Universidad
Rey Juan Carlos

NLP II - Práctica 2

Clasificación Biclase de Reviews

Retrieval, Clasificadores Híbridos, Explicabilidad, Destilación y Generador de Resúmenes

Jorge Camacho y Alberto Pérez

Procesamiento del Lenguaje Natural

Índice

1 Extracción, Armonización e Importación de los Datos

1.1 Carga del Dataset

La clase `DataLoader` es la que se encarga de la descarga del dataset, de procesarlo y de dividirlo en conjuntos de entrenamiento, validación y test. Se han implementado las siguientes funcionalidades:

- Descarga automática mediante la librería de `kagglehub`.
- Limpieza y armonización de los datos.
- División estratificada de los datos.
- Tratado de las etiquetas usando `LabelEncoder`.

1.2 Características del Dataset

El dataset final está formado por varios. Este contiene reviews de Amazon, reviews de películas y reviews de restaurantes.
Las siguientes tablas recogen características del dataset ya armonizado. La primera muestra el número de ejemplos de cada conjunto de datos y la segunda muestra características de nuestro dataset.

Conjunto	**Número de Muestras**
Entrenamiento	200,800
Validación	25,100
Test	25,100
Total	**251,000**

Métrica	**Valor**
Total de muestras	251,000
Número de clases	2
Distribución Clase 0 (Negativo)	125,677 (50.1 %)
Distribución Clase 1 (Positivo)	125,323 (49.9 %)
Longitud media del texto	604.18 caracteres
Longitud mínima	11 caracteres
Longitud máxima	13,704 caracteres

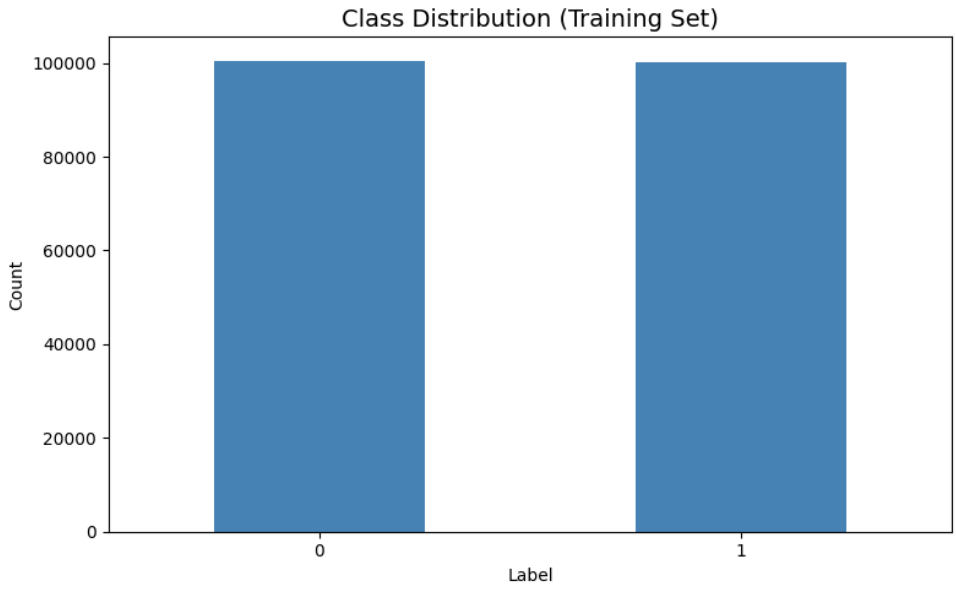

Figura 1: Distribución de clases en el conjunto de entrenamiento

Este gráfico muestra la distribución de las clases. Como se puede observar, estas ya se encuentran balanceadas por lo que no hemos tenido que usar ninguna técnica adicional para conseguirlo.

2 Transformer Práctica 1

2.1 Configuración del modelo

Utilizamos el pipeline de la práctica 1, utilizando un `RoBERTa` en vez de un `DeBERTa` al ser más ligero. El modelo seleccionado es `RoBERTa-base`. Este modelo lo instanciamos utilizando la clase `ModelTrainer` para facilitar el proceso de entrenamiento.

Características principales:

- Arquitectura: `RobertaForSequenceClassification`.
- Número de parámetros: 124,647,170 ($\approx$ 125M).
- Longitud máxima de secuencia: 256 tokens.
- Estrategia de pooling: CLS token.

2.2 Resultados de Evaluación

Durante el entrenamiento, realizamos distintos experimentos que fueron evaluados sobre el conjunto de validación. Estos experimentos nos ayudaron a decidir la mejor configuración de parámetros para nuestro modelo. Al final, tras decidir qué configuración era la mejor, evaluamos el modelo sobre el conjunto de test, del que obtuvimos los siguientes resultados:

Métrica	**Valor**
Loss (Test)	0.1114
Accuracy (Test)	0.9634
Macro F1 (Test)	0.9634

3 Módulo de Retrieval Denso

3.1 Configuración del modelo

El módulo de Retrieval Denso lo hemos implementado a través de las clases `EmbeddingExtractor` y `DenseRetriever`. Este sistema nos permite, dado un documento, encontrar documentos similares a él en el espacio de embeddings utilizando la distancia coseno.

Clases utilizadas:

- `EmbeddingExtractor`: Esta clase es la encargada de, dado un documento, extraer su embedding a través del modelo `RoBERTa` utilizando el token CLS.
- `DenseRetriever`: Esta clase es la que a través de los embeddings construidos por la clase anterior, crea un índice de búsqueda y encuentra sus k vecinos más cercanos.

3.2 Construcción del Índice

El índice lo hemos construido extrayendo los embeddings de los datos de entrenamiento. Estas son las características de nuestro índice:

Métrica	**Valor**
Número de documentos	200,800
Dimensión del embedding	768
Número de etiquetas	2
Nº ejemplos Clase 0	100,541
Nº ejemplos Clase 1	100,259
Métrica de distancia utilizada	Coseno
Tiempo de construcción	$\approx$ 38 minutos

3.3 Resultados obtenidos

La evaluación del retrieval la hicimos utilizando las métricas estándar de recuperación de información. A continuación, se muestran los resultados obtenidos probando diferentes k (número de vecinos cercanos):

k	**Precision**	**Recall**
1	0.9600	0.9600
3	0.9700	0.9950
5	0.9720	1.0000
10	0.9665	1.0000

Cuadro 1: Precision (k = 5) por Clase

Clase 0 (Negativo)	0.9735
Clase 1 (Positivo)	0.9701

Tras analizar los resultados, podemos ver que el sistema de retrieval alcanza un recall perfecto (1.0) con k=5. Gracias a estas métricas, podemos afirmar que los ejemplos más relevantes siempre están entre los 5 vecinos más cercanos.

4 Clasificador k-NN

4.1 Configuración del modelo

La clase `KNNClassifier` es la que implementa un clasificador basado en los k vecinos más cercanos. Esta utiliza el módulo de retrieval denso para encontrar los ejemplos más parecidos y después, realiza una clasificación por votación.

Parámetros de la clase:

- k = 5 (número de vecinos).
- Número de clases: 2.
- Votación: No ponderada por cercanía (simplemente mayoría).
- Métrica para elegir los k más cercanos: Distancia coseno.

4.2 Resultados obtenidos

Como con el modelo transformer, evaluamos el clasificador con los datos de test. El conjunto de validación lo utilizamos para realizar experimentos y ajustar los parámetros. A continuación, se muestra una tabla que recoge varias métricas y una matriz de confusión:

Cuadro 2: Métricas del Clasificador k-NN

Métrica	**Valor**
Accuracy	0.9651
Macro F1	0.9651
Weighted F1	0.9651
Macro Precision	0.9651
Macro Recall	0.9651
F1 Clase 0	0.9653
F1 Clase 1	0.9650

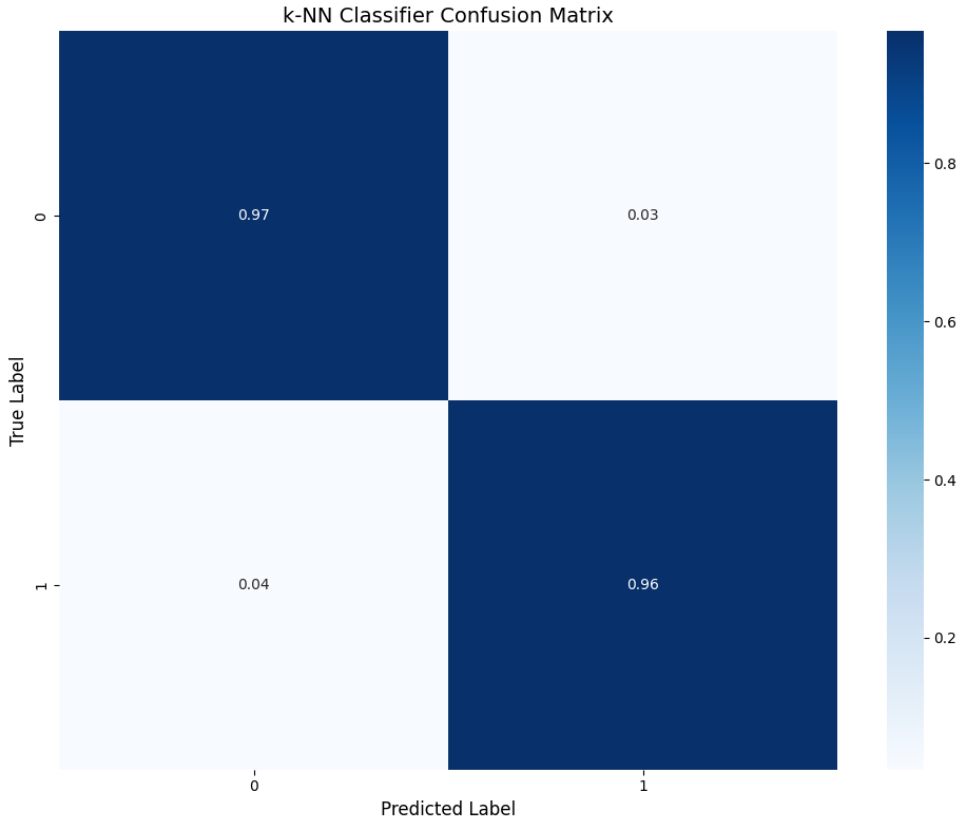

Figura 2: Matriz de confusión del clasificador k-NN

4.3 Comparación entre el modelo transformer y el clasificador KNN

Cuadro 3: Comparación k-NN vs Teacher

Métrica	**k-NN**	**Teacher (RoBERTa)**
Accuracy	0.9651	0.9634
Macro F1	0.9651	0.9634

El clasificador k-NN supera ligeramente al modelo Transformer (+0.17 % en accuracy), demostrando la efectividad del enfoque basado en retrieval para esta tarea.

5 Clasificador Híbrido - Transformer + k-NN (RAG)

5.1 Configuración del modelo

La clase `HybridClassifier` implementa un sistema de clasificación híbrido cuya tarea es combinar las predicciones del modelo Transformer con las del clasificador k-NN mediante un parámetro α. Este parámetro es el que pondera la importancia que se le da a cada una de las predicciones:

$$P_{hybrid} = \alpha \cdot P_{transformer} + (1 - \alpha) \cdot P_{kNN} \tag{1}$$

donde:

- $\alpha = 0$: Solo k-NN.
- $\alpha = 1$: Solo Transformer.
- $0 < \alpha < 1$: Combinación de ambos ponderada.

5.2 Experimento de Valores de Alpha

Como en todas las implementaciones anteriores, realizamos varios experimentos para encontrar el valor óptimo de α. A continuación se muestra una tabla donde se muestran los resultados obtenidos para los distintos valores de α:

Alpha (α)	Accuracy	Macro F1
0.00	0.9651	0.9651
0.25	0.9657	0.9657
0.50	**0.9658**	**0.9658**
0.75	0.9655	0.9655
1.00	0.9656	0.9656

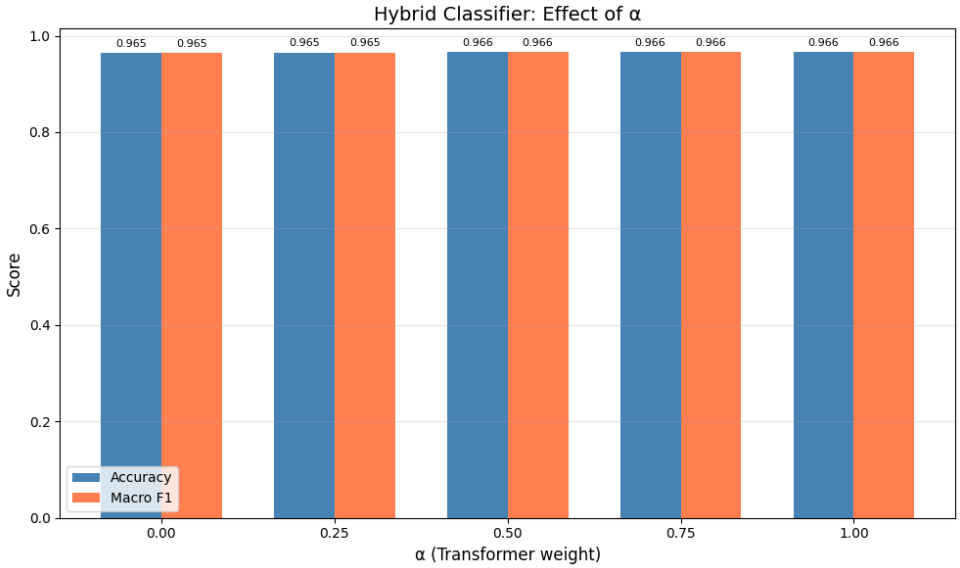

Figura 3: Efecto de alpha al entrenar el modelo

5.3 Resultados obtenidos por el Clasificador Híbrido

Después de realizar los experimentos, decidimos que el mejor valor era $\alpha = 0.5$. A continuación, se muestra una tabla que recoge los resultados obtenidos:

Cuadro 4: Métricas del Clasificador Híbrido ($\alpha = 0.5$)

Métrica	**Valor**
Accuracy	0.9658
Macro F1	0.9658
Weighted F1	0.9658
Macro Precision	0.9658
Macro Recall	0.9658
F1 Clase 0	0.9659
F1 Clase 1	0.9657

Como se puede observar, el clasificador híbrido con $\alpha = 0.5$ ha conseguido el mejor rendimiento global con 96.58 % de accuracy, superando tanto al modelo Transformer como al k-NN en solitario.

6 Explicabilidad Basada en el Índice Retrieval

6.1 Configuración del modelo

La clase `ExplainabilityModule` es la encargada de crear explicaciones para las predicciones, en otras palabras, rompe nuestra caja negra. La clase sigue los siguientes pasos:

1. Recupera los k ejemplos más parecidos del conjunto de entrenamiento.
2. Analiza la distribución de etiquetas entre los vecinos.
3. Identifica palabras o términos parecidos entre los vecinos.
4. Genera una explicación textual basada en lo anterior.

6.2 Análisis de Predicciones Correctas

En la práctica, analizamos 10 ejemplos que han sido correctamente clasificados. A continuación se muestran casos representativos:

Cuadro 5: Ejemplo de Predicción Correcta - Clase Negativa

Campo	**Valor**
Estado	✓CORRECTO
Predicción	0 (Negativo)
Etiqueta Real	0 (Negativo)
Texto (extracto)	"inside electric light orchestra 1970-73 disappointed. expected more performances with the band. not what i expected..."
Vecinos (5/5)	Todos etiquetados como 0
Términos comunes	"what", "money", "more", "not"

6.3 Análisis de Predicciones Incorrectas

Se analizaron 10 ejemplos incorrectamente clasificados para identificar patrones de error:

Cuadro 6: Ejemplo de Predicción Incorrecta

Campo	**Valor**
Estado	× INCORRECTO
Predicción	0 (Negativo)
Etiqueta Real	1 (Positivo)
Texto (extracto)	"Good Product - received broken. So far I've used this product once, and found it somewhat easy to use..."
Vecinos (5/5)	Todos etiquetados como 0
Motivo probable	Presencia de palabras negativas ("broken") domina sobre el sentimiento positivo

7 Compresión y Destilación

7.1 Configuración del modelo

La clase `ModelCompressor` es la encargada de comprimir el modelo Teacher (RoBERTa) en un modelo Student más pequeño (DistilBERT).

Configuración que hemos utilizado:

- Modelo Teacher: RoBERTa-base (124.6M parámetros).
- Modelo Student: DistilBERT-base-uncased (66.9M parámetros).
- Temperatura: 2.0.
- Alpha (pérdida): 0.5.
- Épocas: 3.
- Batch size: 16.

Lo más interesante es cómo la función de pérdida combina las predicciones de ambos modelos:

$$\mathcal{L}_{total} = \alpha \cdot \mathcal{L}_{distillation} + (1 - \alpha) \cdot \mathcal{L}_{student} \tag{2}$$

7.2 Progreso de la pérdida a lo largo del Entrenamiento

Época	Train Loss	Val Loss	Val Acc	Val F1
1	0.3169	0.4003	0.9167	0.9163
2	0.2973	0.3947	0.9245	0.9244
3	0.2882	0.3914	0.9356	0.9355

7.3 Comparación Teacher vs Student

Cuadro 7: Comparación Completa entre Teacher y Student

Métrica	Teacher (RoBERTa)	Student (DistilBERT)
Accuracy	0.9634	0.9359
Macro F1	0.9634	0.9358
Parámetros (M)	124.65	66.96
Tamaño (MB)	475.49	255.41
Tiempo Inferencia (s)	2.59	1.61

Cuadro 8: Métricas de Compresión

Métrica	Valor
Ratio de Compresión	1.86x
Pérdida de Accuracy	-2.75 %
Pérdida de F1	-2.76 %
Speedup	1.61x

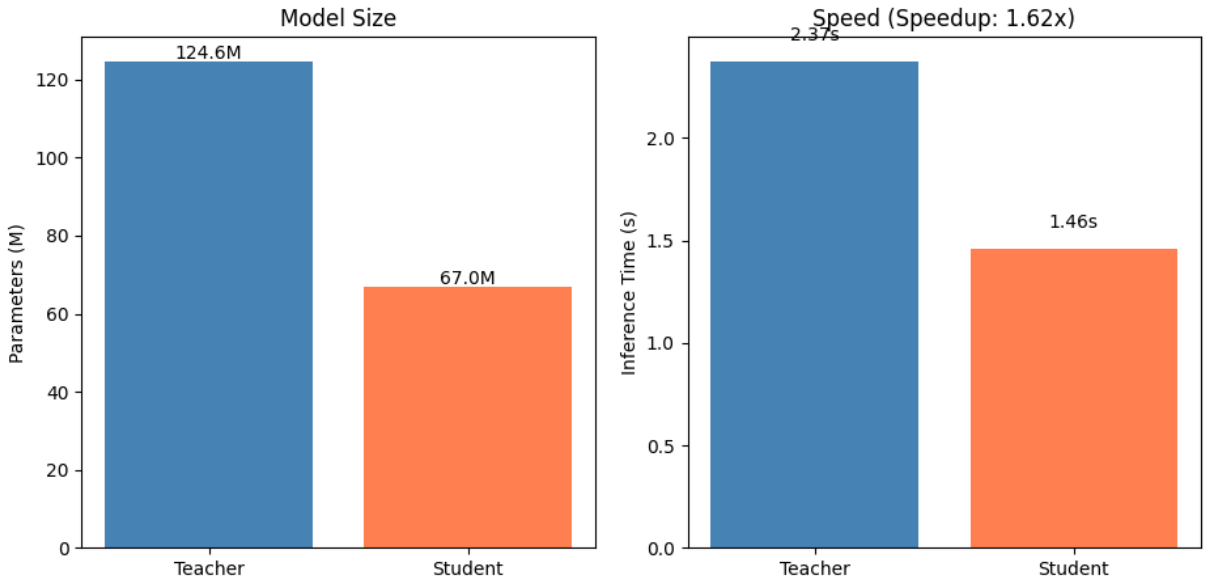

Figura 4: Comparación entre el modelo Teacher y Student

Como se puede observar, el modelo destilado ha sido un gran logro. Hemos conseguido mucha más velocidad, una reducción muy significativa de los pesos y todo esto se ha logrado manteniendo los buenos resultados del modelo teacher.

8 Generador de Resúmenes

8.1 Configuración del modelo

La clase `SummarizationExplainer` ha utilizado el modelo T5-small para generar resúmenes explicativos de las predicciones realizadas y de las características de cada clase.

Configuración del modelo:

- Nombre del Modelo: T5-small.
- Longitud máxima de entrada: 512 tokens.
- Longitud máxima de salida: 150 tokens.
- Device: CUDA.

8.2 Métodos implementados por la clase

1. **Resumen global de cada clase**: El modelo se basa en los datos para generar una descripción de las características típicas de cada clase.
2. **Explicaciones para cada predicción**: Para cada predicción, genera una explicación en lenguaje natural basada en los vecinos recuperados para así dar una explicación de en qué se ha fijado.
3. **Análisis de Alucinaciones**: Detecta posibles fabricaciones en las explicaciones generadas comparando con los textos fuente.

8.3 Integración con Explicabilidad

El generador de resúmenes se integra con `ExplainabilityModule` para proporcionar explicaciones más ricas:

Cuadro 9: Tipos de Explicación Generados

Tipo	Descripción
Case-Based	Basada en la distribución de etiquetas de los vecinos y términos comunes.
LLM-Generated	Explicación en lenguaje natural generada por T5 combinando información de vecinos.
Revisión de alucinación	Comprueba que al generar el resumen el modelo no alucine.

9 Tiempos de ejecución de los modelos

Debemos recalcar que todos los modelos han sido entrenados con la misma GPU. Esto es importante porque para que sea una comparativa válida debemos usar siempre la misma arquitectura.

Tarea	**Tiempo Empleado**
Carga y preprocesamiento de datos	$\approx$ 2 minutos
Entrenamiento del modelo transformer (teacher, 3 épocas)	$\approx$ 240 minutos
Generación de embeddings	$\approx$ 10-15 minutos
Construcción del índice k-NN	$\approx$ 1-2 minutos
Entrenamiento del modelo destilado (student, 3 épocas)	$\approx$ 135 minutos
Generación de resúmenes	$\approx$ 5-10 minutos
Total	$\approx$ **400 minutos**

10 Conclusiones

Gracias a esta práctica hemos sido capaces de entender e implementar los conceptos vistos en clase. Además, hemos puesto en práctica estos resolviendo un problema real como es la clasificación de sentimiento en reviews:

10.1 Descubrimientos

1. **El enfoque híbrido (RAG) es superior**: La combinación de Transformer y k-NN con $\alpha = 0.5$ consigue el mejor rendimiento (96.58 % accuracy), superando a ambos métodos de forma individual.
2. **k-NN basado en retrieval**: Sin necesidad de entrenamiento adicional, basándonos en los embeddings, el clasificador k-NN supera ligeramente al modelo Transformer (96.51 % vs 96.34 %).
3. **La destilación funciona**: Hemos sido capaces de ver cómo es posible conseguir una compresión de 1.86x con speedup de 1.61x, perdiendo solo 2.75 % de accuracy.

10.2 Limitaciones Encontradas

- Textos con señales mixtas o ironías son difíciles de clasificar correctamente.
- Las explicaciones de los LLM pueden contener alucinaciones por eso debemos jugar bien con el parámetro de la temperatura para que no alucine de forma exagerada pero manteniendo la creatividad.

11 Hilo de Gemini

Hilo generado: https://gemini.google.com/share/fabab433265d