

Proyecto Práctico - Redes Neuronales

Clasificación de Sentimientos en Reseñas de Amazon
con Arquitecturas MLP, RNN, LSTM, GRU y Transformer

Herney Eduardo Quintero Trochez

Código: 201528556

Universidad del Valle

Facultad de Ingeniería

Programa de Ingeniería de Sistemas

Diciembre 2025

Resumen

Este proyecto analiza el desempeño de cinco familias de arquitecturas de redes neuronales para clasificación de sentimientos en reseñas de productos de Amazon. Se trabajó con un subconjunto balanceado de 200,000 reseñas en inglés, clasificadas en cinco categorías (1 a 5 estrellas). Se implementaron y evaluaron modelos de Perceptrón Multicapa (MLP) con representaciones Bag-of-Words y Embeddings, redes recurrentes simples (SimpleRNN), redes con memoria (LSTM y GRU bidireccionales), y modelos basados en la arquitectura Transformer Encoder. Los experimentos se realizaron en PyTorch y TensorFlow utilizando GPUs NVIDIA. Los resultados muestran que los modelos secuenciales (GRU: 61.98 %, LSTM: 61.72 %) superan significativamente a los MLP (53–59 %), mientras que el Transformer alcanzó un 61.32 % de accuracy. Todos los modelos presentaron mayor dificultad para clasificar las clases intermedias (2-4 estrellas), logrando mejor precisión en las clases extremas (1 y 5 estrellas). El proyecto demuestra la importancia de capturar la estructura secuencial del texto para tareas de análisis de sentimientos.

Índice

1. Introducción	6
1.1. Objetivos	6
1.1.1. Objetivo General	6
1.1.2. Objetivos Específicos	6
2. Metodología	6
2.1. Dataset: Amazon Reviews Multi	6
2.1.1. Características del Dataset	6
2.1.2. Partición de Datos	7
2.1.3. Distribución de Clases	7
2.2. Preprocesamiento de Texto	7
2.2.1. Preprocesamiento para MLP con Bag-of-Words	7
2.2.2. Preprocesamiento para Modelos Secuenciales y Transformer	7
2.3. Configuración de Entrenamiento	8
2.4. Métricas de Evaluación	8
3. Arquitecturas de Modelos	8
3.1. Perceptrón Multicapa (MLP)	8
3.1.1. MLP con Bag-of-Words (BoW)	8
3.1.2. MLP con Embeddings	9
3.1.3. Resumen de Experimentos MLP	9
3.2. Redes Neuronales Recurrentes	9
3.2.1. SimpleRNN	9
3.2.2. LSTM (Long Short-Term Memory)	9
3.2.3. GRU (Gated Recurrent Unit)	9
3.2.4. Resumen de Experimentos Recurrentes	10
3.3. Transformer Encoder	10
3.3.1. Variantes Implementadas	10
3.3.2. Resumen de Experimentos Transformer	10
4. Resultados y Análisis	11
4.1. Resultados por Familia de Modelos	11
4.1.1. Perceptrón Multicapa (MLP)	11
4.1.2. Redes Recurrentes (SimpleRNN, LSTM, GRU)	14
4.1.3. Transformer Encoder	18
4.2. Comparación Global de Modelos	19
4.3. Análisis por Clase	21
4.4. Análisis de Errores	22
5. Conclusiones	23
5.1. Hallazgos Principales	23
5.2. Limitaciones del Estudio	24
5.3. Trabajo Futuro	24

Índice de figuras

1.	Curvas de entrenamiento MLP BoW (TensorFlow). Se observa overfitting temprano con divergencia train/val.	11
2.	Curvas de entrenamiento MLP Embedding (TensorFlow). Mejor generalización que el modelo BoW.	12
3.	Métricas por clase MLP BoW. Las clases extremas (1 y 5 estrellas) muestran mejor rendimiento.	12
4.	Métricas por clase MLP Embedding. Patrón similar al BoW pero con mejores valores absolutos.	13
5.	Comparativa de variantes MLP BoW (TensorFlow vs PyTorch).	13
6.	Comparativa de variantes MLP Embedding (TensorFlow vs PyTorch).	13
7.	Curvas de entrenamiento GRU Bidireccional. Convergencia estable con early stopping efectivo.	14
8.	Curvas de entrenamiento LSTM Bidireccional. Comportamiento similar al GRU.	15
9.	Curvas de entrenamiento SimpleRNN Bidireccional. Se observa mayor inestabilidad en validación.	15
10.	Métricas por clase SimpleRNN Bidireccional.	15
11.	Métricas por clase LSTM Bidireccional.	16
12.	Métricas por clase GRU Bidireccional.	16
13.	Comparativa de variantes SimpleRNN entrenadas.	17
14.	Comparativa de variantes LSTM entrenadas.	17
15.	Comparativa de todas las variantes GRU entrenadas.	17
16.	Curvas de entrenamiento del Transformer Encoder (configuración grande). El warmup inicial es visible en las primeras épocas.	18
17.	Métricas por clase del mejor Transformer.	19
18.	Comparativa de todas las configuraciones Transformer probadas.	19
19.	Comparación de accuracy y F1-macro de todos los mejores modelos por familia. GRU obtiene el mejor rendimiento global.	20
20.	Comparación de tiempos de entrenamiento. El Transformer requiere significativamente más tiempo debido a su mayor complejidad.	20
21.	Mapa de calor del F1-Score por clase para todos los modelos. Se observa el patrón consistente de mejor rendimiento en clases extremas.	21
22.	Matriz de confusión del modelo GRU. Se observa la diagonal pronunciada para clases 1 y 5, y mayor dispersión en clases intermedias.	22
23.	Matrices de confusión de todos los mejores modelos. Se observa un patrón consistente: las clases extremas (1 y 5 estrellas) son clasificadas con mayor precisión, mientras que las clases intermedias presentan mayor confusión.	23

Índice de cuadros

1.	Distribución del dataset por conjuntos	7
2.	Distribución de clases en el conjunto de entrenamiento	7
3.	Configuración de todos los experimentos MLP	9
4.	Configuración de todos los experimentos recurrentes	10
5.	Configuración de todos los experimentos Transformer	10

6.	Resultados de modelos MLP en el conjunto de prueba	11
7.	Resultados de modelos recurrentes en el conjunto de prueba	14
8.	Resultados de modelos Transformer en el conjunto de prueba	18
9.	Comparacion global de los mejores modelos por familia	19
10.	F1-Score por clase - Modelo GRU_PyTorch_20251128_212324	21

1. Introducción

1.1. Objetivos

1.1.1. Objetivo General

Analizar y comparar el rendimiento de diferentes arquitecturas de redes neuronales en la tarea de clasificación de sentimientos en reseñas de productos, identificando las fortalezas y debilidades de cada enfoque para seleccionar la más adecuada según los requisitos del problema.

1.1.2. Objetivos Específicos

- Implementar y evaluar modelos de Perceptrón Multicapa (MLP) utilizando dos representaciones de texto: Bag-of-Words (BoW) y Embeddings aprendidos.
- Implementar y evaluar modelos secuenciales: SimpleRNN, LSTM bidireccional y GRU bidireccional, comparando su capacidad para capturar dependencias temporales.
- Implementar y evaluar modelos basados en la arquitectura Transformer Encoder, explorando configuraciones con embeddings aprendidos y preentrenados (GloVe).
- Comparar el rendimiento de todos los modelos utilizando métricas estándar como Accuracy, F1-Score macro y matrices de confusión.
- Analizar el comportamiento de los modelos por clase, identificando patrones de error y dificultades particulares.

2. Metodología

2.1. Dataset: Amazon Reviews Multi

Para este proyecto se utilizó el **Amazon Reviews Multi Dataset** [1], un corpus de reseñas reales de productos disponible públicamente. Este dataset contiene reseñas en seis idiomas (inglés, japonés, alemán, francés, español y chino), cada una con una calificación de 1 a 5 estrellas.

2.1.1. Características del Dataset

- **Fuente:** Kaggle / ACL Anthology (EMNLP 2020)
- **Idioma utilizado:** Inglés (filtrado de la colección multiidioma)
- **Campos principales:**
 - **review_title:** Título de la reseña
 - **review_body:** Cuerpo completo de la reseña
 - **stars:** Calificación de 1 a 5 estrellas (variable objetivo)
 - **language:** Idioma de la reseña

2.1.2. Partición de Datos

El dataset se dividió en tres conjuntos de la siguiente manera:

Cuadro 1: Distribución del dataset por conjuntos

Conjunto	Muestras	Porcentaje
Entrenamiento	200,000	95.2 %
Validación	5,000	2.4 %
Prueba	5,000	2.4 %
Total	210,000	100 %

2.1.3. Distribución de Clases

El conjunto de entrenamiento presenta una distribución balanceada, con 40,000 muestras por clase:

Cuadro 2: Distribución de clases en el conjunto de entrenamiento

Clase (Estrellas)	Muestras	Porcentaje
1 *	40,000	20 %
2 *	40,000	20 %
3 *	40,000	20 %
4 *	40,000	20 %
5 *	40,000	20 %

2.2. Preprocesamiento de Texto

Se implementaron dos estrategias de preprocesamiento según el tipo de modelo:

2.2.1. Preprocesamiento para MLP con Bag-of-Words

1. Combinación del título y cuerpo de la reseña en un solo texto
2. Vectorización TF-IDF con los siguientes parámetros:
 - Vocabulario máximo: 5,000 términos
 - Frecuencia mínima de documento: 3
 - Frecuencia máxima de documento: 85 %
3. Resultado: matriz dispersa de $200,000 \times 5,000$

2.2.2. Preprocesamiento para Modelos Secuenciales y Transformer

1. Combinación del título y cuerpo de la reseña
2. Tokenización a nivel de palabra
3. Construcción de vocabulario (máximo 50,000 tokens)

4. Conversión a secuencias de índices
5. Padding/truncado a longitud fija (150-200 tokens según el modelo)
6. Resultado: tensor de secuencias de dimensión [batch_size, max_length]

2.3. Configuración de Entrenamiento

Todos los modelos fueron entrenados siguiendo prácticas estándar de deep learning:

- **Optimizadores:** Adam para MLP/RNN/LSTM/GRU, AdamW para Transformer
- **Función de pérdida:** CrossEntropyLoss (con pesos de clase opcionales)
- **Early Stopping:** Detención temprana basada en la pérdida de validación (pacien-cia: 5-10 épocas)
- **Learning Rate Scheduling:** ReduceLROnPlateau o warmup + decay lineal (Trans-former)
- **Gradient Clipping:** Norma máxima de gradientes entre 1.0 y 1.5
- **Hardware:** GPUs NVIDIA GeForce RTX 3060 (12GB) y RTX 5070 (12GB)

2.4. Métricas de Evaluación

Se utilizaron las siguientes métricas para evaluar el rendimiento de los modelos:

- **Accuracy:** Proporción de predicciones correctas sobre el total.
- **F1-Score Macro:** Promedio no ponderado del F1-Score por clase, adecuado para datasets balanceados.
- **Precisión y Recall por clase:** Para analizar el comportamiento diferenciado en cada categoría de estrellas.
- **Matriz de confusión:** Para visualizar patrones de error entre clases.

3. Arquitecturas de Modelos

3.1. Perceptrón Multicapa (MLP)

Se implementaron dos variantes de MLP para establecer líneas base del problema: una con representación Bag-of-Words (TF-IDF) y otra con embeddings aprendidos.

3.1.1. MLP con Bag-of-Words (BoW)

Este modelo utiliza una representación TF-IDF del texto como entrada directa a una red fully-connected. La arquitectura consiste en una capa de entrada que recibe vectores TF-IDF de 5,000 dimensiones, seguida de tres capas ocultas (256, 128, 64 neuronas) con activación ReLU y Dropout de 0.3 entre cada capa, finalizando con una capa de salida Softmax de 5 clases.

3.1.2. MLP con Embeddings

Este modelo aprende representaciones densas de las palabras mediante una capa de Embedding de dimensión 300, seguida de un Global Average Pooling para obtener una representación fija del documento. Posteriormente, tres capas densas (256, 128, 64 neuronas) procesan esta representación hasta la capa de clasificación final.

3.1.3. Resumen de Experimentos MLP

Cuadro 3: Configuración de todos los experimentos MLP

Experimento	Framework	Tipo	Emb.	Params	Ckpt
MLP_BoW_TF_20250929	TensorFlow	BoW	–	1.32M	val_loss
MLP_BoW_PT_20251128	PyTorch	BoW	–	2.60M	val_loss
MLP_BoW_PT_20251130	PyTorch	BoW	–	2.60M	F1-macro
MLP_Emb_TF_20250929	TensorFlow	Emb	300	14.84M	val_loss
MLP_Emb_PT_20251128	PyTorch	Emb	300	14.92M	val_loss
MLP_Emb_PT_20251130	PyTorch	Emb	300	14.68M	F1-macro

Hiperparámetros comunes: Capas ocultas [256, 128, 64], Dropout 0.3–0.5, LR 0.001–0.0005, Batch 512, Épocas 50, Patience 10.

3.2. Redes Neuronales Recurrentes

Se implementaron tres variantes de arquitecturas recurrentes en PyTorch: SimpleRNN, LSTM y GRU, todas bidireccionales.

3.2.1. SimpleRNN

Red recurrente básica bidireccional para capturar contexto secuencial.

3.2.2. LSTM (Long Short-Term Memory)

Las redes LSTM utilizan mecanismos de compuertas (input, forget, output) para capturar dependencias a largo plazo, resolviendo el problema del vanishing gradient.

3.2.3. GRU (Gated Recurrent Unit)

Las redes GRU son una simplificación de LSTM que combina las compuertas en una sola de “update”, reduciendo la complejidad computacional.

3.2.4. Resumen de Experimentos Recurrentes

Cuadro 4: Configuración de todos los experimentos recurrentes

Experimento	Tipo	Emb	Hid	Cap	Bid	Params	Ckpt
SimpleRNN_PT_20251128	RNN	200	256	2	Si	10.3M	val_loss
SimpleRNN_PT_20251130	RNN	200	256	2	Si	10.3M	F1-macro
LSTM_PT_20251128_192500	LSTM	128	128	2	Si	3.2M	val_loss
LSTM_PT_20251128_210359	LSTM	256	256	2	Si	15.0M	val_loss
LSTM_PT_20251130	LSTM	256	256	2	Si	15.0M	F1-macro
GRU_PT_20251128_182500	GRU	128	128	2	Si	3.1M	val_loss
GRU_PT_20251128_212324	GRU	200	200	2	Si	10.9M	val_loss
GRU_PT_20251130	GRU	200	200	2	Si	10.9M	F1-macro

Hiperparámetros comunes: Dropout 0.3, LR 0.001, Batch 48–64, MaxLen 150–200, Gradient Clip 1.0, Patience 5.

3.3. Transformer Encoder

El modelo Transformer utiliza mecanismos de auto-atención (self-attention) para capturar relaciones entre todas las posiciones de la secuencia simultáneamente, permitiendo paralelización completa durante el entrenamiento.

3.3.1. Variantes Implementadas

Se exploraron múltiples configuraciones:

- **Base:** d_model=256, 8 cabezas, 4 capas, FFN=512, pooling CLS
- **Grande:** d_model=512, 8 cabezas, 6 capas, FFN=2048, pooling Mean
- **GloVe:** Embeddings GloVe-200d congelados, 4 cabezas, 4 capas

3.3.2. Resumen de Experimentos Transformer

Cuadro 5: Configuración de todos los experimentos Transformer

Experimento	d_m	H	L	FFN	Pool	LR	Params	Ckpt
Transf_PT_20251128	256	8	4	512	CLS	1e-4	14.5M	val_loss
Transf_PT_20251129	256	8	4	512	CLS	1e-4	14.5M	val_loss
Transf_PT_20251130_01	512	8	6	2048	Mean	5e-5	43.6M	val_loss
Transf_GloVe_Frozen_20251130	200	4	4	800	CLS	1e-4	11.6M	F1-macro
Transf_PT_20251130_17	256	8	4	512	CLS	1e-4	14.5M	F1-macro

Hiperparámetros comunes: MaxLen 200, Dropout 0.1, Gradient Clip 1.0, Warmup 1000–2000 steps, Patience 5–7.

4. Resultados y Análisis

4.1. Resultados por Familia de Modelos

4.1.1. Perceptrón Multicapa (MLP)

Los modelos MLP sirvieron como línea base para el proyecto. La Tabla 6 resume los resultados obtenidos.

Cuadro 6: Resultados de modelos MLP en el conjunto de prueba

Modelo	Test Acc.	F1 Macro	Tiempo	Ckpt
MLP_BoW_TF_20250929	53.1 %	0.524	33s	val_loss
MLP_BoW_PT_20251128	50.1 %	0.502	28s	val_loss
MLP_Emb_TF_20250929	59.1 %	0.588	289s	val_loss
MLP_Emb_PT_20251128	58.9 %	0.587	95s	val_loss

El modelo MLP con BoW alcanzó 50–53 % de accuracy, limitado por la pérdida de información secuencial inherente a la representación BoW. El uso de embeddings mejoró significativamente el rendimiento (~59 %), ya que permite capturar relaciones semánticas entre palabras.

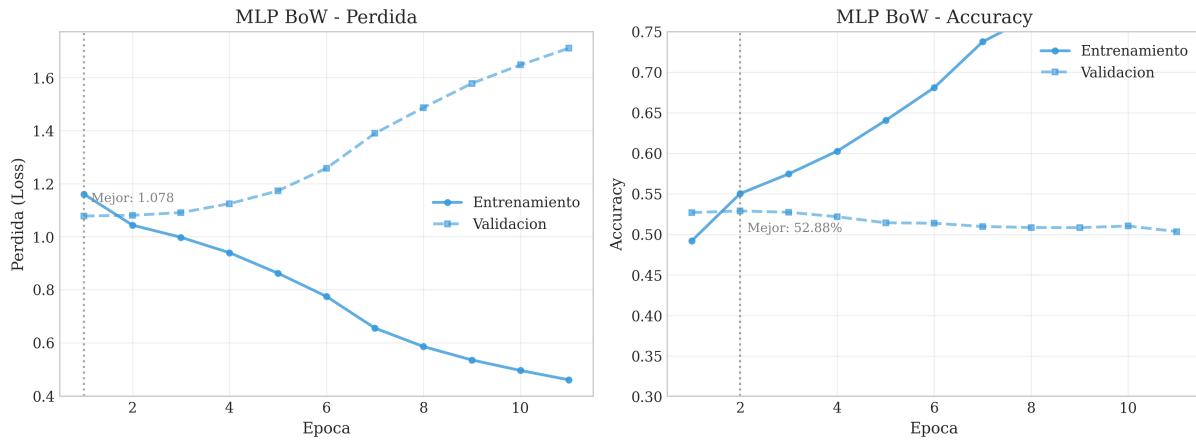


Figura 1: Curvas de entrenamiento MLP BoW (TensorFlow). Se observa overfitting temprano con divergencia train/val.

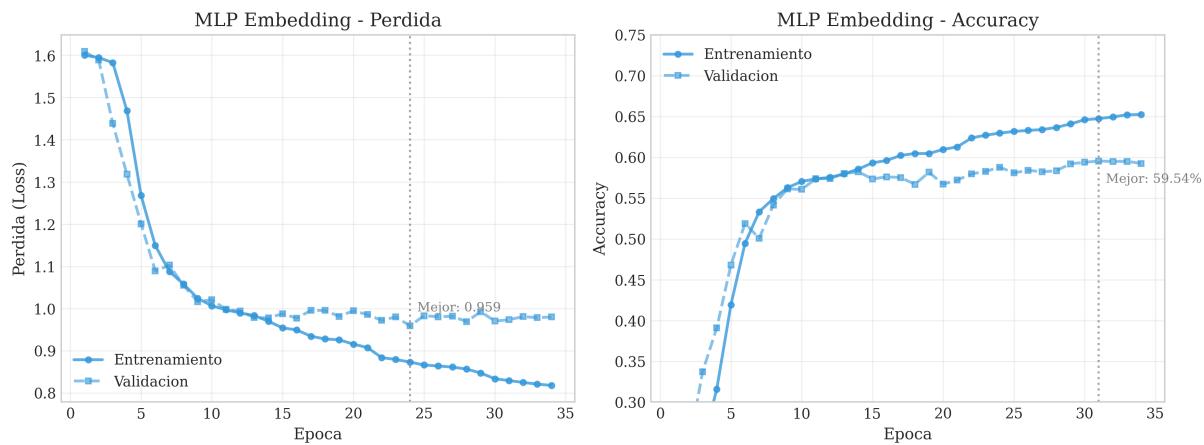


Figura 2: Curvas de entrenamiento MLP Embedding (TensorFlow). Mejor generalización que el modelo BoW.

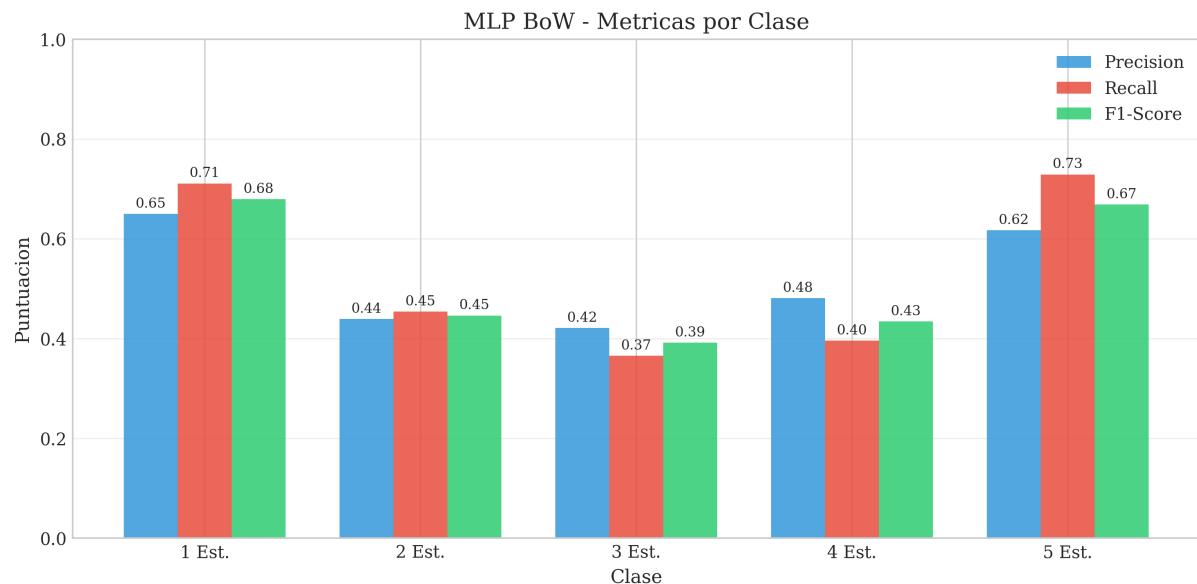


Figura 3: Métricas por clase MLP BoW. Las clases extremas (1 y 5 estrellas) muestran mejor rendimiento.

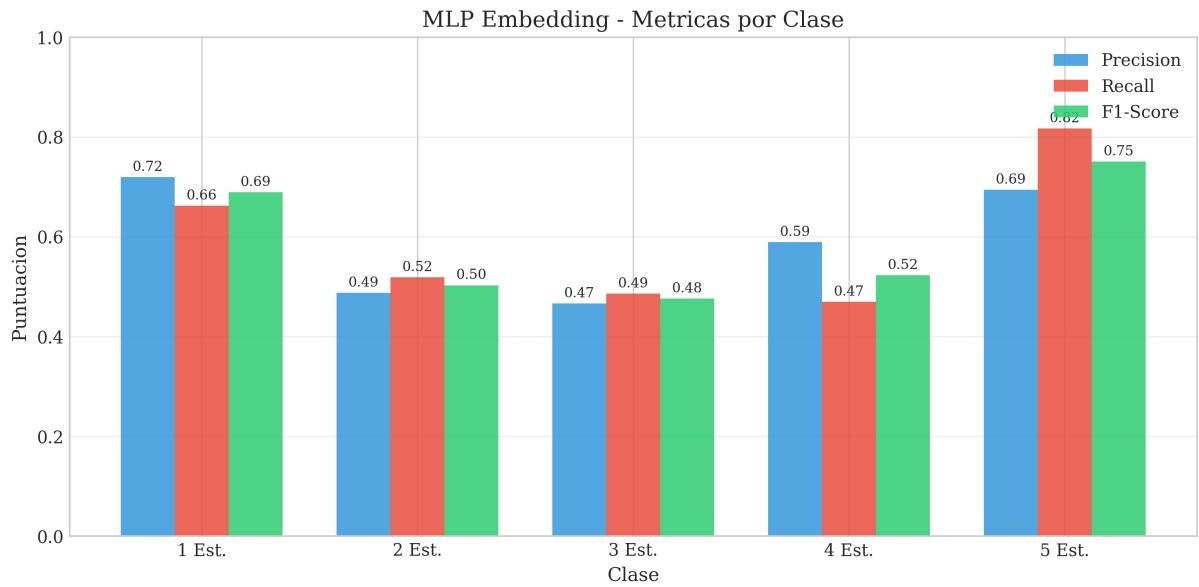


Figura 4: Métricas por clase MLP Embedding. Patrón similar al BoW pero con mejores valores absolutos.

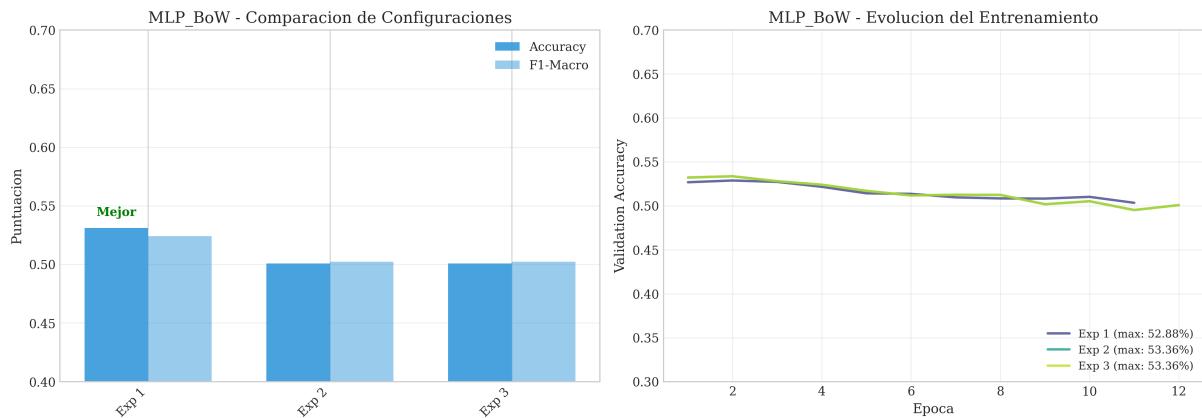


Figura 5: Comparativa de variantes MLP BoW (TensorFlow vs PyTorch).

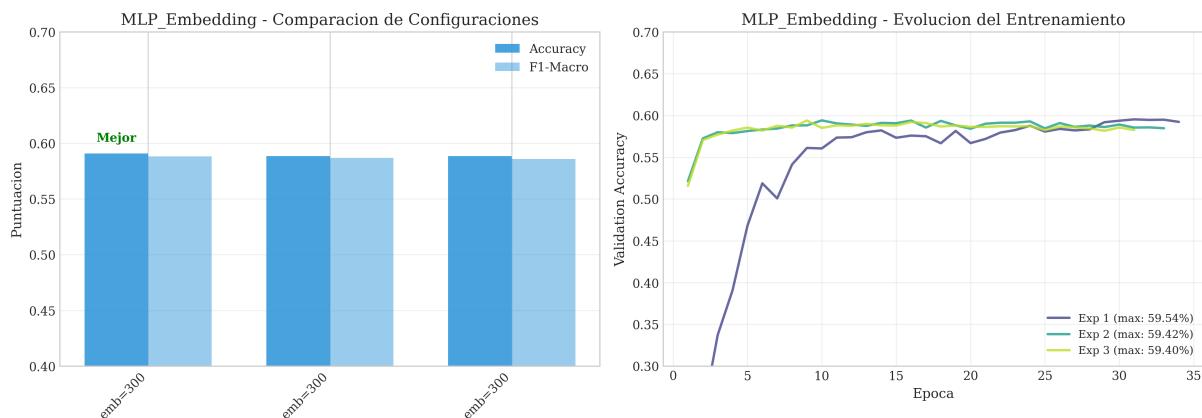


Figura 6: Comparativa de variantes MLP Embedding (TensorFlow vs PyTorch).

4.1.2. Redes Recurrentes (SimpleRNN, LSTM, GRU)

Los modelos recurrentes mostraron mejoras consistentes sobre los MLP, demostrando la importancia de capturar la estructura secuencial del texto.

Cuadro 7: Resultados de modelos recurrentes en el conjunto de prueba

Modelo	Test Acc.	F1 Macro	Ep.	Tiempo	Ckpt
SimpleRNN_PT_20251128	60.0 %	0.595	27	12.5 min	val_loss
LSTM_PT_20251128_210359	61.7 %	0.619	19	24.7 min	val_loss
GRU_PT_20251128_212324	62.0 %	0.614	15	13.8 min	val_loss

Los resultados muestran que:

- SimpleRNN alcanzó 59.98 % de accuracy, superando a los MLP pero mostrando cierta inestabilidad en el entrenamiento debido a problemas de vanishing gradients.
- LSTM y GRU obtuvieron resultados similares (61.72 % y 61.98 % respectivamente), siendo GRU ligeramente más eficiente computacionalmente (menos parámetros y menor tiempo de entrenamiento).
- La bidireccionalidad resultó crucial, permitiendo al modelo considerar tanto el contexto anterior como el posterior de cada token.

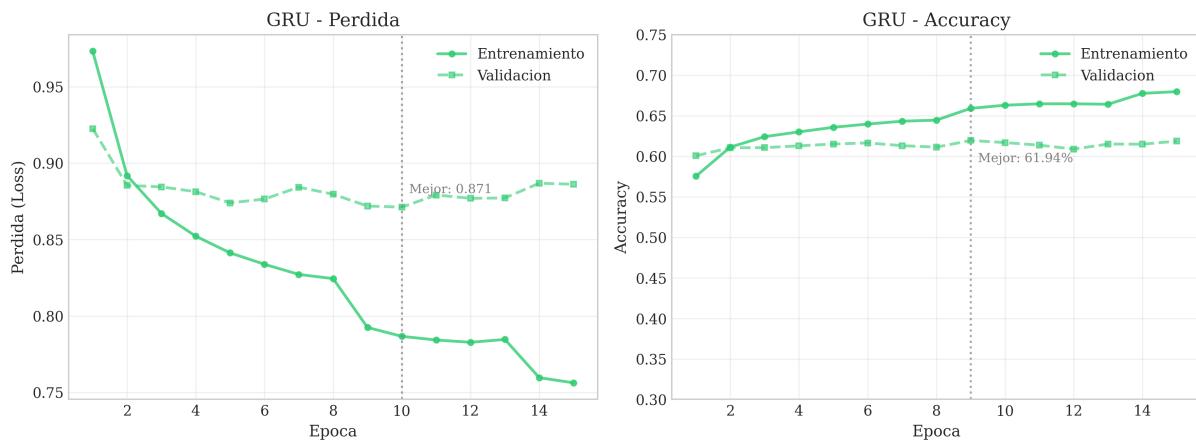


Figura 7: Curvas de entrenamiento GRU Bidireccional. Convergencia estable con early stopping efectivo.

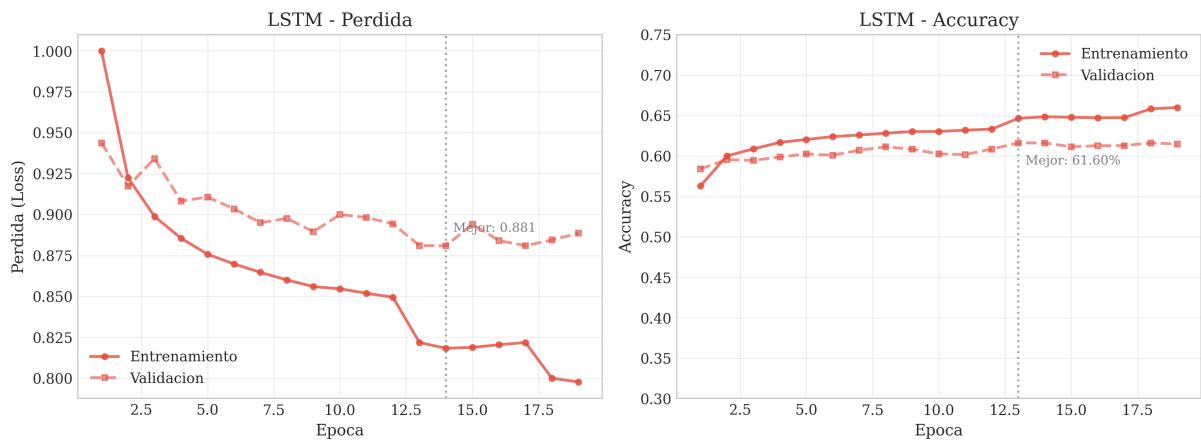


Figura 8: Curvas de entrenamiento LSTM Bidireccional. Comportamiento similar al GRU.

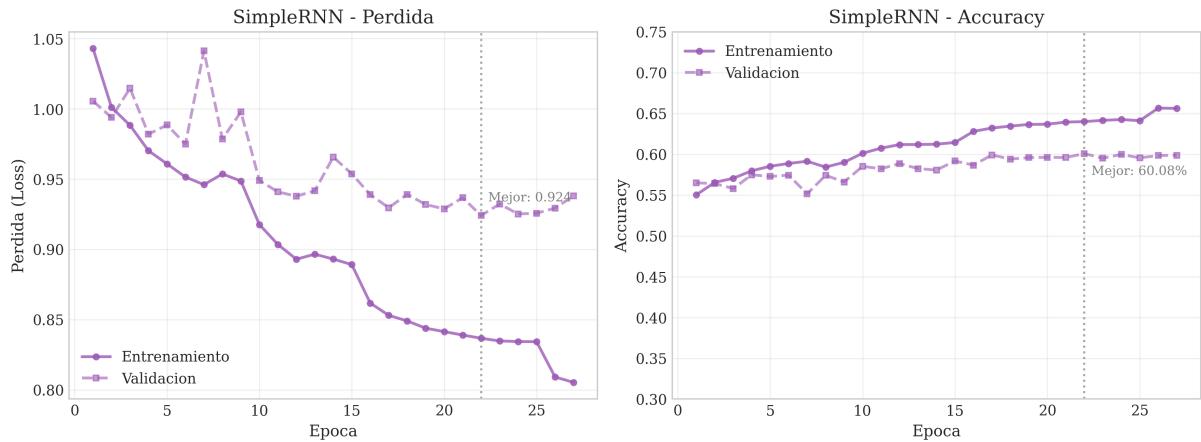


Figura 9: Curvas de entrenamiento SimpleRNN Bidireccional. Se observa mayor inestabilidad en validación.

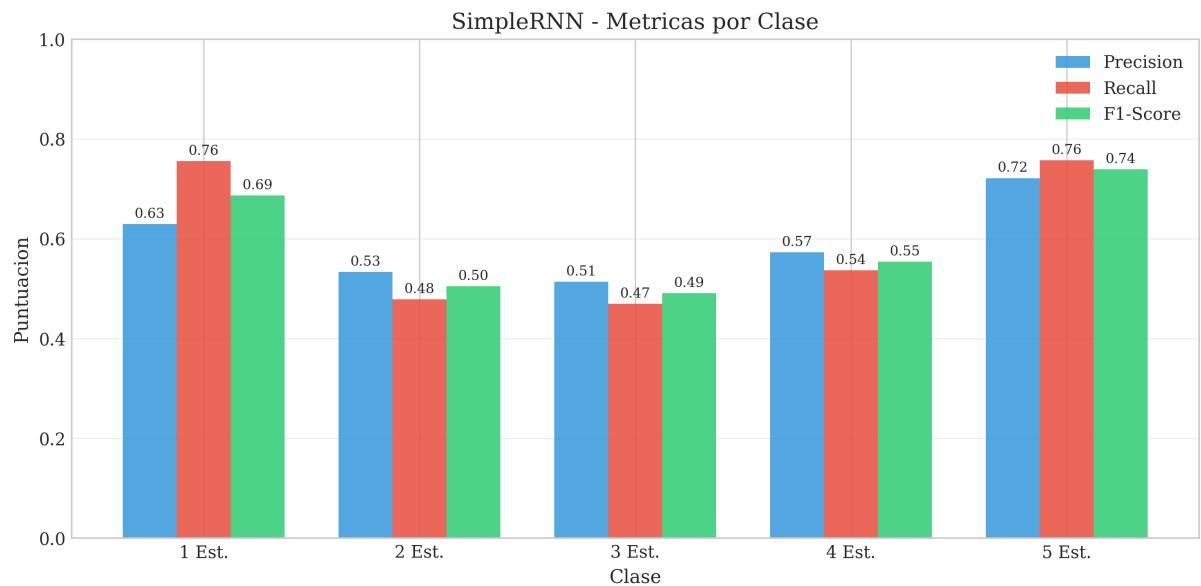


Figura 10: Métricas por clase SimpleRNN Bidireccional.

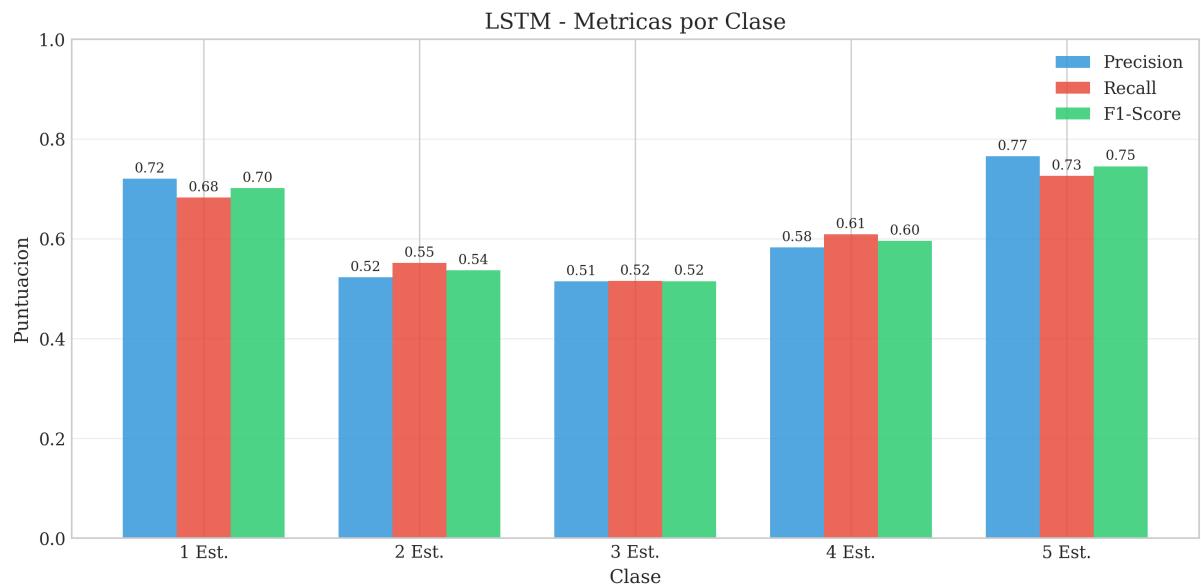


Figura 11: Métricas por clase LSTM Bidireccional.



Figura 12: Métricas por clase GRU Bidireccional.

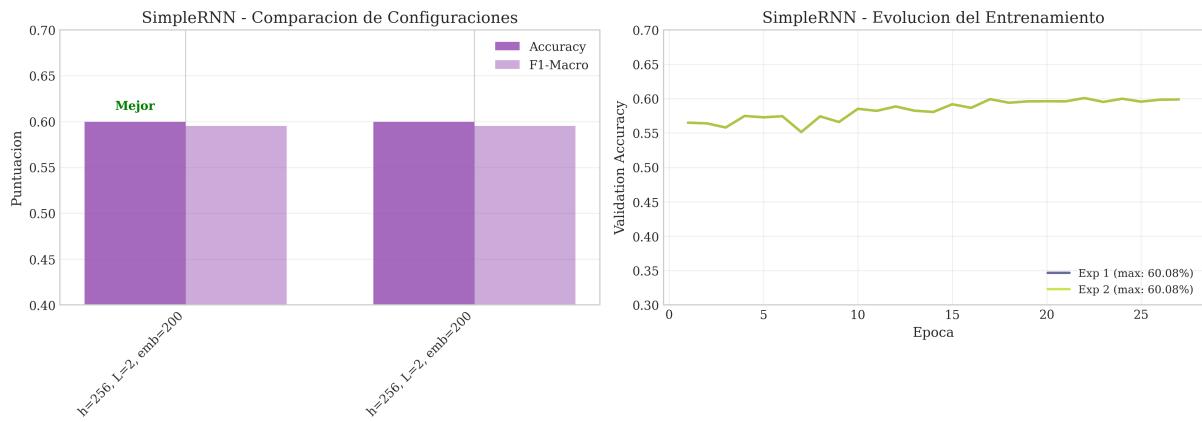


Figura 13: Comparativa de variantes SimpleRNN entrenadas.

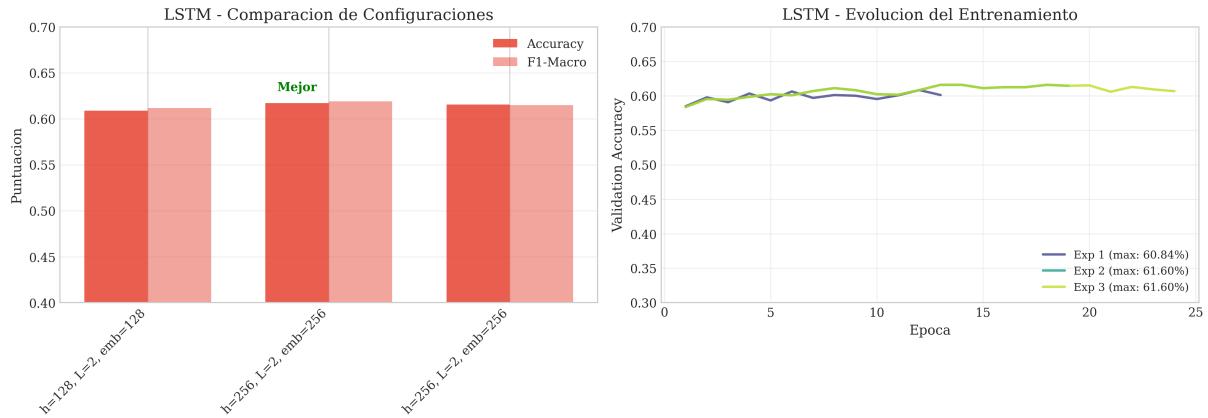


Figura 14: Comparativa de variantes LSTM entrenadas.

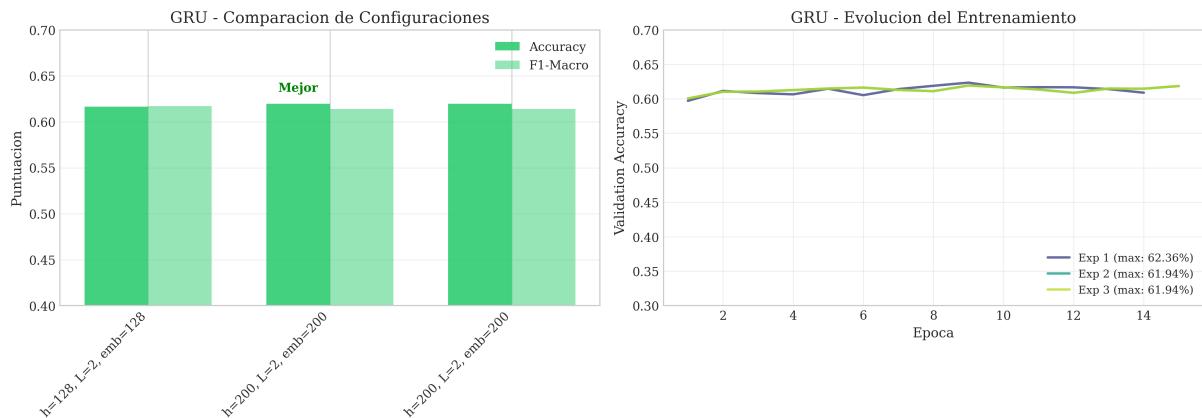


Figura 15: Comparativa de todas las variantes GRU entrenadas.

4.1.3. Transformer Encoder

Cuadro 8: Resultados de modelos Transformer en el conjunto de prueba

Configuracion	Test Acc.	F1 Macro	Ep.	Tiempo	Ckpt
Transf_PT_20251128	60.4 %	0.600	30	1.08 h	val_loss
Transf_PT_20251129	60.5 %	0.606	35	1.21 h	val_loss
Transf_PT_20251130_01	61.3 %	0.607	45	6.55 h	val_loss
Transf_GloVe_Frozen_20251130	60.0 %	0.597	50	1.26 h	F1-macro
Transf_PT_20251130_17	61.0 %	0.607	40	1.42 h	F1-macro

Observaciones sobre el Transformer:

- El modelo base (256d, 4 capas) alcanzó 60.4 % de accuracy, comparable a LSTM/-GRU.
- Aumentar el tamaño del modelo (512d, 6 capas) mejoró ligeramente el rendimiento (61.3 %), pero con un costo computacional significativamente mayor (6.5 horas vs. 1 hora).
- Los embeddings GloVe congelados no aportaron mejoras (60.0 %), posiblemente porque el dataset es suficientemente grande para aprender representaciones específicas del dominio.
- El cambio de criterio de checkpoint a F1-macro no produjo mejoras significativas.

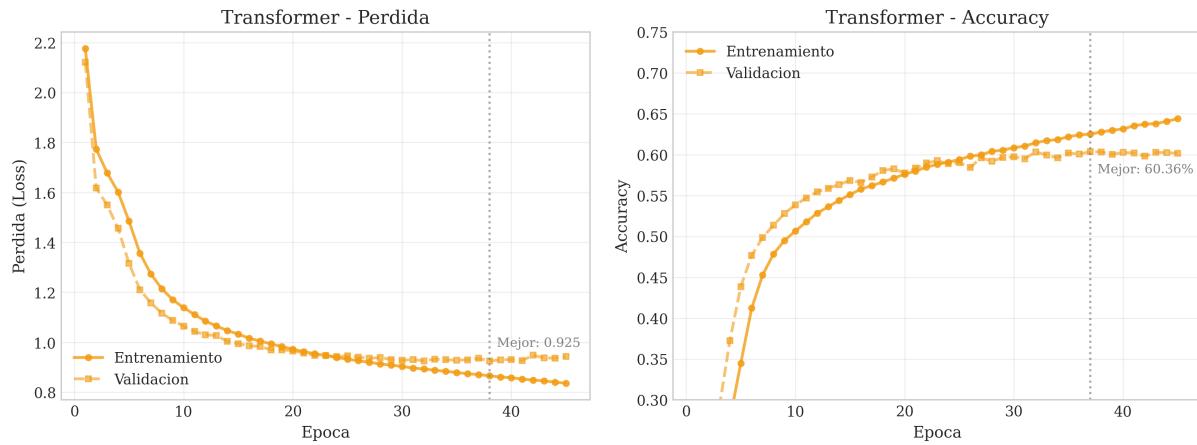


Figura 16: Curvas de entrenamiento del Transformer Encoder (configuración grande). El warmup inicial es visible en las primeras épocas.

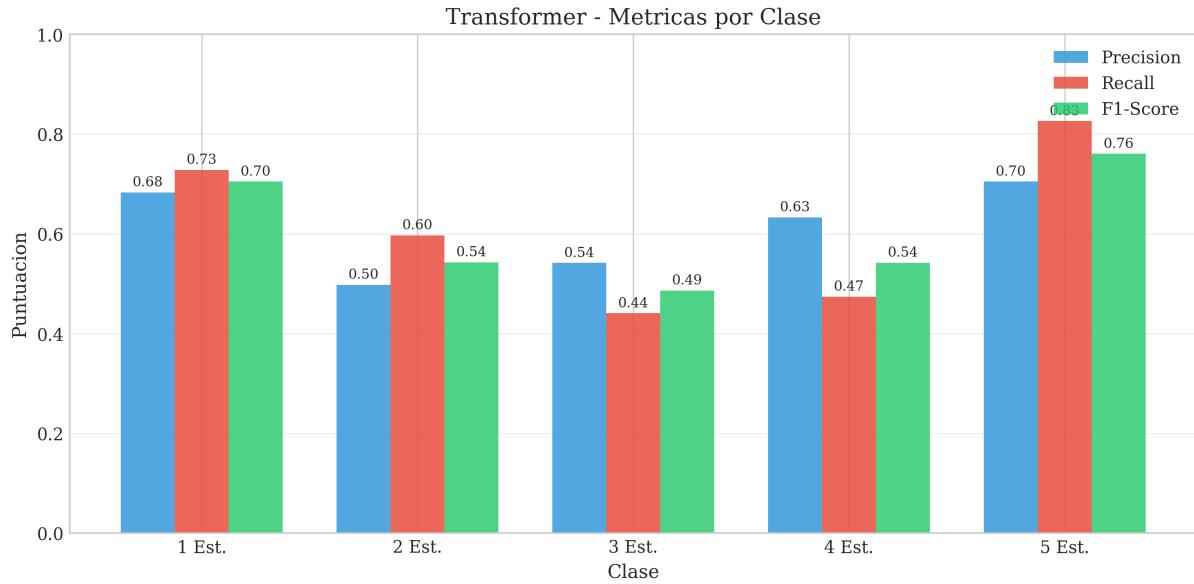


Figura 17: Métricas por clase del mejor Transformer.

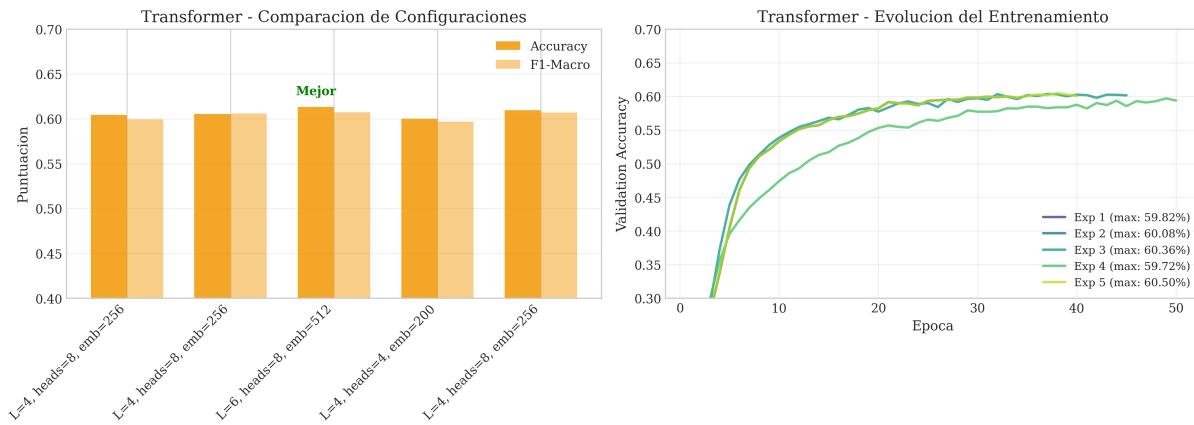


Figura 18: Comparativa de todas las configuraciones Transformer probadas.

4.2. Comparación Global de Modelos

La Tabla 9 presenta una comparación consolidada de los mejores modelos de cada familia:

Cuadro 9: Comparacion global de los mejores modelos por familia

Modelo	Test Acc.	F1 Macro	Params.	Tiempo	Ckpt
MLP_BoW_TF_20250929	53.12 %	0.524	1.3M	33s	val_loss
MLP_Emb_TF_20250929	59.08 %	0.588	14.8M	289s	val_loss
SimpleRNN_PT_20251128	59.98 %	0.595	10.3M	12.5min	val_loss
LSTM_PT_20251128_210359	61.72 %	0.619	15.0M	24.7min	val_loss
GRU_PT_20251128_212324	61.98 %	0.614	10.9M	13.8min	val_loss
Transf_PT_20251130_01	61.32 %	0.607	43.6M	6.5h	val_loss

Nota sobre el criterio de checkpoint: Durante el entrenamiento, se guarda el mejor modelo según el criterio indicado. Los experimentos con criterio “val_loss” guardan el

checkpoint cuando la pérdida de validación alcanza su mínimo, mientras que los experimentos con criterio “F1-macro” lo hacen cuando el F1-Score macro de validación alcanza su máximo. Esto puede afectar el rendimiento final del modelo seleccionado.

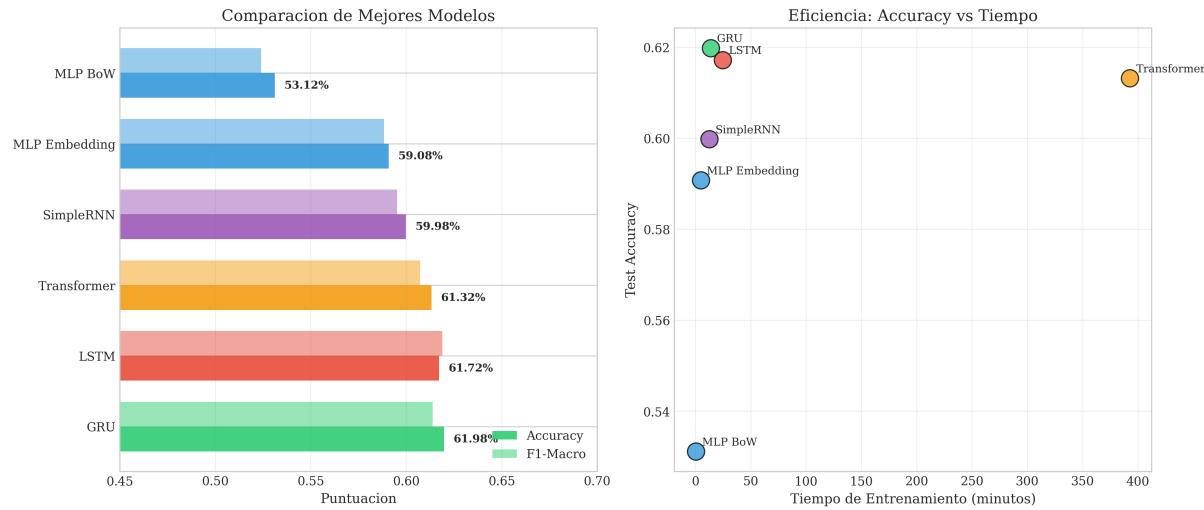


Figura 19: Comparación de accuracy y F1-macro de todos los mejores modelos por familia. GRU obtiene el mejor rendimiento global.

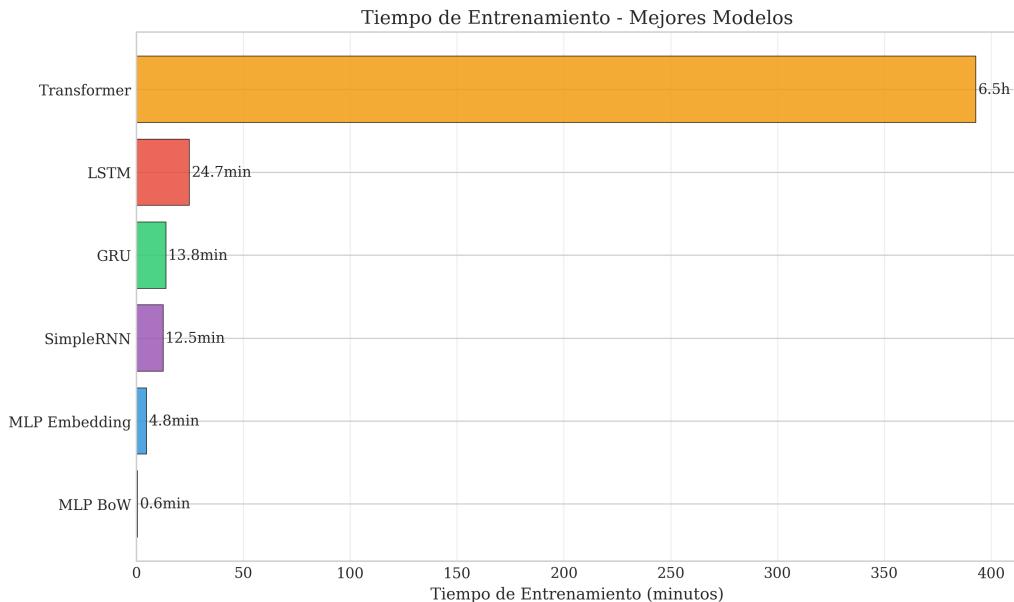


Figura 20: Comparación de tiempos de entrenamiento. El Transformer requiere significativamente más tiempo debido a su mayor complejidad.



Figura 21: Mapa de calor del F1-Score por clase para todos los modelos. Se observa el patrón consistente de mejor rendimiento en clases extremas.

4.3. Análisis por Clase

Un patrón consistente en todos los modelos es la diferencia de rendimiento entre clases. La Tabla 10 muestra el F1-Score por clase del mejor modelo (GRU_PyTorch_20251128_212324):

Cuadro 10: F1-Score por clase - Modelo GRU_PyTorch_20251128_212324

Clase	Precision	Recall	F1-Score	Observación
1★	0.648	0.776	0.706	Alta recall
2★	0.540	0.508	0.524	Confusión con 1★ y 3★
3★	0.546	0.466	0.503	Clase más difícil
4★	0.609	0.550	0.578	Confusión con 3★ y 5★
5★	0.722	0.799	0.758	Mejor rendimiento

Interpretación: Las clases extremas (1 y 5 estrellas) son clasificadas con mayor precisión porque tienden a contener un lenguaje más polarizado y distintivo. Las clases intermedias (2, 3 y 4 estrellas) presentan mayor confusión porque:

1. El lenguaje utilizado es más ambiguo y matizado.
2. Las diferencias entre una reseña de 3 estrellas y una de 4 estrellas pueden ser sutiles.
3. Existe una superposición semántica natural entre opiniones moderadas.

4.4. Análisis de Errores

Se observaron los siguientes patrones de error:

- **Confusión entre clases adyacentes:** Los errores más comunes ocurren entre clases consecutivas (ej: predecir 4★ cuando la etiqueta es 3★).
- **Sesgo hacia clases extremas:** Los modelos tienden a predecir con más confianza las clases 1★ y 5★.
- **Dificultad con reseñas mixtas:** Reseñas que contienen tanto aspectos positivos como negativos son particularmente difíciles de clasificar.

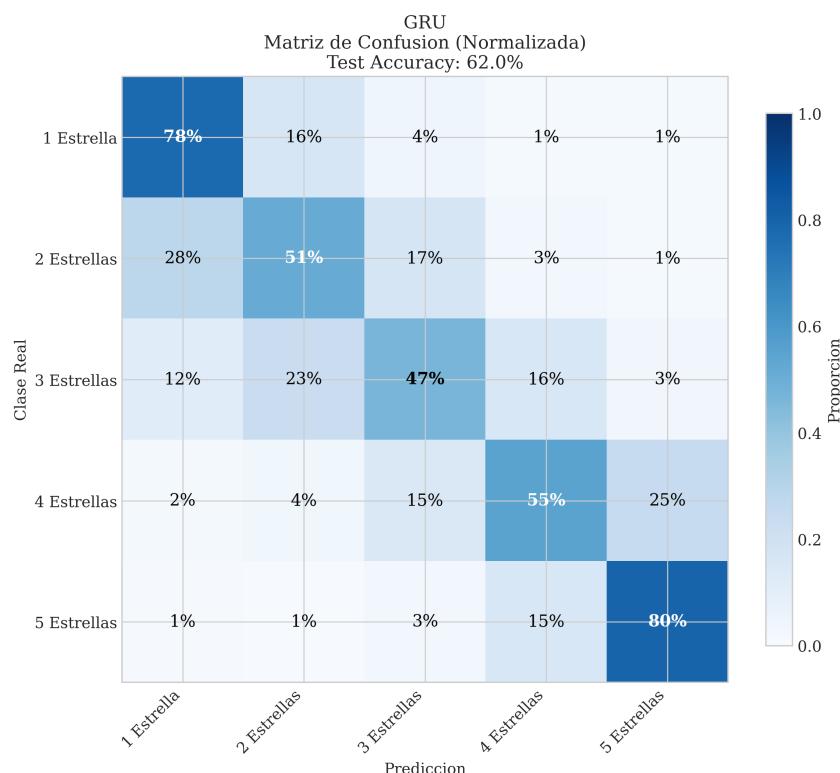


Figura 22: Matriz de confusión del modelo GRU. Se observa la diagonal pronunciada para clases 1 y 5, y mayor dispersión en clases intermedias.



Figura 23: Matrices de confusión de todos los mejores modelos. Se observa un patrón consistente: las clases extremas (1 y 5 estrellas) son clasificadas con mayor precisión, mientras que las clases intermedias presentan mayor confusión.

5. Conclusiones

5.1. Hallazgos Principales

Este proyecto permitió obtener las siguientes conclusiones:

- Superioridad de modelos secuenciales:** Los modelos que capturan la estructura secuencial del texto (RNN, LSTM, GRU, Transformer) superan consistentemente a los modelos que ignoran el orden de las palabras (MLP con BoW). Esto confirma la importancia del contexto secuencial para el análisis de sentimientos.
- Equivalencia práctica entre LSTM y GRU:** Ambas arquitecturas alcanzaron resultados similares (61.72 % y 61.98 % accuracy respectivamente), siendo GRU más eficiente computacionalmente. Para este tipo de tarea, GRU representa una excelente relación costo-beneficio.
- Transformer competitivo pero costoso:** El Transformer Encoder alcanzó resultados comparables a LSTM/GRU (61.32 % vs. 61.98 %), pero con un costo computacional significativamente mayor. Su ventaja de paralelización no se traduce en mejor rendimiento para este tamaño de dataset.
- Desafío de clases intermedias:** Todos los modelos mostraron dificultad para distinguir las clases 2, 3 y 4 estrellas. Esto sugiere que la ambigüedad inherente en

opiniones moderadas es un límite fundamental que no se resuelve solo con arquitecturas más complejas.

5. **Embeddings aprendidos vs. preentrenados:** Los embeddings GloVe no mejoraron el rendimiento del Transformer, indicando que con 200,000 ejemplos de entrenamiento el modelo puede aprender representaciones específicas del dominio tan efectivas como las preentrenadas.

5.2. Limitaciones del Estudio

- Se trabajó únicamente con texto en inglés; el comportamiento podría variar en otros idiomas.
- No se exploraron técnicas de data augmentation ni modelos de lenguaje grandes (BERT, RoBERTa).
- El límite de ~62 % de accuracy podría estar relacionado con la calidad intrínseca del etiquetado (ruido en las etiquetas).

5.3. Trabajo Futuro

- **Fine-tuning de modelos preentrenados:** Explorar BERT, RoBERTa o DistilBERT con fine-tuning para potencialmente superar el límite actual de ~62 %.
- **Reformulación del problema:** Convertir la clasificación de 5 clases en una tarea de regresión o en clasificación binaria (positivo/negativo) podría mejorar los resultados.
- **Análisis multimodal:** Incorporar información adicional como el título del producto, categoría o características del revisor.
- **Análisis de atención:** Utilizar los pesos de atención del Transformer para interpretar qué partes del texto influyen más en la clasificación.

Referencias

- [1] Keung, P., Lu, Y., Szarvas, G., & Smith, N. A. (2020). The Multilingual Amazon Reviews Corpus. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4563-4568. <https://aclanthology.org/2020.emnlp-main.369/>
- [2] Paszke, A., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems 32*, pp. 8024-8035. <https://pytorch.org/>
- [3] Abadi, M., et al. (2016). TensorFlow: A System for Large-Scale Machine Learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265-283. <https://www.tensorflow.org/>