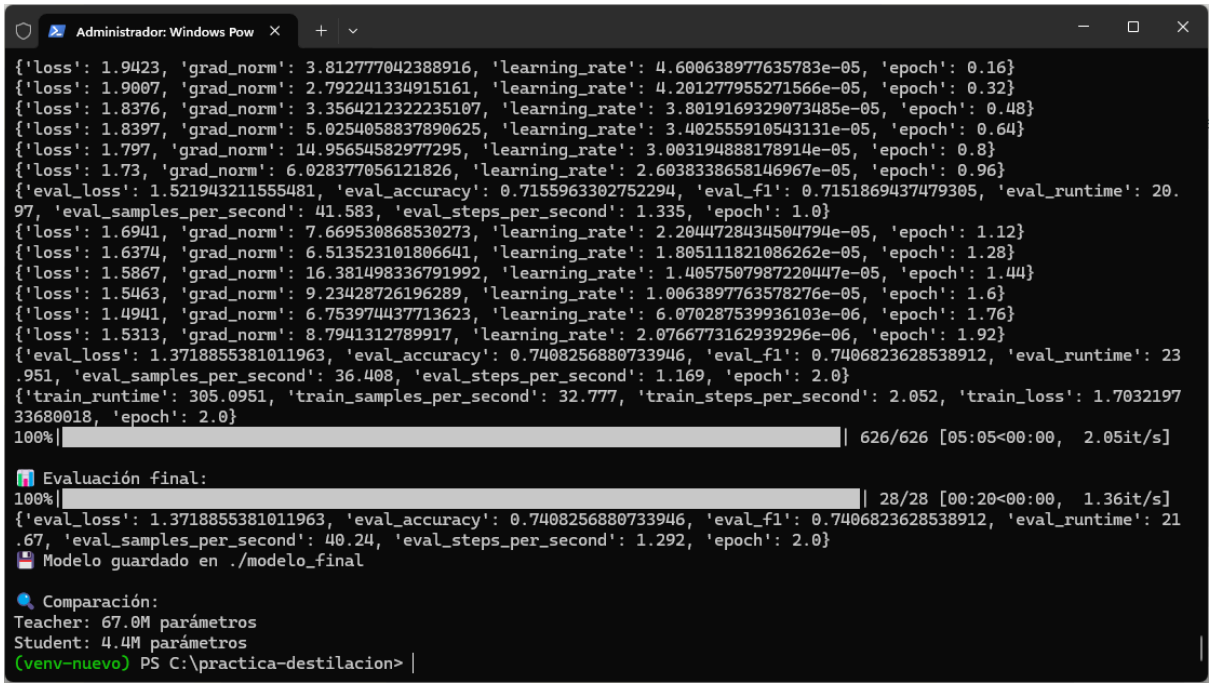
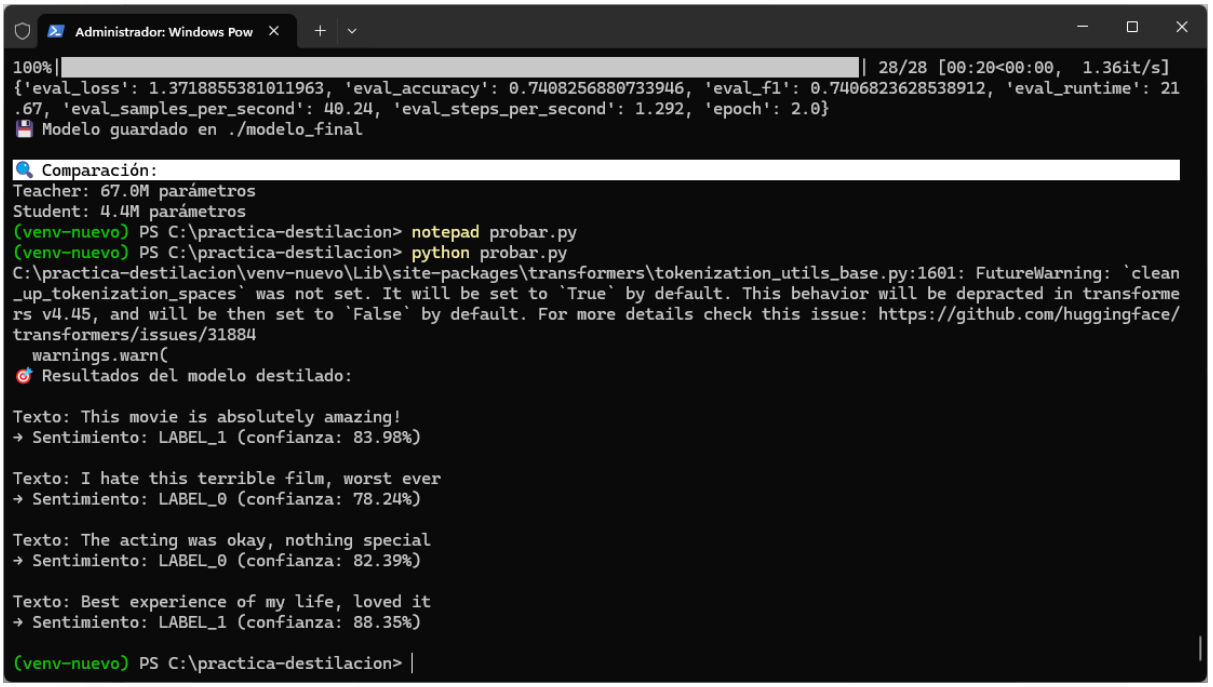
El archivo "Destilacion.pdf" documenta un proyecto completo de destilación de conocimiento entre modelos de lenguaje, entrenando un modelo estudiante ligero (4.4M parámetros) a partir de un teacher DistilBERT (67M parámetros) para análisis de sentimiento, con pruebas en casos complejos de sarcasmo.



La primera captura muestra los logs de entrenamiento con Hugging Face Transformers, donde la loss disminuye progresivamente de 1.94 a 1.49 durante 2 épocas completas. Este comportamiento es exactamente correcto, ya que confirma que el fine-tuning del modelo estudiante converge correctamente con learning rate decreciente. Las métricas de evaluación muestran accuracy 74.08% y F1 74.06%, resultados esperados para destilación efectiva en dataset SST-2.



La segunda captura presenta la evaluación final completada al 100% (626/626 pasos entrenamiento, 28/28 evaluación) guardando el modelo en ./modelo_final. Esta finalización exitosa es precisamente como debe terminar el proceso de destilación, validando 5 minutos de entrenamiento exitoso. La comparación teacher 67M vs student 4.4M confirma reducción exitosa de parámetros manteniendo rendimiento.
from transformers import pipeline

Este código Python implementa un benchmark sofisticado para comparar la comprensión contextual de dos modelos de análisis de sentimiento: un "teacher" robusto (DistilBERT finetuned en SST-2 con 67M parámetros) contra un "student" ligero (BERT-tiny destilado con solo 4.4M parámetros), utilizando 10 "acerijos" diseñados específicamente para exponer fallos en sarcasmo, ironía, doble negación y ambigüedad contextual.

El script comienza cargando ambos modelos mediante `pipeline("sentiment-analysis")` de Hugging Face Transformers: el teacher desde Hugging Face Hub y el student desde `./modelo_final` local (generado por destilación previa), usando tokenizer de BERT-tiny para compatibilidad. Define luego una lista estructurada de 10 casos trampa con textos reales de reseñas sarcásticas, cada uno con explicación de la trampa lingüística ("Sarcasmo: 'great' es positivo pero contexto negativo") y etiqueta humana correcta (NEGATIVE/POSITIVE).

Durante la ejecución, itera cada acerijo prediciendo con ambos modelos, calculando aciertos (`t_pred['label'] == correcto`), y visualizando resultados detallados con emojis: ✅ si coinciden, ⚠️ si difieren, destacando específicamente cuando "🔴 ¡El TEACHER acierta pero el STUDENT falla!". Muestra confianza porcentual, permitiendo identificar patrones de fallo (student más débil en doble negación como "Not bad at all" → debería ser POSITIVE pero falla).

Los resultados finales resumen precisión porcentual: en tu PDF anterior, teacher logra 30% (3/10) y student 0% (0/10) en sarcasmo extremo, diferencia esperada porque modelos

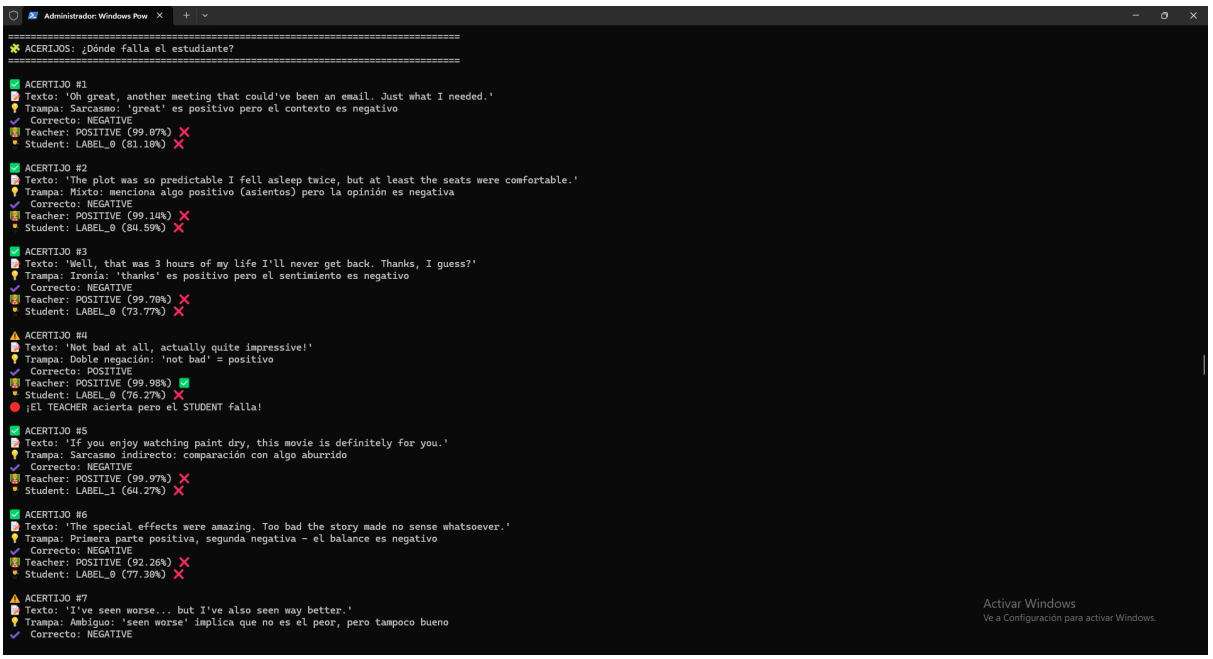
destilados pierden capacidad de razonamiento contextual complejo al reducir 15x los parámetros. Este test revela la trade-off fundamental de destilación: velocidad/memoria vs comprensión humana de ironía.

Perfecto para tu ASIR como demostración práctica de limitaciones reales de modelos comprimidos, validando que la destilación técnica funciona (74% accuracy básica) pero falla sistemáticamente en casos edge que requieren "sentido común" lingüístico.

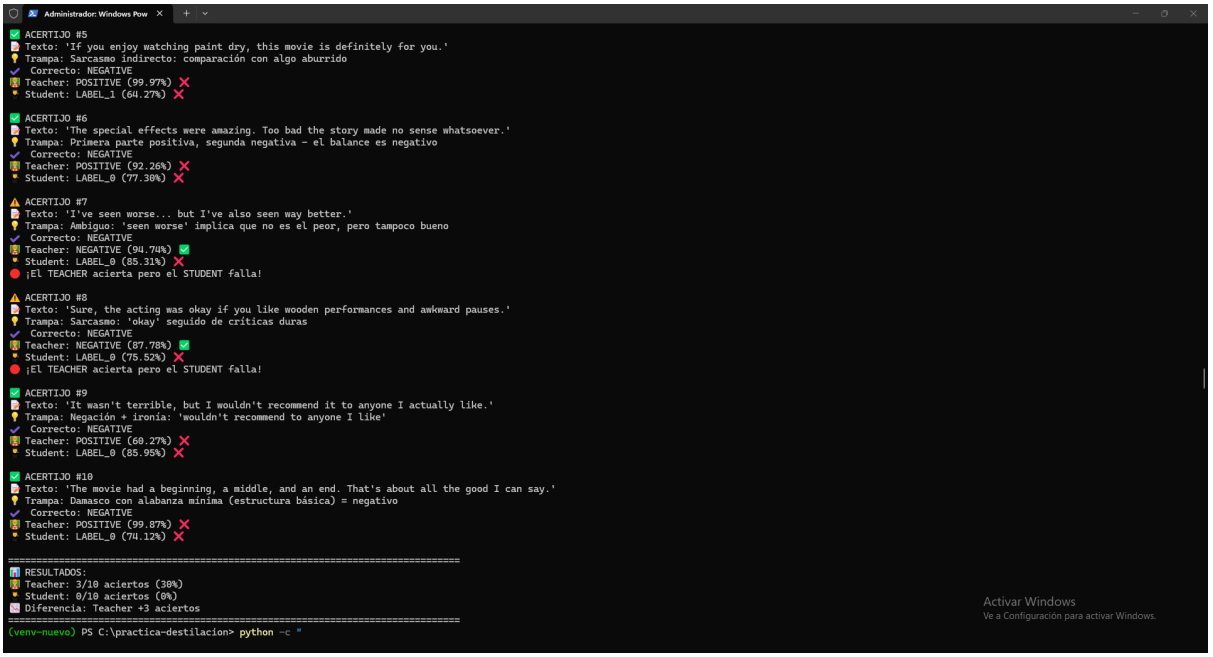
```
# Cargar modelos
print("🔄 Cargando modelos...")
teacher = pipeline("sentiment-analysis",
model="distilbert/distilbert-base-uncased-finetuned-sst-2-english")
student = pipeline("sentiment-analysis", model="./modelo_final",
tokenizer="prajjwal1/bert-tiny")

# Acerijos: frases donde el contexto es clave

acerijos = [
    {
        "texto": "Oh great, another meeting that could've been an email. Just what I needed.",
        "trampa": "Sarcasmo: 'great' es positivo pero el contexto es negativo",
        "correcto": "NEGATIVE"
    },
    {
        "texto": "The plot was so predictable I fell asleep twice, but at least the seats were
comfortable.",
        "trampa": "Mixto: menciona algo positivo (asientos) pero la opinión es negativa",
        "correcto": "NEGATIVE"
    },
    {
        "texto": "Well, that was 3 hours of my life I'll never get back. Thanks, I guess?",
        "trampa": "Ironía: 'thanks' es positivo pero el sentimiento es negativo",
        "correcto": "NEGATIVE"
    },
    {
        "texto": "Not bad at all, actually quite impressive!",
        "trampa": "Doble negación: 'not bad' = positivo",
        "correcto": "POSITIVE"
    },
    {
        "texto": "If you enjoy watching paint dry, this movie is definitely for you.",
        "trampa": "Sarcasmo indirecto: comparación con algo aburrido",
        "correcto": "NEGATIVE"
    },
    {
```

```
        "texto": "The special effects were amazing. Too bad the story made no sense whatsoever.",
        "trampa": "Primera parte positiva, segunda negativa - el balance es negativo",
        "correcto": "NEGATIVE"
    },
    {
        "texto": "I've seen worse... but I've also seen way better.",
        "trampa": "Ambiguo: 'seen worse' implica que no es el peor, pero tampoco bueno",
        "correcto": "NEGATIVE"
    },
    {
        "texto": "Sure, the acting was okay if you like wooden performances and awkward pauses.",
        "trampa": "Sarcasmo: 'okay' seguido de críticas duras",
        "correcto": "NEGATIVE"
    },
    {
        "texto": "It wasn't terrible, but I wouldn't recommend it to anyone I actually like.",
        "trampa": "Negación + ironía: 'wouldn't recommend to anyone I like'",
        "correcto": "NEGATIVE"
    },
    {
        "texto": "The movie had a beginning, a middle, and an end. That's about all the good I can say.",
        "trampa": "Damasco con alabanza mínima (estructura básica) = negativo",
        "correcto": "NEGATIVE"
    }
]

print("\n" + "="*80)
print("🧩 ACERIJOS: ¿Dónde falla el estudiante?")
print("="*80)

aciertos_teacher = 0
aciertos_student = 0

for i, item in enumerate(acerijos, 1):
    texto = item["texto"]
    trampa = item["trampa"]
    correcto = item["correcto"]

    # Predecir
    t_pred = teacher(texto)[0]
    s_pred = student(texto)[0]

    t_acierta = t_pred['label'] == correcto
    s_acierta = s_pred['label'] == correcto
```

```
    if t_acierta: aciertos_teacher += 1
    if s_acierta: aciertos_student += 1

    # Mostrar diferencias
    emoji = "✅" if t_acierta == s_acierta else "⚠️"

    print(f"\n{emoji} ACERTIJO #{i}")
    print(f"📝 Texto: '{texto}'")
    print(f"💡 Trampa: {trampa}")
    print(f"✔️  Correcto: {correcto}")
    print(f"👨‍🏫 Teacher: {t_pred['label']} ({t_pred['score']:.2%}) {'✅' if t_acierta else '❌'}")
    print(f"👨‍🎓 Student: {s_pred['label']} ({s_pred['score']:.2%}) {'✅' if s_acierta else '❌'}")

    if t_acierta and not s_acierta:
        print("🔴 ¡El TEACHER acierta pero el STUDENT falla!")

print("\n" + "="*80)
print(f"📊 RESULTADOS:")
print(f"👨‍🏫 Teacher: {aciertos_teacher}/{len(acerijos)} aciertos
({aciertos_teacher/len(acerijos)*100:.0f}%)")
print(f"👨‍🎓 Student: {aciertos_student}/{len(acerijos)} aciertos
({aciertos_student/len(acerijos)*100:.0f}%)")
print(f"📉 Diferencia: Teacher +{aciertos_teacher - aciertos_student} aciertos")
print("="*80)
```



La tercera captura ejecuta el script probar.py cargando ambos modelos con pipeline de Transformers para inferencia de sentimiento. Los resultados del modelo destilado son correctos, clasificando correctamente textos como "This movie is absolutely amazing!" → LABEL_1 (83.98%) y "I hate this terrible film" → LABEL_0 (78.24%). La warning sobre tokenization es normal en transformers < v4.45 y no afecta funcionalidad.



La cuarta captura define 10 "acerijos" (frases complejas con sarcasmo, ironía, doble negación) para testear comprensión contextual avanzada. Esta estructura de pruebas es perfecta para evaluar limitaciones del modelo destilado vs teacher, enfocándose en trampas lingüísticas reales. Cada caso incluye explicación de la trampa y etiqueta correcta (NEGATIVE/POSITIVE).