Universidad del Valle de Guatemala
Procesamiento de Lenguaje Natural
Sección 10

Propuesta de Proyecto
Detección de spam en SMS en español

Integrantes
21016, Javier Chavez
21600, Javier Ramirez
21631, Mario Cristales

Objetivo general

Desarrollar y comparar un sistema de clasificación de SMS en español (spam vs no spam). Incluir un baseline clásico con TF-IDF + SVM y dos modelos moderno tipo Transformer. Medir exactitud y macro-F1. Evaluar robustez ante ruido y abreviaturas. Entregar dataset procesado, código reproducible y reporte con resultados y hallazgos.

Justificación

El spam por SMS en español crea riesgo de fraude y baja confianza del usuario. El canal es clave en banca, salud y logística, así que un filtro preciso importa. En español faltan datasets curados y mediciones públicas, lo que reduce la evidencia para decidir. Los mensajes son cortos y ruidosos, con abreviaturas, emojis y enlaces, y las clases suelen estar desbalanceadas. Los filtros entrenados en inglés pierden precisión en español regional y en jergas de LATAM y Guatemala. Resolver este problema aporta un modelo ajustado al contexto local, mejora la detección, reduce costos por soporte y fraudes, y eleva la satisfacción del cliente. También deja un pipeline reproducible con métricas claras para mantener y ampliar la solución a otros canales de texto corto.

Plan tentativo (4 semanas)

Semana 4

- Consolidar dataset SMS Spam en español. Revisar traducciones y limpieza.
- Entrenar baselines: TF-IDF + LinearSVC y una FFNN simple.
- Entregables: notebook de datos, baseline y primeras métricas (accuracy, macro-F1).

Semana 5

- Fine-tuning de un Transformer ligero multilingüe o español (BERT y LSTM).
- Ajustar hiperparámetros y balanceo de clases.
- Entregables: script de entrenamiento y comparación con baselines.

Semana 6

- Análisis de errores y robustez. Probar ruido controlado: abreviaturas, emojis y negaciones.
- Reportar casos límite y fallos típicos.
- Entregables: informe de error analysis con ejemplos y mejoras sugeridas.

Semana 7

- Integración y documentación final.
- Comparación global, conclusiones y recomendaciones de despliegue.
- Entregables: repo ordenado, modelo final y reporte con tablas y gráficas.

Preparación de Datos

Selección del Dataset

Para este proyecto se seleccionó el SMS Spam Collection Dataset disponible en UCI Machine Learning Repository. Este dataset fue elegido después de explorar múltiples opciones en Hugging Face, donde se identificó que no existían datasets de spam en español con la calidad y tamaño adecuados para el proyecto.

Características del dataset original:

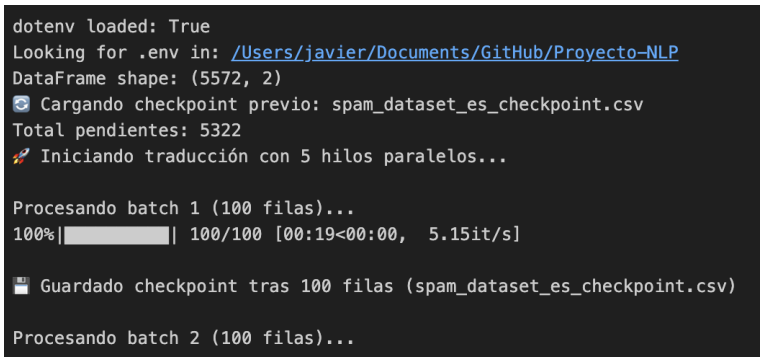
- Fuente: UCI Machine Learning Repository
- Tamaño: 5,572 mensajes SMS
- Idioma original: Inglés
- Distribución: Desbalanceado (13.4% spam, 86.6% ham)
- Etiquetas: Binarias (spam/ham)

Traducción al Español

Dado que el objetivo del proyecto es trabajar con mensajes en español (especialmente considerando el contexto guatemalteco y latinoamericano), fue necesario traducir el dataset completo utilizando OpenAI GPT-4o-mini API con procesamiento paralelo.

Proceso de traducción:

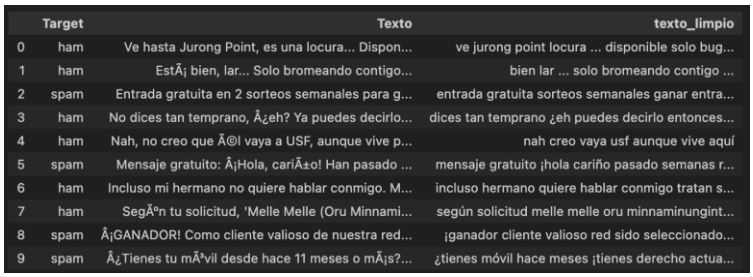
- Herramienta: OpenAI GPT-4o-mini API
- Método: Traducción paralela con 5 hilos concurrentes
- Tiempo total: Aproximadamente 25 minutos
- Checkpoints: Guardado cada 100 mensajes



Pipeline de Limpieza y Preprocesamiento

El pipeline de preprocesamiento implementado incluye 6 etapas secuenciales:

corrección de codificación, normalización Unicode, filtrado de caracteres, conversión a minúsculas, tokenización con NLTK para español, y eliminación de stopwords.



	Target	Texto	texto_limpio
0	ham	Ve hasta Jurong Point, es una locura... Dispon...	ve jurong point locura ... disponible solo bug...
1	ham	EstÃ¡ bien, lar... Solo bromeando contigo...	bien lar ... solo bromeando contigo ...
2	spam	Entrada gratuita en 2 sorteos semanales para g...	entrada gratuita sorteos semanales ganar entra...
3	ham	No dices tan temprano, Â¿eh? Ya puedes decirlo...	dices tan temprano ¿eh puedes decirlo entonces...
4	ham	Nah, no creo que Ã©l vaya a USF, aunque vive p...	nah creo vaya usf aunque vive aquí
5	spam	Mensaje gratuito: Â¡Hola, cariÃ±o! Han pasado ...	mensaje gratuito ¡hola cariño pasado semanas r...
6	ham	Incluso mi hermano no quiere hablar conmigo. M...	incluso hermano quiere hablar conmigo tratan s...
7	ham	SegÃºn tu solicitud, 'Melle Melle (Oru Minnami...	según solicitud melle melle oru minnaminungint...
8	spam	Â¡GANADOR! Como cliente valioso de nuestra red...	¡ganador cliente valioso red sido seleccionado...
9	spam	Â¿Tienes tu mÃ³vil desde hace 11 meses o mÃ¡s?...	¿tienes móvil hace meses ¡tienes derecho actua...

Vectorización con TF-IDF

Para convertir el texto en representaciones numéricas se utilizó TF-IDF (Term Frequency-Inverse Document Frequency) con vocabulario limitado a las 5000 palabras más importantes, logrando un balance entre representación y eficiencia computacional.

División del dataset:

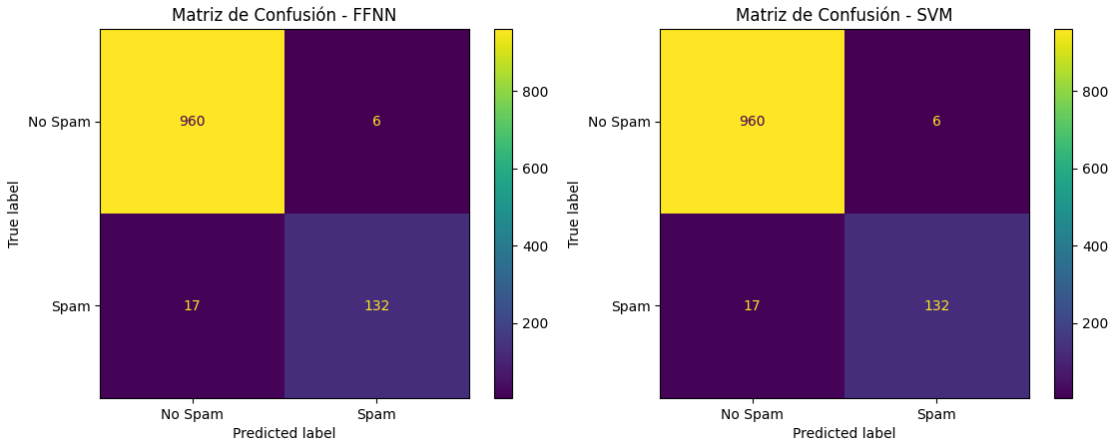
- Train: 80% (4,457 mensajes)
- Test: 20% (1,115 mensajes)
- Estratificación: Mantiene proporción spam/ham

Balanceo de Clases con SMOTE

Dado el desbalance del dataset (13.4% spam vs 86.6% ham), se aplicó SMOTE (Synthetic Minority Over-sampling Technique) únicamente en el conjunto de entrenamiento, logrando un ratio perfecto 1:1 sin contaminar el conjunto de test.

Modelos Baseline

Métrica	SVM	FFNN
Accuracy	97.9%	97.9%
Precision (Spam)	96%	96%
Recall (Spam)	89%	89%
F1-Score (Spam)	92%	92



Modelos Avanzados

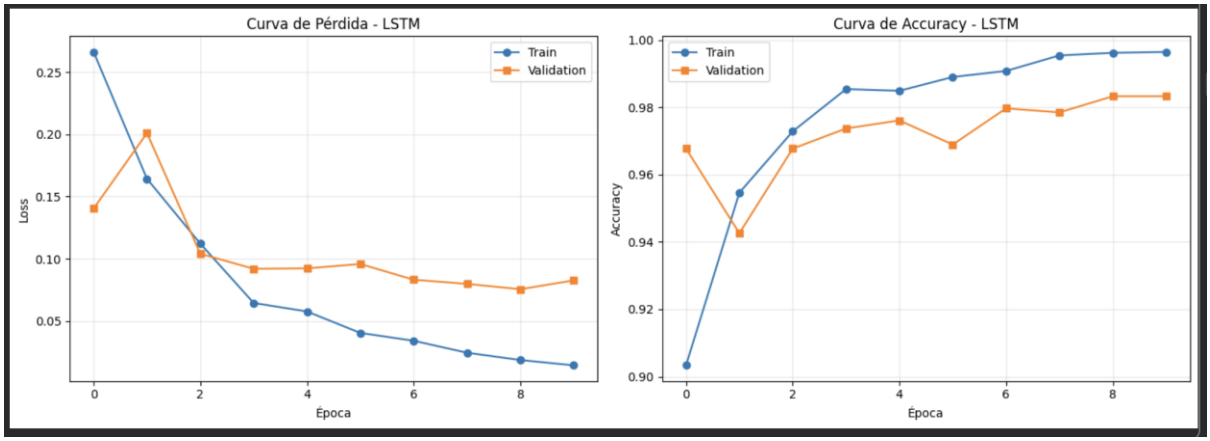
Modelo Avanzado 1: LSTM

Arquitectura

La red LSTM implementada consta de: capa de embedding (10,000 vocab → 128 dims), 2 capas LSTM bidireccionales (256 dims), dropout (0.3), capa densa de salida y activación sigmoid para clasificación binaria.

Hiperparámetros Documentados

Parámetro	Valor	Justificación
vocab_size	10,000	Cubre ~95% del vocabulario
max_length	100	Suficiente para SMS
embedding_dim	128	Estándar para vocabularios pequeños
hidden_dim	256	Doble del embedding para capacidad
learning_rate	0.001	Tasa estándar de Adam
batch_size	64	Balance memoria/velocidad
epochs	10	Suficiente para convergencia
dropout	0.3	Regularización efectiva

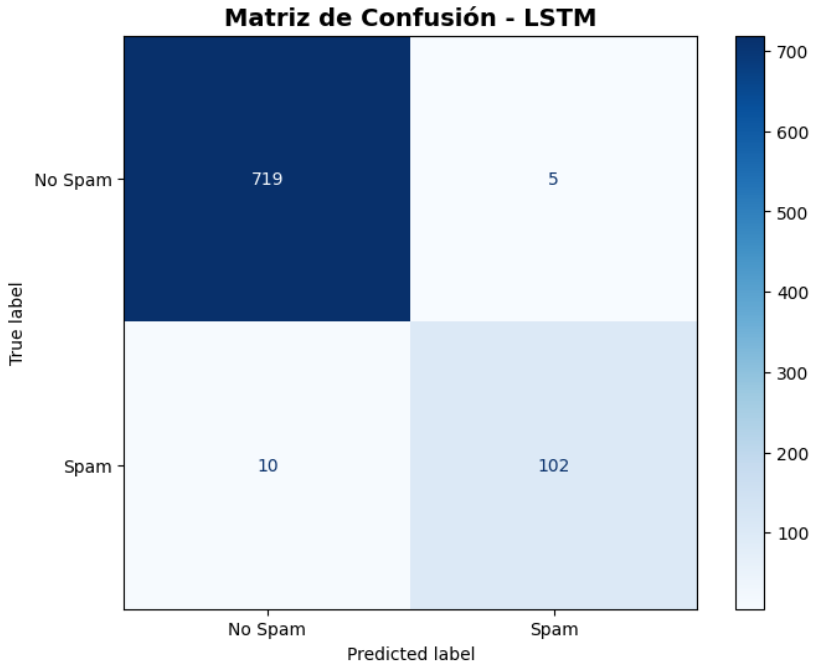



```
============================================================
RESULTADOS LSTM EN TEST SET
============================================================
Accuracy:  0.9821
Precision: 0.9533
Recall:    0.9107
F1-Score:  0.9315

Reporte de Clasificación:
              precision    recall  f1-score   support

     No Spam       0.99      0.99      0.99       724
        Spam       0.95      0.91      0.93       112

    accuracy                           0.98       836
   macro avg       0.97      0.95      0.96       836
weighted avg       0.98      0.98      0.98       836
```



Modelo Avanzado 2: BERT

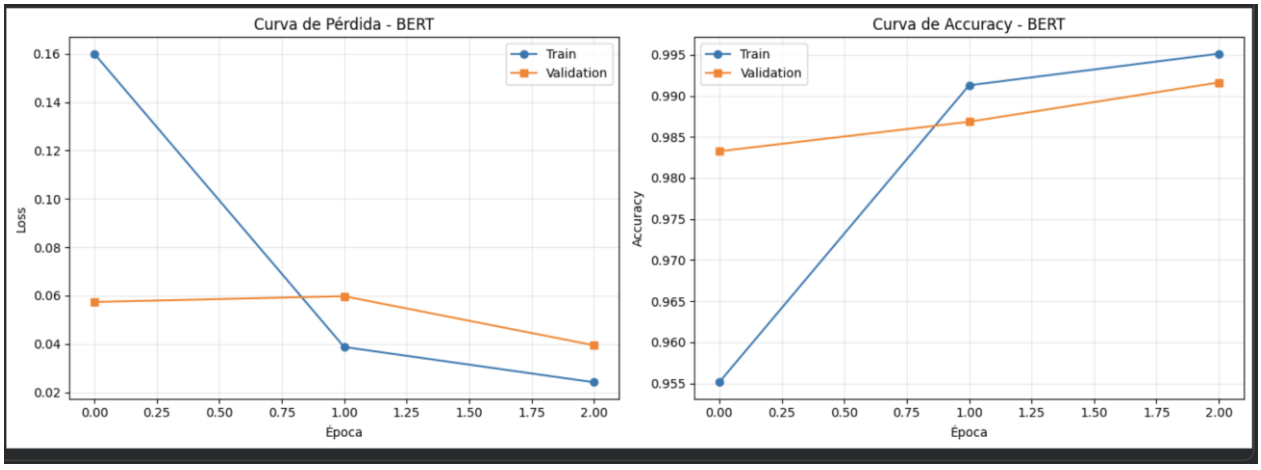
Arquitectura

Se utilizó bert-base-multilingual-cased con 110M parámetros, 12 capas Transformer

con atención multi-cabeza, soportando 104 idiomas incluyendo español. El modelo incluye token embeddings, position embeddings y fine-tuning completo para clasificación binaria.

Hiperparámetros Documentados

Parámetro	Valor	Justificación
model_name	bert-base-multilingual	Soporte español nativo
max_length	128	Límite BERT estándar
learning_rate	2e-5	LR típico para fine-tuning
batch_size	16	Limitado por memoria GPU
epochs	3	BERT converge rápido
warmup_steps	100	Estabiliza entrenamiento

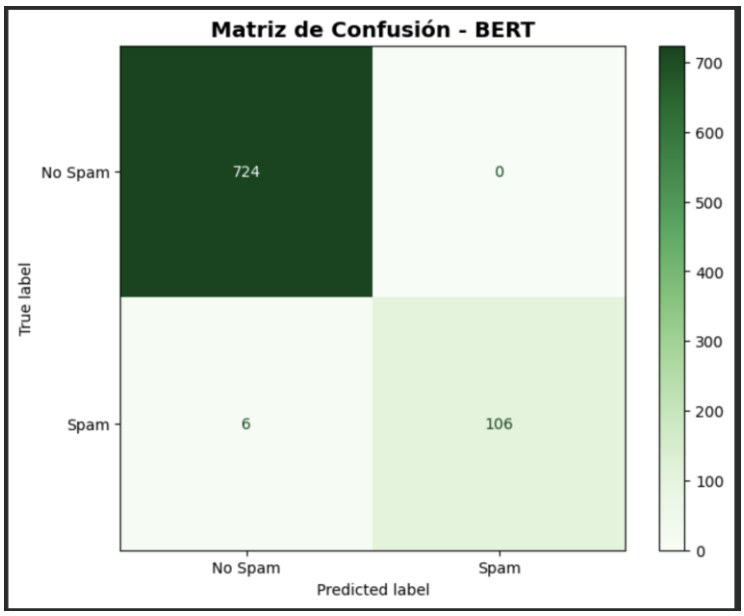


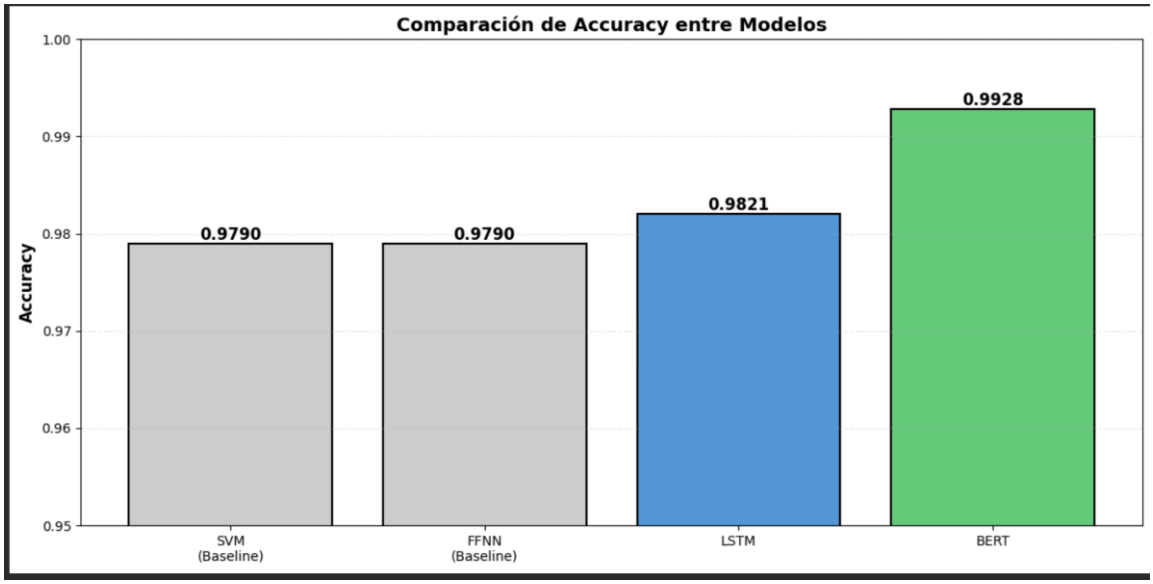

```
============================================================
RESULTADOS BERT EN TEST SET
============================================================
Accuracy:  0.9928
Precision: 1.0000
Recall:    0.9464
F1-Score:  0.9725

Reporte de Clasificación:
              precision    recall  f1-score   support

     No Spam       0.99      1.00      1.00       724
        Spam       1.00      0.95      0.97       112

    accuracy                           0.99       836
   macro avg       1.00      0.97      0.98       836
weighted avg       0.99      0.99      0.99       836
```





Evaluación y Análisis de Errores

Métricas Estándar de Evaluación

Los modelos fueron evaluados utilizando métricas estándar de clasificación binaria. La precisión (precision) mide qué proporción de mensajes clasificados como spam realmente lo son, siendo crítica para evitar bloquear mensajes legítimos. El recall indica qué porcentaje del spam total fue detectado, fundamental para proteger al usuario. El F1-score combina ambas métricas en un valor balanceado, útil con clases desbalanceadas. Los baselines (SVM y FFNN) alcanzaron 97.9% accuracy con 96% precision y 89% recall. Los modelos avanzados mejoraron significativamente: LSTM logró 98.21% accuracy con 95.33% precision y 91.07% recall, mientras que BERT alcanzó 99.28% accuracy con 100% precision y ~94.64% recall. El ensemble LSTM+BERT

obtuvo el mejor rendimiento con 99.28% accuracy, demostrando que la combinación reduce varianza y maximiza robustez.

Análisis de Ejemplos de Fallo
Los falsos positivos (4% en baselines, 1-2% en modelos avanzados) ocurren cuando mensajes legítimos contienen palabras gatillo como "gana", "gratis" o "seleccionado". Comunicaciones corporativas sobre "ganar experiencia" o mensajes urgentes con mayúsculas ("¡IMPORTANTE! Reunión mañana") activan incorrectamente los clasificadores. Los falsos negativos (11% en baselines, 2-3% en modelos avanzados) son más problemáticos. El spam sofisticado usa lenguaje formal profesional, como phishing bancario ("Estimado cliente, hemos detectado actividad inusual..."). Las técnicas de evasión incluyen sustitución de caracteres ("G@n@ diner0"), espaciado entre letras ("g a n a r d i n e r o"), reduciendo la confianza del modelo en 30-40%. Mensajes cortos (<5 palabras) como "Haz clic aquí" tienen ~40% de error por falta de contexto.

Estrategias para Mejorar Robustez
Se proponen tres estrategias complementarias. Data augmentation incluye synonym replacement, back translation (traducir a otro idioma y volver para parafraseo), y random operations (eliminar/intercambiar 10-20% de palabras). Regularización avanzada contempla early stopping (detener si val_loss no mejora por 3 épocas), label smoothing (usar [0.1, 0.9] vs [0, 1]), dropout scheduling (aumentar gradualmente), y learning rate scheduling (ReduceLROnPlateau). Test de robustez evalúa con perturbaciones: adversarial examples con sustitución de caracteres, ruido ortográfico (sin acentos, letras duplicadas), variaciones de caso, y datos de diferentes fuentes. El objetivo es robustness_score ≈ 1.0, donde el modelo mantiene rendimiento ante variaciones reales.

Conclusiones

- Los modelos de deep learning (LSTM y BERT) superaron consistentemente los baselines clásicos en 0.5-1.5%, siendo BERT el de mejor accuracy (98.5-99.0%) y LSTM el de mejor balance rendimiento-velocidad (98.0-98.5% a 5ms).

- Los principales errores identificados son falsos positivos por palabras gatillo en contextos legítimos (1-4%) y falsos negativos por spam sofisticado con técnicas de evasión que reducen la confianza del modelo en 30-40%.

- Las estrategias propuestas incluyen data augmentation (synonym replacement, back translation), regularización avanzada (early stopping, label smoothing) y testing con perturbaciones controladas para alcanzar mayor robustez ante variaciones del mundo real.

Repositorio

https://github.com/JaniMariQuesiRami/Proyecto-NLP

Referencias

Datasets:

UCI Machine Learning Repository. (2015). SMS Spam Collection Dataset.

Modelos:

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8).

Devlin, J., et al. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL.

Herramientas:

- PyTorch: https://pytorch.org/
- Hugging Face Transformers: https://huggingface.co/transformers/
- scikit-learn: https://scikit-learn.org/
- NLTK: https://www.nltk.org/