accents="unicode", sublinear_tf=True
- **Dimensión final**: 28,782 características

1.3 Modelos Evaluados

1. Logistic Regression (`logreg_C2`)

```
"logreg_C2": OneVsRestClassifier(
    LogisticRegression(solver="saga", penalty="l2", C=2.0,
                       max_iter=MAX_ITER_LR, n_jobs=N_JOBS,
                       class_weight="balanced", random_state=SEED),
    n_jobs=N_JOBS),
```

2. Ridge Classifier (`ridge_a1`)

```
"ridge_a1": OneVsRestClassifier(
    RidgeClassifier(alpha=1.0, class_weight="balanced", random_state=SEED),
    n_jobs=N_JOBS),
```

3. Linear SVM Calibrado (`linsvm_C1_cal`)

```
"linsvm_C1_cal": OneVsRestClassifier(
    CalibratedClassifierCV(LinearSVC(C=1.0, random_state=SEED),
                           method="sigmoid", cv=5),
    n_jobs=N_JOBS),
```

4. Complement Naive Bayes (`cnb_a05`)

```
"cnb_a05": OneVsRestClassifier(ComplementNB(alpha=0.5), n_jobs=N_JOBS),
```

1.4 Métricas de Evaluación

Para cada modelo se calculan:

- **F1 Micro**: F1-score promediado por instancias
- **F1 Macro**: F1-score promediado por clases
- **PR Macro**: Average Precision Score macro-promediado
- **ROC Macro**: AUC-ROC macro-promediado

1.5 Optimización de Thresholds

- Se optimizan umbrales por clase para maximizar F1-score individual
- Búsqueda en grid: `np.linspace(0.2, 0.8, 31)`
- Fallback a 0.5 para clases sin ejemplos positivos

```
Entrenando logreg_C2 ...
Entrenando ridge_a1 ...
Entrenando linsvm_C1_cal ...
Entrenando cnb_a05 ...
```

	f1_micro	f1_macro	pr_macro	roc_macro	name
0	0.892269	0.884619	0.950481	0.969225	linsvm_C1_cal
1	0.888318	0.878914	0.947031	0.967621	ridge_a1
2	0.878353	0.870171	0.941549	0.963196	logreg_C2
3	0.810845	0.773244	0.842227	0.906894	cnb_a05

1.6 Rendimiento en Test (mejor modelo)

Linear SVM Calibrado (`linsvm_C1_cal`):

- F1 Micro: 0.8710
- **F1 Macro: 0.8628**
- PR Macro: 0.9377
- ROC Macro: 0.9582

Ensemble

Se evaluó un ensemble promediando las probabilidades de todos los modelos:

- **Validación**: F1 Macro = 0.8708
- **Test**: F1 Macro = 0.8459

Resultado: El modelo individual (Linear SVM) superó al ensemble.

Conclusiones

Modelo Ganador

Linear SVM Calibrado (`linsvm_C1_cal`)

- Mejor F1 macro en validación (0.8846) y test (0.8628)
- Excelente balance entre precisión y recall
- La calibración mejora significativamente las probabilidades

Insights Clave

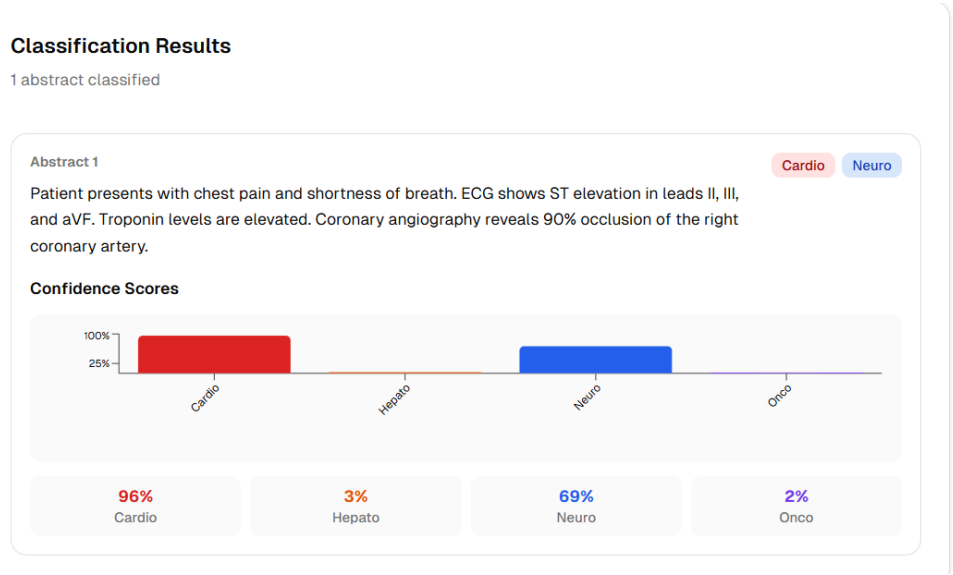
1. **Linear SVM** mostró la mejor capacidad de generalización
2. **Ridge Classifier** fue segundo lugar con rendimiento muy similar
3. **Complement Naive Bayes** tuvo el peor rendimiento, posiblemente por asumir independencia entre features
4. **Calibración** es crucial para SVM en tareas multilabel
5. **Ensemble simple** no mejoró sobre el mejor modelo individual

Limitaciones Identificadas con TF-IDF

Problema Principal: TF-IDF mostró limitaciones significativas para entender **semántica médica**:

- No captura relaciones semánticas entre términos médicos
- Sinónimos médicos son tratados como tokens completamente diferentes
- Conceptos médicos relacionados no se vinculan automáticamente
- Dependencia excesiva de co-ocurrencias exactas de palabras

Lo descubrimos, *un poco tarde*, a la hora de probar en V0:



El modelo confundía las etiquetas cardio + neuro, posiblemente por la incidencia en el dataset (aproximadamente el 10% de los casos eran abstracts combinados de cardiología y neurología). En el ejemplo se puede ver como un abstract evidentemente cardio, tenía un porcentaje significativo en neuro.

Características Técnicas

- **Dimensionalidad alta** (28K features) manejada eficientemente por modelos lineales
- **Class weighting** benefició a todos los modelos
- **TF-IDF sublinear** ayudó con la escala de características
- **OneVsRest** permitió manejar la naturaleza multilabel del problema

Artefactos Generados

El modelo ganador se guardó con los siguientes componentes:

- `model.joblib`: Modelo Linear SVM calibrado entrenado
- `tfidf.joblib`: Vectorizador TF-IDF ajustado
- `thresholds.json`: Umbrales optimizados por clase

1.7 Próximos Pasos y evolución del proyecto

Migración a BERT

Debido a las limitaciones semánticas identificadas con TF-IDF en el dominio médico (ver ejemplo anterior en v0), se decidió evolucionar hacia **BERT (Bidirectional Encoder Representations from Transformers)**:

Motivación para el cambio:

- BERT puede capturar relaciones semánticas complejas en texto médico
- Embeddings contextuales que entienden sinónimos médicos

- Mejor comprensión de terminología especializada
- Capacidad de transferencia desde modelos pre-entrenados en textos médicos

Migración de Entorno: VSCode a Google Colab

Problema técnico identificado:

- El kernel de VSCode se **colgaba consistentemente** al trabajar con BERT
- Modelos transformer son **computacionalmente intensivos**
- Limitaciones de memoria RAM local
- Tiempo de procesamiento prohibitivo en hardware local

Solución implementada:

- Migración completa del proyecto a **Google Colab**
- Acceso a GPUs gratuitas (Tesla T4/P100)
- Mejor gestión de memoria para modelos grandes
- Entorno más estable para experimentación con transformers

2. Migración a Bio_ClinicalBERT (Fine-Tuning)

Notebook: aqui

2.1. Preparación del Dataset

- **Columnas de texto usadas:** `title`, `abstract`, `text_norm` (concatenadas en una sola columna `all_text`).
- **Etiquetas multilabel:** `cardiovascular`, `hepatorenal`, `neurological`, `oncological`.
- **Representación de etiquetas:** tensores binarios 0/1 en PyTorch.
- **División de datos:** se mantuvo el mismo split estratificado de train/valid/test usado en los modelos clásicos para comparabilidad.

2.2. Tokenización

- **Modelo base:** `emilyalsentzer/Bio_ClinicalBERT`.
- **Tokenizer:** `AutoTokenizer` con truncamiento a `max_length=256`.
- **Encodings generados:** tensores de `input_ids` y `attention_mask` para train/valid/test.

- **Motivación del cambio:** TF-IDF no captura semántica médica; BERT genera embeddings contextuales pre-entrenados en notas clínicas.

2.3. Arquitectura del Modelo

- **Wrapper personalizado (`BertForMultilabel`)**:
    - Entrada: embeddings de Bio_ClinicalBERT.
    - Se toma el vector `[CLS]` como representación global del abstract.
    - Capa final: `Linear(hidden_size → 4)` para predecir las cuatro etiquetas.
- **Loss:** `BCEWithLogitsLoss` para multilabel.
- **Optimizer:** `AdamW(lr=2e-5)`.
- **Scheduler:** decay con `StepLR`.

2.4. Entrenamiento

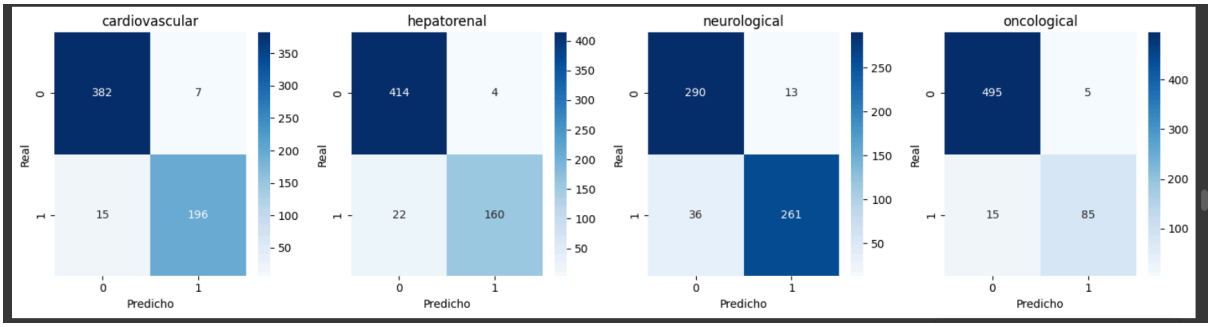
- **Dataset y Dataloaders:** batch size = 8.
- **Épocas:** 3.
- **Validación en cada época:** métricas F1/AP/ROC micro.
- **Selección del mejor modelo:** se guarda `bioclinicalbert_finetuned.pt` cuando mejora F1.
- **Hardware usado:** GPU T4 en Google Colab.

2.5. Evauación Final

- **Resultados en test set:**

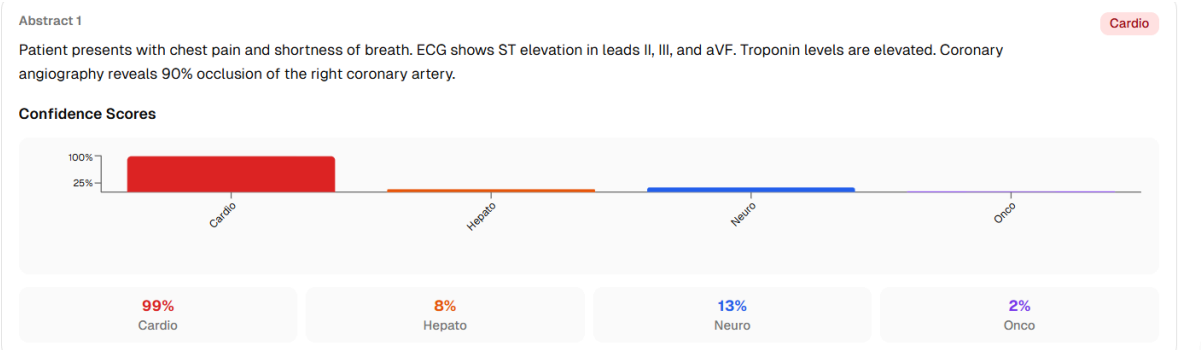
```
Classification Report
                precision    recall  f1-score   support

cardiovascular      0.966     0.929     0.947       211
   hepatorenal      0.976     0.879     0.925       182
  neurological      0.953     0.879     0.914       297
   oncological      0.944     0.850     0.895       100

     micro avg      0.960     0.889     0.923       790
     macro avg      0.960     0.884     0.920       790
  weighted avg      0.960     0.889     0.923       790
   samples avg      0.964     0.920     0.931       790
```

- **Reporte detallado por clase:**



- **Hallazgos clave:**
    - Bio_ClinicalBERT entiende mejor los sinónimos y relaciones clínicas.
    - Mejora sustancial en abstracts ambiguos cardio/neuro.
    - Reducida confusión respecto a TF-IDF.

Lo comprobamos con la mejora del abstract de ejemplo anterior:



2.6. Optimización de Thresholds

- **Procedimiento:** barrido 0.1 → 0.9 en validación para cada etiqueta.
- **Archivo generado:** `best_thresholds.json`.
- **Ejemplo:** cardio 0.55, hepato 0.45, etc.
- **Impacto:** mejora recall en clases minoritarias y F1 macro global.

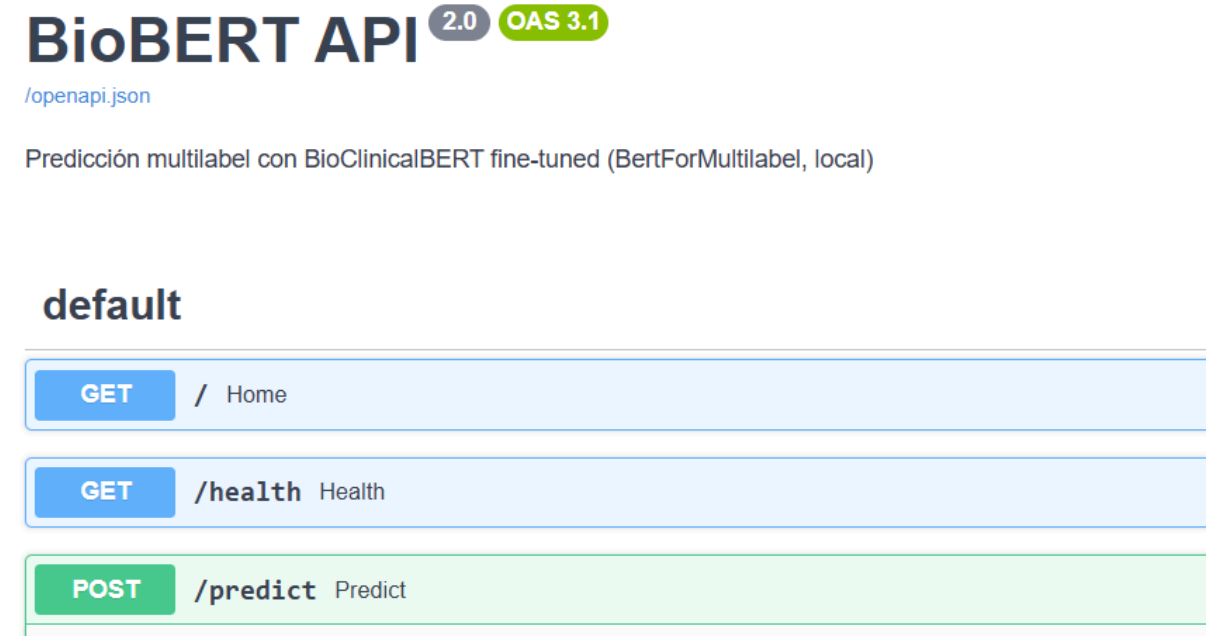
2.7. Artefactos Guardados

- `pytorch_model.bin` → pesos del modelo fine-tuned.
- `tokenizer/` → vocab, special_tokens_map, tokenizer_config.
- `config.json` → estructura del modelo (labels, hidden_size).
- `best_thresholds.json` → thresholds optimizados.

3. API y Despliegue

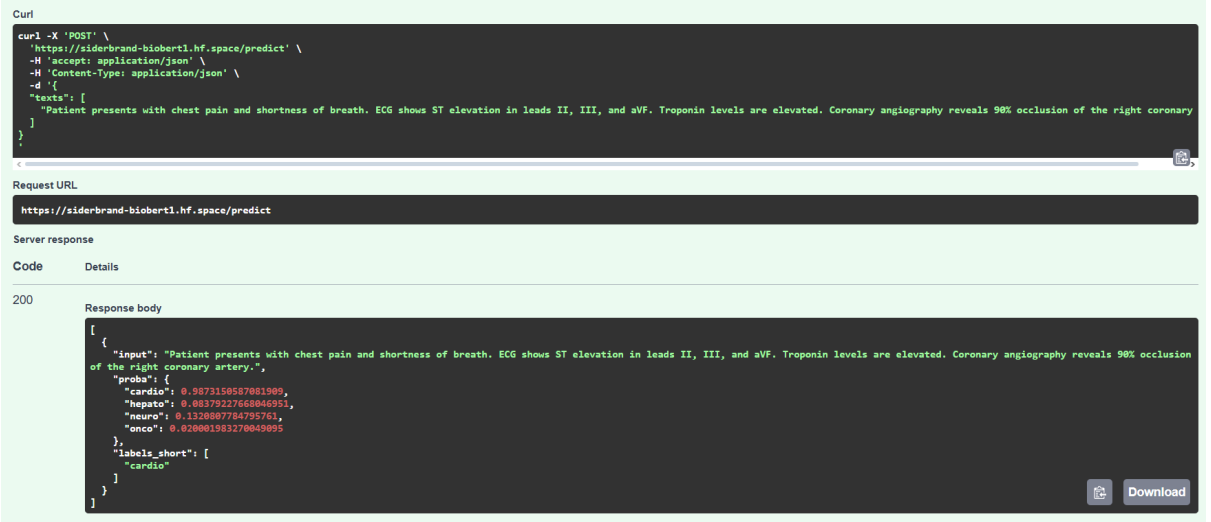
3.1. Diseño de la API

- **Framework elegido:** `FastAPI` (ligero, rápido y con soporte nativo para documentación Swagger).
- **Endpoints disponibles:**
  - `GET /` → mensaje de bienvenida.
  - `GET /health` → chequeo de estado ("ok" si el modelo está cargado).
  - `POST /predict` → recibe uno o varios abstracts clínicos en JSON y devuelve:
    - `proba`: probabilidades por clase (con nombres cortos: cardio, hepato, neuro, onco).
    - `labels_short`: etiquetas activadas según thresholds optimizados.



- **CORS:** habilitado globalmente para permitir consumo desde frontend en V0/Next.js.

Ejemplo en **/predict:**



3.2. Carga del Modelo

- **Clase usada:** `BertForMultilabel` (misma arquitectura que en el entrenamiento).
- **Artefactos cargados:**
    - `pytorch_model.bin` (pesos fine-tuned).

    - `tokenizer/` (config + vocab).
    - `best_thresholds.json` (umbrales optimizados).
- **Problema inicial:**
    - El `.bin` pesaba ~414 MB, lo cual hacía inviable subirlo directamente a GitHub/Render (limitaciones de almacenamiento y tiempo de build).

3.3. Problemas con Render + Git LFS

- Render no manejó bien la descarga de modelos grandes con Git LFS.
- Los tiempos de despliegue eran muy largos y la API no se levantaba.
- El error más frecuente: `OutOfMemory` y `PermissionError` al intentar descargar el `.bin`.

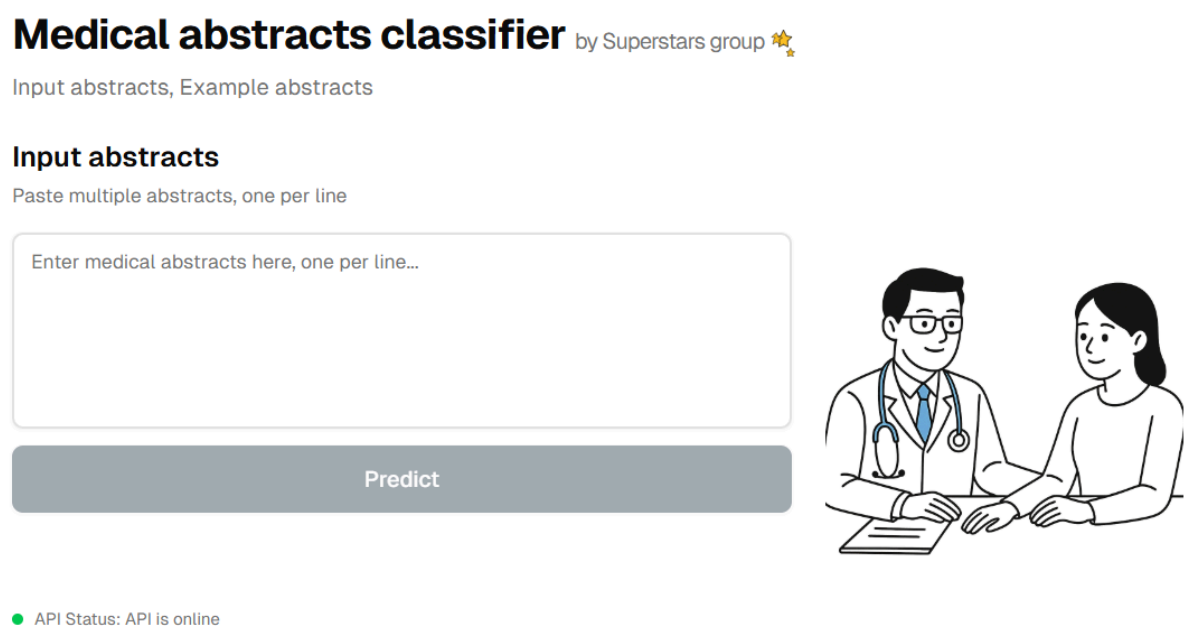
3.4. Migración a Hugging Face Spaces

- **Solución adoptada:**
    - Subir el modelo completo (bin + tokenizer + thresholds) a un repositorio en **Hugging Face Hub**.
    - Modificar `app.py` para que `AutoTokenizer` y `AutoModel` pudieran cargar desde `siderbrand/bioclinicalbert-finetuned` en lugar de solo archivos locales.
    - Configurar caché en `/code/.cache` para evitar errores de permisos en contenedores HF.
- **Ventajas de Spaces sobre Render:**
    - Manejo nativo de modelos grandes.
    - Infraestructura optimizada para Transformers.
    - Integración con Docker y con la librería `transformers`.
    - URL pública y persistente: `https://siderbrand-biobert1.hf.space`.

3.5. Dockerización

- Se creó un `Dockerfile` ligero basado en `tiangolo/uvicorn-gunicorn-fastapi:python3.10`.
- Paquetes mínimos instalados: `transformers`, `torch`, `uvicorn`, `fastapi`.
- El modelo se descarga automáticamente desde HF al primer uso.

3.6. Integración con Frontend (V0)



Arquitectura del Frontend:

Framework: Next.js 14 con App Router desarrollado en V0

UI Components: shadcn/ui + Tailwind CSS para diseño responsive

Visualización: Recharts para gráficos de confianza en tiempo real

Estado: React hooks (useState, useEffect) para manejo de estados local

Implementación de la API:

Endpoint: POST → `/predict` **con payload** `{ "texts": ["..."] }`

URL Base: https://siderbrand-biobert1.hf.space

Configuración: Variable de entorno `NEXT_PUBLIC_API_URL` para flexibilidad

Health Check: Monitoreo automático cada 5 minutos con indicador visual

Características Técnicas:

Procesamiento por lotes: Soporte para múltiples abstracts simultáneos

Visualización de resultados: Cards individuales con mini-gráficos de barras para confianzas

Estados de carga: Indicadores visuales durante predicciones

Manejo de errores: Captura y display de errores de red/API

Ejemplos integrados: Botones pre-cargados para cada categoría médica

Problemas Resueltos:

Error 405 inicial: Configuración incorrecta de endpoints → solucionado separando URL base de /predict

Llamadas "pegadas" en móviles: Optimización de health checks y configuración CORS

Parsing de respuesta: Mapeo correcto de proba object y labels_short array