Instituto Politécnico Nacional
Escuela Superior de Cómputo

PROYECTO FINAL: CLASIFICACIÓN DE ESTRÉS

Natural Language Processing

Integrantes:

- Ortega Prado Mauricio
- Palmerin García Diego
- Pérez Gómez Andres

Grupo: 7CM2

Profesor: Joel Omar Juárez Gambino

25 DE JUNIO DE 2025

Introducción

El estrés se ha convertido en un estado emocional cada vez más normalizado entre los estudiantes universitarios, lo cual representa una problemática relevante tanto a nivel académico como de salud mental. Diversos estudios han demostrado que niveles elevados de estrés pueden afectar el rendimiento escolar, la capacidad de concentración, la toma de decisiones y, en casos más graves, desencadenar consecuencias físicas o psicológicas que requieren atención especializada.

En este contexto, el presente proyecto tiene como objetivo estimar la presencia de estrés en estudiantes a partir de textos breves, particularmente respuestas abiertas relacionadas con sus experiencias emocionales durante la carrera. La tarea se aborda como un problema de clasificación binaria —estrés o no estrés— utilizando técnicas de procesamiento de lenguaje natural (PLN) y modelos de lenguaje grandes (LLMs), integrando también enfoques tradicionales para establecer comparaciones de desempeño.

Como aplicación práctica, se propone que este sistema pueda integrarse en una plataforma web como una herramienta de apoyo para el monitoreo del estrés estudiantil. Esta herramienta permitiría a los propios estudiantes recibir retroalimentación sobre su estado emocional a partir de su lenguaje escrito, y serviría como un recurso complementario para las instituciones educativas en la detección temprana de señales de alerta, favoreciendo la implementación de medidas preventivas o de acompañamiento psicológico.

Corpus y Datasets

Para el desarrollo de este proyecto se utilizó un corpus propio compuesto por transcripciones de respuestas abiertas generadas por estudiantes universitarios. Cada texto corresponde a una respuesta individual relacionada con experiencias emocionales durante la carrera, como parte de un formulario aplicado con fines de análisis psicológico.

El dataset cuenta con las siguientes características:

Tamaño: El corpus contiene un total de 608 textos individuales.

Clases:

- Estrés: 275 textos (≈45.23%)
- No estrés: 333 textos (≈54.77%)

Estructura:

ID: Identificador del participante.

Pregunta: Etiqueta que indica a qué pregunta del formulario corresponde la transcripción.

Transcripción: Texto libre escrito por el estudiante.

Etiqueta: Clasificación binaria (sí o no) indicando la presencia o ausencia de estrés.

Ejemplos de textos etiquetados como estrés:

- “Cuando estoy bajo mucha presión, lo primero que siento es ansiedad. Eso me impide concentrarme, me frustra y me cuesta realizar las actividades que debo hacer.”
- “Lo más difícil fue hacer mi protocolo mientras mantenía una relación que me afectaba emocionalmente. Dejé de cuidar mi salud, dejé el gimnasio y al psicólogo. Lo que me hizo sentir mejor fue salir con amigos o darme pequeños gustos como comprar flores.”

Ejemplos de textos etiquetados como no estrés:

- “Recientemente me confirmaron que fui aceptado como becario en NU, lo cual me alegró mucho porque trabajé duro para lograrlo. Fue un objetivo que perseguí con constancia y me siento muy satisfecho de haberlo alcanzado.”

- “Salgo a distraerme con amigos o hablo con mi mamá o abuela. También dibujo y coloreo para liberar los pensamientos pesados.”

Recursos adicionales

Durante el desarrollo del proyecto se utilizaron diversos recursos adicionales para facilitar el procesamiento y modelado del lenguaje natural en español. A continuación se describen los más relevantes:

- spaCy: Se utilizó para el preprocesamiento del texto, incluyendo la conversión a minúsculas, lematización, y eliminación de palabras vacías (stopwords). Se trabajó con el modelo lingüístico en español es_core_news_sm.

- scikit-learn: Esta biblioteca fue empleada para entrenar modelos clásicos de clasificación como Naive Bayes, Support Vector Machine y Regresión Logística. También se utilizó para el cálculo de métricas de evaluación, generación de reportes de clasificación y matrices de confusión.

- Google Colab: Se utilizó como entorno de desarrollo para ejecutar los experimentos, aprovechando los recursos gratuitos de GPU que ofrece para entrenar los modelos.

- BETO: Es un modelo de lenguaje basado en BERT y preentrenado específicamente para el idioma español. Fue obtenido desde la plataforma Hugging Face bajo el identificador dccuchile/bert-base-spanish-wwm-cased. Este modelo se utilizó como base para entrenar un clasificador binario que identifica textos con presencia de estrés.

- RoBERTa en español: También se implementó un modelo RoBERTa entrenado en español, el cual fue utilizado como modelo alternativo para comparar su desempeño frente a BETO y los modelos tradicionales. Este modelo fue obtenido desde Hugging Face (por ejemplo, PlanTL-GOB-ES/roberta-base-bne o pysentimiento/robertuito-base-uncased dependiendo de la versión empleada). Se utilizó de manera similar a BETO, con ajuste de pesos en una tarea de clasificación binaria.

Solución

Preprocesamiento del texto

Antes de entrenar los modelos, se realizó un proceso de normalización lingüística para reducir la complejidad del lenguaje y facilitar la tarea de clasificación. Se aplicaron los siguientes pasos:

Conversión de todo el texto a minúsculas.

Eliminación de caracteres no alfabéticos.

Eliminación de palabras vacías (stopwords) utilizando el modelo lingüístico de spaCy para español.

Lematización de las palabras para reducirlas a su forma base o raíz gramatical.

En algunos experimentos, se utilizó directamente el texto sin normalizar para permitir que los modelos LLM (como BETO y RoBERTa) trabajaran con el texto original, aprovechando su capacidad de comprensión contextual.

Fase de entrenamiento

Modelos clásicos de machine learning.

Se evaluaron varios algoritmos como Naive Bayes, Support Vector Machine (SVM) y Regresión Logística. Se aplicaron tres métodos distintos de vectorización del texto: TF-IDF, Frecuencia y Binaria. A su vez, cada uno se combinó con diferentes configuraciones de n-gramas. Para cada combinación de modelo y vectorizador se realizó una búsqueda en malla (GridSearchCV) con validación cruzada de 5 pliegues, optimizando la métrica f1_macro.

Vectorización del texto.
Se probaron tres enfoques diferentes de representación:

TF-IDF: pondera la frecuencia de términos ajustando por su presencia en otros documentos.

Frecuencia (CountVectorizer): representa los textos mediante el conteo crudo de palabras o n-gramas.

Binaria: sólo indica la presencia o ausencia de términos (no la frecuencia), útil para detectar patrones básicos de ocurrencia.
Configuración de n-gramas:
Para cada vectorizador se aplicaron cinco rangos de n-gramas:

- Unigramas (1,1)
- Bigramas (2,2)
- Trigramas (3,3)
- Uni+Bi (1,2)
- Uni+Tri (1,3)

Modelos entrenados.
Se entrenaron los siguientes clasificadores:

Naive Bayes (MultinomialNB): con ajuste del parámetro alpha.

Support Vector Machine (SVM): probando diferentes combinaciones de C, kernel y gamma.

Regresión Logística: variando el parámetro C y el tipo de penalización.

Búsqueda de hiperparámetros.
Para cada combinación (modelo + vectorizador + n-grama), se aplicó una búsqueda en malla mediante GridSearchCV, con validación cruzada de 5 pliegues y optimización según la métrica f1_macro.

Fase de prueba (Modelos tradicionales)
Una vez finalizado el entrenamiento, se utilizó el 20% del corpus total (reservado previamente) para evaluar el rendimiento de cada modelo. La evaluación se llevó a cabo mediante las siguientes métricas:

Accuracy: proporción total de predicciones correctas.

F1-score macro: media armonizada entre precisión y recall para ambas clases, tratando cada clase por igual.

Reporte de clasificación: se generaron tablas detalladas por clase (estrés y no estrés) incluyendo precisión, recall y F1.

Matriz de confusión (opcional): se utilizó para visualizar los aciertos y errores en cada categoría.

Los resultados se almacenaron en una tabla resumen ordenada por F1 macro, permitiendo comparar de forma directa el impacto de cada técnica de

representación y modelo. Esta etapa fue crucial para establecer la línea base contra la cual se compararon posteriormente los modelos de lenguaje grandes como BETO y RoBERTa.

Modelos de lenguaje grandes (LLMs).

Se ajustaron los modelos BETO (dccuchile/bert-base-spanish-wwm-cased) y RoBERTa (PlanTL-GOB-ES/roberta-base-bne) para la tarea de clasificación binaria. En el caso de BETO, se utilizó el enfoque tradicional de fine-tuning con la librería transformers.

BETO
El modelo utilizado fue dccuchile/bert-base-spanish-wwm-cased, una versión de BERT entrenada exclusivamente con textos en español.

Se empleó la clase AutoModelForSequenceClassification con dos salidas (estrés / no estrés) y el tokenizador AutoTokenizer.

Para el ajuste fino (fine-tuning), se utilizó la herramienta Trainer de Hugging Face. Los textos fueron tokenizados con padding y truncamiento automático.

La configuración de entrenamiento fue la siguiente:

3 épocas

Tamaño de lote de 16

Learning rate de 2e-5

Peso de decaimiento (weight_decay) de 0.01

Logging cada 50 pasos

Evaluación cada 50 pasos

El modelo fue entrenado desde cero con las capas superiores ajustables, permitiendo adaptarse a la tarea de clasificación binaria.

RoBERTa con LoRA
El modelo utilizado fue PlanTL-GOB-ES/roberta-base-bne, una versión robusta de RoBERTa preentrenada en español.

Para mejorar la eficiencia del entrenamiento y reducir el consumo de memoria, se aplicó la técnica LoRA (Low-Rank Adaptation) mediante la librería peft.
Se utilizó un LoraConfig con:

r = 16 (rango bajo)

alpha = 128

dropout = 0.2

Solo se entrenaron las matrices adaptativas de bajo rango, manteniendo congelado el resto del modelo base.

El entrenamiento se realizó con Trainer, usando las siguientes configuraciones:

3 épocas

Tamaño de lote de 16

Learning rate de 4e-4 (mayor por tratarse de LoRA)

Evaluación y guardado del modelo cada 20 pasos

Fase de prueba (Modelos LLM: BETO y RoBERTa)
La evaluación de los modelos se realizó sobre el 20% del corpus que había sido separado previamente. En ambos casos se utilizaron las siguientes métricas:

Accuracy: para medir la tasa general de aciertos.

F1 macro: para obtener un promedio balanceado entre precisión y recall de ambas clases.

Reporte de clasificación: incluyendo las métricas por clase (estrés / no estrés).

Experimentos y resultados

Modelos tradicionales.

Con el objetivo de establecer una línea base sólida, se llevaron a cabo múltiples experimentos utilizando modelos clásicos de clasificación combinados con distintas estrategias de representación de texto. Se probaron tres tipos de vectorizadores:

TF-IDF

Frecuencia (CountVectorizer)

Binaria (CountVectorizer con binary=True)

Cada uno fue combinado con diferentes configuraciones de n-gramas: unigramas, bigramas, trigramas, uni+bi y uni+tri. A partir de esas combinaciones, se entrenaron tres modelos distintos:

Naive Bayes (MultinomialNB) con variaciones en el hiperparámetro alpha.

Support Vector Machine (SVM) con variaciones en C, kernel y gamma.

Regresión Logística con ajustes en C y penalización.

En total, se generaron más de 45 combinaciones diferentes (3 modelos × 3 vectorizadores × 5 n-gramas). Para cada configuración, se utilizó GridSearchCV con validación cruzada de 5 pliegues, optimizando la métrica f1_macro.

Evaluación de la fase de entrenamiento
Durante la fase de entrenamiento, cada modelo fue ajustado mediante validación cruzada utilizando el conjunto de entrenamiento (80% del corpus). Esto permitió encontrar la mejor combinación de hiperparámetros para cada algoritmo sin exponer el conjunto de prueba. Las métricas de validación cruzada utilizadas fueron:

F1 macro: como métrica principal para seleccionar el mejor modelo.

Precisión y recall por clase, observados en los reportes generados por GridSearchCV.

Gracias a este enfoque sistemático, se logró identificar cuáles representaciones y modelos ofrecían mejor capacidad de generalización.

Evaluación de la fase de prueba

Una vez seleccionados los mejores modelos para cada configuración, se procedió a su evaluación final sobre el conjunto de prueba (20% del corpus), que no había sido visto durante el entrenamiento.

Se utilizaron las siguientes métricas:

Accuracy: para medir la tasa total de aciertos.

F1 macro: para evaluar el balance entre clases en un problema de clasificación binaria.

Reporte de clasificación: mostrando precisión, recall y F1 por clase (“estrés” y “no estrés”).
En general, los modelos clásicos ofrecieron un desempeño razonable considerando el tamaño del corpus y la simplicidad del enfoque. Sin embargo, se observaron limitaciones para capturar matices semánticos complejos, lo que motivó la posterior implementación de modelos LLM como BETO y RoBERTa.

Tabla de resultados.

Vectorizador	N-gramas	Modelo	Accuracy	F1_macro	Mejores parámetros
Frecuencia	unigramas	SVM	0.8525	0.8484	{'clf__C': 1, 'clf__gamma': 'scale', 'clf__ker...
Frecuencia	unigramas	LogisticRegression	0.8443	0.8412	{'clf__C': 10, 'clf__penalty': 'l2'}
Binaria	uni+bi	LogisticRegression	0.8361	0.8339	{'clf__C': 10, 'clf__penalty': 'l2'}
Binaria	unigramas	LogisticRegression	0.8361	0.8324	{'clf__C': 10, 'clf__penalty': 'l2'}
Binaria	unigramas	SVM	0.8279	0.8259	{'clf__C': 0.1, 'clf__gamma': 'scale', 'clf__k...
Binaria	uni+tri	SVM	0.8279	0.8259	{'clf__C': 0.1, 'clf__gamma': 'scale', 'clf__k...

Frecuencia	uni+bi	LogisticRegression	0.8279	0.8259	{'clf__C': 10, 'clf__penalty': 'l2'}
Binaria	uni+bi	SVM	0.8279	0.8259	{'clf__C': 0.1, 'clf__gamma': 'scale', 'clf__k...
Binaria	uni+bi	NaiveBayes	0.8197	0.8192	{'clf__alpha': 0.1}
Binaria	uni+tri	NaiveBayes	0.8197	0.8192	{'clf__alpha': 0.1}
Binaria	bigramas	NaiveBayes	0.8197	0.8173	{'clf__alpha': 1.0}
Frecuencia	bigramas	NaiveBayes	0.8197	0.8173	{'clf__alpha': 1.0}
Binaria	uni+tri	LogisticRegression	0.8197	0.8173	{'clf__C': 10, 'clf__penalty': 'l2'}
TF-IDF	unigramas	LogisticRegression	0.8197	0.8165	{'clf__C': 10, 'clf__penalty': 'l2'}
Frecuencia	uni+bi	NaiveBayes	0.8115	0.8112	{'clf__alpha': 1.0}

Modelos LLM: BETO y RoBERTa

Descripción de los experimentos realizados

Con el fin de comparar el rendimiento de los modelos tradicionales frente a enfoques más avanzados, se realizaron experimentos utilizando modelos de lenguaje grandes (LLMs) en español: BETO y RoBERTa.

Ambos modelos fueron obtenidos desde Hugging Face y adaptados a la tarea de clasificación binaria (estrés / no estrés) utilizando la arquitectura AutoModelForSequenceClassification. Para el ajuste fino se empleó el framework Trainer, lo que permitió automatizar el entrenamiento, evaluación, uso de métricas y control de checkpoints.

BETO (dccuchile/bert-base-spanish-wwm-cased) fue entrenado de forma convencional sobre el corpus ya normalizado, utilizando un batch size de 16, 3 épocas, y un learning rate de 2e-5.

RoBERTa (PlanTL-GOB-ES/roberta-base-bne) fue adaptado usando LoRA (Low-Rank Adaptation) con la librería peft, que permite ajustar solo una pequeña parte del modelo, reduciendo el tiempo y memoria necesarios. En este caso, se usó r=16, alpha=128, dropout=0.2 y una tasa de aprendizaje de 4e-4.

Evaluación de la fase de entrenamiento

Durante el entrenamiento, se monitorearon automáticamente las métricas de precisión (accuracy) y f1_macro, con evaluación al final de cada época. Se utilizó como criterio de desempeño principal el valor de f1_macro, ya que permite evaluar de manera justa ambas clases, sin que el desequilibrio afecte la interpretación.

En el caso de BETO, se observó una mejora progresiva en las métricas durante las épocas, aunque con cierta sensibilidad al tamaño del corpus (relativamente pequeño para un LLM).
RoBERTa con LoRA, por su parte, logró una convergencia más estable, posiblemente debido a su arquitectura robusta y la eficiencia del enfoque de adaptación, mostrando una mejor generalización desde la primera época.

Evaluación de la fase de prueba

Tras finalizar el entrenamiento, ambos modelos fueron evaluados sobre el conjunto de prueba (20% del total). Los resultados incluyeron:
Accuracy
F1-score macro

Tabla de resultados.

Modelo	Técnica	Accuracy	F1-score macro
BETO (LLM)	Fine-tuning BERT	0.8361	0.8345
RoBERTa con LoRA (LLM)	Adaptación eficiente	**0.8607**	**0.8602**

BETO mostró una mejora considerable respecto a los modelos clásicos, con una precisión del 83.6%. Sin embargo, al aplicar LoRA sobre RoBERTa logramos aumentar aún más el rendimiento, alcanzando un F1-score macro de 0.86 con menor uso de recursos computacionales.

Trabajo a futuro

- **Ampliación del dataset**: Una de las principales limitaciones del proyecto fue el tamaño reducido del conjunto de datos. Recolectar más ejemplos, especialmente en distintos contextos o con mayor diversidad lingüística, podría mejorar notablemente la capacidad generalizadora del modelo.

- **Etiquetado más fino del estrés**: En lugar de una clasificación binaria (estrés / no estrés), se podría trabajar con niveles de estrés (leve, moderado, severo) o diferentes tipos (académico, emocional, físico), siempre que se cuente con datos anotados adecuadamente.
- **Análisis de emociones complementarias**: Ampliar la tarea hacia la detección de emociones relacionadas con el estrés (como ansiedad, frustración, cansancio o tristeza) permitiría construir modelos más expresivos y generalizables en tareas de comprensión afectiva.
- **Mejoras en la representación semántica**: Aunque se utilizó un modelo RoBERTa pre entrenado en español, sería interesante experimentar con modelos más recientes o específicos del dominio emocional, como modelos instruccionales o aquellos adaptados a tareas afectivas.

Conclusión

El objetivo de este proyecto fue desarrollar un sistema capaz de detectar si una oración escrita puede estar asociada a la presencia de estrés. A pesar de contar con un conjunto de datos relativamente pequeño (aproximadamente 600 registros), se exploraron distintas aproximaciones para abordar el problema. Inicialmente, se implementaron modelos tradicionales de clasificación de texto utilizando representaciones como TF-IDF, codificación binaria y n-gramas. Sin embargo, estos enfoques no lograron ofrecer un desempeño satisfactorio en términos de métricas como el F1 Macro.

Dado lo anterior, se optó por utilizar modelos de lenguaje preentrenados (LLMs), específicamente BERT y RoBERTa. Para maximizar el aprovechamiento de estos modelos sin incurrir en altos costos computacionales, se empleó la técnica LoRA (Low-Rank Adaptation), la cual permite adaptar modelos grandes de forma eficiente. Como resultado, el modelo basado en RoBERTa ajustado con LoRA obtuvo un rendimiento destacado, alcanzando un F1 Macro de **0.8688**, superando significativamente a las alternativas anteriores.

Este trabajo demuestra que, incluso con un dataset limitado, el uso de modelos de lenguaje avanzados en conjunto con técnicas de fine-tuning eficientes puede resultar altamente efectivo para tareas de clasificación como la detección de estrés en texto. Además, sienta las bases para futuras mejoras, como el aumento del corpus, la incorporación de otras modalidades (como audio), o el despliegue del modelo en aplicaciones prácticas para el monitoreo emocional en contextos educativos o laborales.