UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Métodos de aprendizaje automático y modelos de lenguaje masivos para la detección de estereotipos en comentarios de texto en español

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Frances Perez, Nerea

Tutor/a: Rosso, Paolo

Cotutor/a: Pelechano Ferragud, Vicente

CURSO ACADÉMICO: 2023/2024

Resum

Aquest treball de fi de grau s'enfoca en la detecció d'estereotips en textos en espanyol, una tasca vital per a mitigar el discurs d'odi a internet. Els estereotips són idees preconcebudes sobre diferents grups socials basades en la raça, gènere, religió, entre altres factors. Amb l'apogeu de les xarxes socials i la gran quantitat d'informació generada diàriament, la detecció manual d'aquests estereotips es torna insostenible. Aquest projecte utilitza models d'aprenentatge automàtic, així com models de llenguatge massiu com el conegut GPT per identificar i classificar estereotips en grans volums de dades. L'avaluació d'aquests models es realitza en el marc de la competició DETESTS-Dis, que se centra en la detecció d'estereotips en espanyol. Al llarg de l'estudi, s'utilitzaran tècniques comunes de PLN i es compararan diferents models i els seus resultats, amb l'objectiu de mostrar quin és més eficaç exercint la tasca de classificació d'estereotips i així, poder proporcionar noves possibilitats per millorar la moderació de contingut en línia.

Key words: Estereotips, immigració, processament del llenguatge natural, xarxes socials, transformers, GPT, Regressió Logística, SVM

Resumen

Este trabajo de fin de grado se enfoca en la detección de estereotipos en textos en español, una tarea vital para mitigar el discurso de odio en internet. Los estereotipos son ideas preconcebidas sobre diferentes grupos sociales basadas en su raza, género, religión, entre otros factores. Con el auge de las redes sociales y la gran cantidad de información generada a diario, la detección manual de estos estereotipos se vuelve insostenible. Este proyecto utiliza modelos de aprendizaje automático, así como modelos de lenguaje masivo como el conocido GPT para identificar y clasificar estereotipos en grandes volúmenes de datos. La evaluación de estos modelos se realiza en el marco de la competición DETESTS-Dis, que se centra en la detección de estereotipos en español. A lo largo del estudio, se utilizarán técnicas comunes de PLN y se compararán diferentes modelos y sus resultados con el objetivo de mostrar cuál de ellos es más eficaz ejerciendo la tarea de clasificación de estereotipos y así, poder proporcionar nuevas posibilidades para mejorar la moderación de contenido en línea.

Key words: Estereotipos, inmigración, procesamiento del lenguaje natural, redes sociales, transformers, GPT, Regresión Logística, SVM

Abstract

This final degree project focuses on the stereotypes detection in Spanish texts, a vital task to mitigate hate speech on the internet. Stereotypes are preconceived ideas about different social groups based on their race, gender, religion, among other factors. With the rise of social media and the large amount of information daily generated, manual detection of these stereotypes becomes unsustainable. This project uses machine learning models, as well as large language models such as the well-known GPT to identify and classify stereotypes in large volumes of data. The evaluation of these models is carried out within the framework of the DETESTS-Dis competition, which focuses on the detection of stereotypes in Spanish. Throughout the study, common NLP techniques will

be used and different models and their results will be compared with the aim of showing which of them is most effective in performing the task of stereotype classification and thus, being able to provide new possibilities to improve the moderation of online content.

Key words: Stereotypes, immigration, natural language processing, social networks, transformers, GPT, Logistic Regression, SVM

Índice general

Índice de figuras

Índice de cuadros

CAPÍTULO 1
Introducción

La expansión del entorno digital ha dado lugar a un incremento significativo en el uso de plataformas donde los individuos pueden compartir sus opiniones y sentimientos sobre una amplia variedad de temas. Estas plataformas, como Twitter (ahora conocida como X), y los comentarios en artículos de noticias proporcionan una gran cantidad de datos que pueden ser utilizados para entender mejor la percepción pública y los sentimientos generales. No obstante, convertir estos datos en información útil y comprensible sigue siendo un desafío importante.

Los estereotipos, que son creencias generalizadas y simplificadas sobre ciertos grupos de personas, juegan un papel crucial en la propagación de discursos tóxicos y de odio. Comprender cómo surgen y se difunden es esencial para abordar este problema, ya que no siempre se expresan de manera explícita. Su presencia en las redes sociales y comentarios en plataformas online, y la necesidad de identificarlos y mitigarlos ha impulsado el desarrollo de sistemas para su detección automática.

La identificación de estereotipos en comentarios online presenta múltiples desafíos debido al volumen masivo de datos y la variabilidad en la manera en que se expresan. Estos pueden estar implícitos en el lenguaje, lo que dificulta su detección mediante métodos convencionales. Además, los comentarios a menudo contienen lenguaje informal, jerga, abreviaturas y errores tipográficos, añadiendo una capa adicional de complejidad en el análisis automático ya que la ambigüedad del lenguaje y la necesidad de comprender el contexto son aspectos necesarios para la tarea.

En este trabajo, se introduce la tarea DETESTS-Dis (DETEction and classification of racial STereotypes in Spanish) [1] de IberLEF 2024 [1], cuyo objetivo es detectar estereotipos en textos en español en redes sociales y comentarios en artículos de noticias. Los textos analizados consisten en publicaciones completas de la red social X en respuesta a falsos rumores verificados (un tipo de noticias falsas) y oraciones extraídas de comentarios en artículos de noticias relacionadas con la inmigración. Este estudio no busca solamente detectar estereotipos explícitos e implícitos, sino también proporcionar herramientas efectivas para combatir el discurso de odio en el entorno digital.

Además, a lo largo del proyecto se analizarán y compararán modelos de aprendizaje automático tradicionales con modelos de lenguaje masivo (LLM, por su acrónimo en inglés) [2] más recientes y en tendencia, para determinar cuáles son más efectivos en la detección de estereotipos.

[1] https://detests-dis.github.io/

1.1 Motivación

El incremento de estereotipos en redes sociales y plataformas online puede generar graves consecuencias sociales, desde la perpetuación de prejuicios hasta la incitación a la violencia. Ejemplos recientes muestran cómo estos comentarios refuerzan estereotipos negativos, como comentarios racistas relacionados con la inmigración [2].

Esta problemática subraya la necesidad urgente de sistemas automáticos que detecten y mitiguen los estereotipos en línea. Mi interés en el procesamiento del lenguaje natural (PLN) [3] me motiva a contribuir en esta tarea. En este proyecto, se analizarán y compararán modelos de aprendizaje automático tradicionales y avanzados, como los modelos de lenguaje masivo (LLM), para determinar su eficacia en la detección de estereotipos.

La tarea DETESTS-Dis, que utiliza un corpus de publicaciones en X reaccionando a noticias falsas sobre inmigración (StereoHoax-ES) y comentarios en artículos de noticias (DETESTS), será fundamental en este estudio. Este trabajo tiene como objetivo no solo la detección de estereotipos explícitos e implícitos en los textos analizados, sino también el desarrollo de estrategias efectivas para mitigar el discurso de odio en plataformas digitales.

1.2 Objetivos

El presente trabajo tiene como objetivos principales, los siguientes:

- **Evaluar modelos tradicionales**: Analizar el rendimiento de modelos de aprendizaje automático (ML [4], por su acrónimo en inglés) como SVM [5] y Regresión Logística [6] en la detección de estereotipos.

- **Evaluar modelos avanzados**: Examinar la eficacia de modelos de lenguaje masivo como BETO [3], RoBERTa [4] y GPT-3.5 Turbo [5] en la misma tarea.

- **Comparación de rendimiento**: Comparar los resultados obtenidos por los modelos tradicionales y los modelos de lenguaje masivo, considerando métricas como precisión[7], recall [7] y puntuación F1 [8].

- **Análisis de resultados**: Determinar cuál de los enfoques, tradicional o de lenguaje masivo, es más efectivo para la identificación de estereotipos, así como cuál de todos los modelos es mejor en esta tarea.

[2] https://theobjective.com/espana/2024-02-04/discursos-odio-gobierno-inmigracion-seguridad/. Visitado el 23/05/2024.

[3] https://github.com/dccuchile/beto

[4] https://huggingface.co/docs/transformers/model_doc/roberta

[5] https://platform.openai.com/docs/models/gpt-3-5-turbo

1.3 Estructura

La presente memoria se encuentra organizada en 7 capítulos, junto a una bibliografía y un anexo.

1. **Introducción.** Consiste en el capítulo actual. En este capítulo, se define la relación del trabajo con los estereotipos así como la motivación y los objetivos de este.

2. **Estado del arte para la detección de estereotipos.** Esta parte describe los conceptos básicos entorno al PLN y el *hate speech* además de explicar el contexto tecnológico y resultados de la tarea DETESTS-Dis de la edición de IberLEF 2022.

3. **Análisis del problema.** En este apartado, se analiza la importancia de la detección de estereotipos, se exploran las características del dataset utilizado y se abordan los retos y desafíos específicos de esta tarea en español.

4. **Diseño.** Este capítulo detalla el diseño experimental del estudio, la elección de modelos y técnicas utilizadas así como las métricas de evaluación empleadas para comparar los modelos y se presentará el esquema general de la metodología utilizada para llevar a cabo los experimentos y análisis.

5. **Desarrollo.** Se comenta el proceso desde la propuesta inicial hasta la solución final junto con los problemas y dificultades encontrados, y las decisiones tomadas para resolverlos. Se presentan también las tecnologías y lenguajes utilizados durante el desarrollo del proyecto.

6. **Marco experimental.** Durante este apartado se describe la experimentación llevada a cabo siguiendo la tarea DETESTS-Dis y se muestran y comparan los resultados obtenidos.

7. **Conclusiones.** Recapitulación del trabajo y los resultados, relación con los objetivos de la memoria y propuesta de ideas y mejoras.

8. **Bibliografía.** Esta sección recoge todas las fuentes bibliográficas utilizadas a lo largo del trabajo, proporcionando referencias detalladas de artículos científicos, libros y otros recursos que respaldan la investigación y los análisis realizados en el estudio.

9. **Anexo ODS.** Se vincula el trabajo con los Objetivos de Desarrollo Sostenible (ODS).

CAPÍTULO 2
Estado del arte

El campo del Procesamiento de Lenguaje Natural ha experimentado avances notables en la última década, impulsados por el desarrollo de modelos de lenguaje masivos. Estos modelos han transformado nuestra capacidad para comprender y generar texto, superando significativamente a los métodos tradicionales en una amplia gama de tareas. Desde la traducción automática y el resumen de textos hasta la clasificación de sentimientos y la detección de discurso de odio, los modelos basados en transformadores han demostrado ser herramientas extremadamente poderosas y versátiles.

En este capítulo, se revisan los modelos de lenguaje masivos más influyentes y su impacto en el PLN, con un enfoque particular en la detección de estereotipos y lenguaje ofensivo. Además, se discutirán estudios relevantes y resultados obtenidos utilizando estos modelos, destacando las técnicas y enfoques que han demostrado ser más efectivos en la mitigación de prejuicios y la promoción de una comunicación más equitativa y justa en el ámbito digital.

2.1 Modelos de lenguaje masivos

Los transformadores [9] han revolucionado el campo del procesamiento de lenguaje natural, proporcionando avances significativos en la comprensión y generación de texto. Los modelos basados en transformadores han demostrado ser extremadamente eficaces en tareas como la traducción automática, el resumen de textos, y la clasificación de sentimientos. Una de las principales innovaciones de los transformadores es su capacidad para manejar contextos largos y relaciones complejas dentro del texto, superando a los modelos basados en redes neuronales recurrentes (RNN) [10] y Long Short-Term Memory (LSTM) [11] en varias métricas de rendimiento.

Los modelos de GPT (Generative Pre-trained Transformer), como GPT-2 y GPT-3, han llevado estas capacidades un paso más allá, permitiendo la generación de texto coherente y contextualmente relevante con una precisión impresionante. GPT-3, en particular, es conocido por su capacidad de aprendizaje en pocos disparos (few-shot learning) [12], donde puede realizar nuevas tareas con solo unos pocos ejemplos, sin necesidad de un entrenamiento adicional extensivo. Esta capacidad ha permitido aplicaciones inesperadas, desde la generación automática de código hasta la detección de sesgos en los medios de comunicación [13].

En investigaciones recientes, como la de [14], se ha demostrado que los modelos de lenguaje masivo pueden utilizar técnicas de *few-shot learning* para mejorar su capacidad de clasificación en tareas como la detección de teorías conspirativas y estereotipos. Estos enfoques permiten que modelos como GPT-3.5 Turbo se adapten rápidamente a nuevas

tareas utilizando solo unos pocos ejemplos. Sin embargo, para maximizar la efectividad de estos modelos, es crucial que los ejemplos proporcionados sean precisos y altamente representativos del tipo de clasificación deseada.

La detección de discurso de odio y lenguaje ofensivo es una tarea crítica en el PLN, especialmente con el creciente uso de las redes sociales. Los transformadores han demostrado ser altamente efectivos en esta área. Por ejemplo, estudios recientes han mostrado que modelos como BERT [1] y sus variantes multilingües (mBERT) pueden detectar con precisión el discurso de odio y el lenguaje ofensivo en múltiples idiomas. Estos modelos se benefician de su capacidad para comprender el contexto y las sutilezas del lenguaje, lo que les permite diferenciar mejor entre comentarios ofensivos y discursos de odio [15].

Un estudio que utilizó BERT y mBERT para la detección de discurso de odio y lenguaje ofensivo en Twitter y Reddit encontró que estos modelos superaban a los métodos tradicionales en precisión y recall. El enfoque multitarea (MTL, por su acrónimo en inglés) utilizado en el estudio permitió que el modelo compartiera características entre tareas relacionadas, mejorando aún más su rendimiento [16]. Otro estudio evaluó la efectividad de GPT-3 en la detección de discurso de odio y encontró que, aunque GPT-3 tiene una alta capacidad para identificar contenido ofensivo, su rendimiento puede verse afectado por sesgos inherentes en los datos de entrenamiento, lo que resalta la importancia de utilizar datos balanceados y diversos para el entrenamiento [17].

Además de los estudios mencionados, los modelos de lenguaje masivo también han sido aplicados con éxito en otras áreas del PLN. Por ejemplo, en la tarea de resumen automático de texto, modelos como BART y T5 han mostrado un rendimiento superior en la generación de resúmenes concisos y coherentes de documentos largos [18]. En la tarea de respuesta a preguntas, modelos como ALBERT han demostrado ser excepcionales en benchmarks como SQuAD, proporcionando respuestas precisas a preguntas formuladas a partir de información contenida en párrafos de texto proporcionados [19]. Otra área importante es la generación de texto creativo, donde modelos como GPT-3 han sido utilizados para escribir artículos, cuentos y poesía con una calidad sorprendente [20].

Diversos estudios han apoyado la superioridad de los transformadores y los GPTs en tareas de PLN. Por ejemplo, el uso de transformadores en la tarea de DETOXIS [2] para detectar toxicidad en comentarios en español demostró que los modelos basados en BERT lograron los mejores resultados en comparación con los métodos convencionales de aprendizaje automático. Asimismo, la tarea EXIST [3] sobre identificación de sexismo en redes sociales también evidenció la eficacia de los transformadores aplicados con técnicas de fine-tuning [21].

En resumen, los transformadores y los modelos GPTs han transformado el campo del PLN, proporcionando herramientas poderosas para la detección de discurso de odio y lenguaje ofensivo. Estos modelos no solo mejoran la precisión y la eficacia de estas tareas, sino que también abren nuevas posibilidades para aplicaciones innovadoras en diversos dominios del lenguaje natural.

[1] https://huggingface.co/docs/transformers/model_doc/bert
[2] http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/6390
[3] https://www.damianospina.com/publication/plaza-2023-overview/

2.2 Identificación de estereotipos

La identificación de estereotipos en el procesamiento de lenguaje natural (PLN) es un campo crucial debido a su impacto en la perpetuación de prejuicios y sesgos en plataformas online. Los estereotipos sobre ciertos grupos de personas, pueden llevar a consecuencias negativas, como la discriminación y la marginación social. La necesidad de detectar y mitigar estos estereotipos es vital para promover la equidad y la justicia en la sociedad digital. El Observatorio Español del Racismo y la Xenofobia (OBERAXE) [4] subraya la importancia de este problema, destacando el aumento del discurso de odio y los estereotipos en las redes sociales. Un artículo reciente, publicado en colaboración con OBERAXE, presenta una taxonomía y una guía para detectar el discurso de odio contra inmigrantes en las redes sociales, evidenciando la necesidad de herramientas avanzadas para monitorear y combatir este fenómeno [22].

El problema de los estereotipos también se refleja en el discurso de los políticos, quienes a menudo utilizan estereotipos al referirse a los inmigrantes. En diversos estudios [23] [24], se propone detectar estereotipos sobre los inmigrantes enfocándose en los marcos narrativos en los que se sitúa al grupo en los discursos públicos y la política. Se realizaron experimentos empleando modelos de transformadores de última generación (monolingües y multilingües) y cuatro clasificadores de aprendizaje automático clásicos. Los resultados mostraron que el modelo BETO alcanzó una precisión superior al 83 % en ambos experimentos, demostrando que los transformadores pueden capturar estereotipos sobre inmigrantes con un alto nivel de exactitud.

La tarea DETESTS [25], presentada en el foro de evaluación IberLEF 2022, es un ejemplo destacado en la identificación y clasificación de estereotipos en textos. En esta tarea, los participantes tuvieron que identificar la presencia de estereotipos en comentarios de noticias relacionados con la inmigración en español y clasificarlos en diferentes categorías [26]. En la Subtarea 1, los participantes tuvieron que determinar si las oraciones contenían algún estereotipo o ninguno. Los resultados más destacados en esta subtarea fueron obtenidos por el equipo I2C III, que utilizó un enfoque basado en BETO y logró una F-Score de 0,70, seguido por UMUTeam, quien también empleó BETO, con una F-Score de 0,69, y Lak NLP, que utilizó modelos basados en RoBERTa, con una F-Score de 0,66. La Subtarea 2, consistía en clasificar las oraciones etiquetadas con estereotipos en una o más de diez categorías en las cuales se presentaba a los immigrantes como: víctimas de xenofobia, víctimas que sufren, recursos económicos, un problema de control migratorio, personas con diferencias culturales y religiosas, personas que se aprovechan de las ayudas sociales, un problema para la salud pública, una amenaza para la seguridad, deshumanización y otros. El equipo MALNIS obtuvo los mejores resultados, empleando una estrategia de aprendizaje multitarea que modeló la distribución conjunta de categorías. Se demostró que los modelos avanzados lograron resultados superiores en comparación con enfoques tradicionales. Este estudio evidenció que los modelos de lenguaje como BETO y RoBERTa no solo eran más precisos, sino que también podían adaptarse mejor a las variaciones del lenguaje utilizado en los comentarios online . Este hallazgo subraya la importancia de utilizar modelos de lenguaje avanzados para abordar el problema de los estereotipos en plataformas digitales.

En resumen, la tarea DETESTS y otros estudios relacionados han mostrado avances significativos en la detección y clasificación de estereotipos, destacando la importancia de

[4] https://www.inclusion.gob.es/oberaxe/es/index.htm

utilizar modelos de lenguaje avanzados para mitigar los efectos negativos en aplicaciones online y en la sociedad. Sin embargo, en la mayoría de los trabajos, no se han empleado modelos de lenguaje masivo. Es por esto que, en este trabajo se ha utilizado el modelo GPT-3.5 Turbo de manera que podamos analizar su desempeño en este tipo de tareas y compararlo con otros modelos más estudiados.

CAPÍTULO 3
Análisis del problema

Tras finalizar el estudio sobre el estado del arte, se hace evidente la necesidad de abordar el problema de la detección de estereotipos en textos en español. En este apartado, se analizarán en detalle los desafíos y las características del problema, con el objetivo de entender mejor las dificultades que plantea y cómo pueden ser superadas.

3.1 Definición del problema

La tarea DETESTS-Dis [1], que se llevará a cabo como parte de IberLEF 2024, se centra en la detección y clasificación de estereotipos en textos de redes sociales y comentarios en noticias. Esta tarea es crucial debido a la influencia negativa que los estereotipos pueden tener en la sociedad, perpetuando prejuicios y fomentando el discurso de odio. Comprender cómo surgen y se propagan estos estereotipos es esencial para desarrollar sistemas efectivos que puedan identificarlos y mitigarlos.

El objetivo principal de la tarea es detectar y clasificar estereotipos explícitos e implícitos en textos, incorporando técnicas de aprendizaje con desacuerdo para reflejar las posibles discrepancias en las anotaciones debido a la ambigüedad y la subjetividad del lenguaje. Sin embargo, en lugar del método de aprendizaje con desacuerdo, en este proyecto se han llevado a cabo experimentos utilizando etiquetas definitivas (*hard labels*) [27]. Esto permitió una evaluación más directa y uniforme de los modelos, facilitando la comparación de resultados.

La tarea se divide en dos subtareas:

Subtarea 1: Identificación de estereotipos

Esta subtarea es de clasificación binaria, cuyo objetivo es determinar si un comentario contiene al menos un estereotipo o ninguno. La clasificación se basa en la distribución completa de etiquetas proporcionadas por los anotadores. Este enfoque reconoce que no siempre hay una única etiqueta correcta debido a la naturaleza subjetiva y a veces ambigua de los estereotipos. En este trabajo, se ha optado por utilizar *hard labels* representadas por la etiqueta *stereotype*, derivadas del método de votación mayoritario de los anotadores.

[1] https://detests-dis.github.io/

Subtarea 2: Identificación de la implicitud

Esta subtarea introduce un problema adicional de clasificación binaria para determinar si el estereotipo identificado es explícito o implícito. Los estereotipos implícitos no se expresan directamente en el texto y requieren un proceso de inferencia por parte de los anotadores. Este desafío adicional se presenta en forma de una clasificación jerárquica binaria. Los estereotipos implícitos pueden ser codificados mediante diversas estrategias, como metáforas, ironía y otras figuras retóricas, evaluaciones del grupo interno, y la sobregeneralización de un grupo social a partir de las características de algunos de sus miembros [28]. La identificación de estereotipos implícitos añade una capa de complejidad debido a la necesidad de interpretar el contexto y el subtexto de las oraciones.

3.2 Exploración y características del dataset

El dataset DETESTS-Dis empleado en este proyecto se compone de dos géneros de texto: comentarios en artículos de noticias (DETESTS) y publicaciones en Twitter que reaccionan a bulos sobre la integración de inmigrantes (StereHoax-ES).

DETESTS

El corpus DETESTS fue introducido inicialmente para la competición en IberLEF 2022 y se compone de dos partes: una del corpus NewsCom-TOX y otra del corpus StereoCom. Ambos corpus contienen comentarios publicados en respuesta a artículos de noticias en línea de diversos medios españoles y foros de discusión. NewsCom-TOX incluye comentarios de artículos seleccionados por su contenido controvertido, potencial toxicidad y número de comentarios. StereoCom se recopiló de manera similar, centrado en comentarios sobre inmigración para asegurar un volumen de datos balanceado.

Cada comentario fue segmentado en oraciones y anotado con dos características principales: la presencia de estereotipos y si estos se expresan de manera explícita o implícita. Cada oración fue anotada por tres anotadores, y se alcanzó un consenso para los casos de desacuerdo.

En total, el corpus DETESTS contiene 3.306 oraciones de NewsCom-TOX y 2.323 oraciones de StereoCom, sumando un total de 5.629 oraciones anotadas.

Algunos ejemplos de comentarios conteniendo estereotipos en este dataset son:

- *"Desgraciadamente, y visto lo visto, al final vamos a tener que aplicar lo que hace los israelíes en sus fronteras."*
- *"Primero que se civilicen y luego ya veremos."*

StereHoax-ES

El corpus StereHoax-ES contiene tweets recogidos de Twitter en 2021, que reaccionan a bulos dirigidos contra inmigrantes en España. Estos tweets se recopilaron a partir de hilos de conversación iniciados por bulos identificados manualmente en sitios de verifi-

cación como maldita.es y newtral.es. En total, se recopilaron 5.349 tweets.

En línea con el corpus DETESTS, los tweets también fueron segmentados y anotados para identificar la presencia de estereotipos y si estos eran explícitos o implícitos, empleando el mismo enfoque de anotación descrito anteriormente.

Algunos ejemplos de comentarios conteniendo estereotipos en este dataset son:

- *"Que aprenda a hablar el orco."*
- *"Eres tonto y me pones negro con esas afirmaciones. Quien te paga a ti el nivel de vida que llevas en vez de estar cuidando cabras??? Que parece ser es a lo que te tienes que dedicar, MAMON"*

Atributos del dataset

El conjunto de datos proporcionado para el entrenamiento incluye las siguientes columnas:

- **source**: {"detests", "stereohoax"}
- **id**: identificador único
- **comment_id**: identificador del comentario
- **text**: oración o tweet
- **level1** a **level4**: niveles de contexto, refiriéndose a comentarios o tweets previos y el texto original de la noticia o bulo
- **stereotype_a1**, **stereotype_a2**, **stereotype_a3**: anotaciones individuales de estereotipos
- **stereotype**: votación mayoritaria para etiqueta dura (hard label)
- **stereotype_soft**: etiqueta suave (soft label) mediante normalización softmax [33]
- **implicit_a1**, **implicit_a2**, **implicit_a3**: anotaciones individuales de implicitud
- **implicit**: votación mayoritaria para implicitud
- **implicit_soft**: etiqueta suave para implicitud

En este proyecto, se han llevado a cabo experimentos utilizando las etiquetas duras, representadas por la columna **stereotype**, que se obtuvieron mediante el método de votación mayoritaria entre los anotadores.

Restricciones

Cabe destacar que, debido a las restricciones impuestas por los términos y condiciones de la competición DETESTS-Dis, no es posible dejar el código de este proyecto accesible en un repositorio público para su reproducción completa.Estas condiciones limitan la posibilidad de compartir los datos necesarios para replicar los experimentos. Por lo tanto, cualquier interesado en la obtención de los datos debería dirigirse directamente a la página web oficial de la competición DETESTS-Dis para más información.

3.3 Retos y desafíos de la detección de estereotipos en español

La detección de estereotipos en español enfrenta varios retos significativos. En comparación con el inglés, hay menos recursos y estudios disponibles, lo que dificulta el desarrollo de modelos robustos. La variedad dialectal y cultural del español añade complejidad, ya que los modelos deben adaptarse a diferentes contextos y usos del lenguaje. Además, la subjetividad y ambigüedad de los estereotipos complican su detección. La falta de benchmarks estandarizados limita la evaluación consistente de los modelos. La infraestructura computacional necesaria para entrenar modelos de lenguaje masivos, como GPT-3.5 turbo, también puede ser una barrera significativa. Sin embargo, la aplicación de estos modelos avanzados en proyectos específicos subraya su potencial para mejorar la precisión y eficacia en la detección de estereotipos en textos en español, abordando así los desafíos mencionados.

Complejidad lingüística y cultural

Uno de los desafíos más destacados en la detección de estereotipos en textos en español es la diversidad lingüística y cultural del idioma [29]. El español es una lengua hablada en múltiples países, cada uno con sus propias variantes dialectales y culturales. Esta diversidad afecta la forma en que se expresan los estereotipos, ya que una frase que podría considerarse estereotipo en un país, puede no tener el mismo impacto o significado en otro. Por ejemplo, las palabras y frases que contienen connotaciones estereotípicas en el español de España pueden ser diferentes de aquellas en el español de México, Argentina o Colombia.

Esta variabilidad lingüística exige que los modelos de PLN sean altamente adaptables y sensibles a los contextos culturales y sociales. Un modelo debe ser capaz de entender estas sutilezas culturales para poder identificar estereotipos de manera efectiva.

Subjetividad y ambigüedad

Los estereotipos a menudo son subjetivos y pueden ser expresados de manera ambigua, lo que dificulta su detección automática [30]. Los comentarios en línea y las publicaciones en redes sociales pueden contener sarcasmo, ironía o lenguaje figurado, que los modelos de PLN deben ser capaces de interpretar correctamente. La subjetividad implica que lo que una persona percibe como un estereotipo, otra persona puede no considerarlo así.

Esta subjetividad se refleja también en el proceso de anotación de datos. Los anotadores pueden tener diferentes opiniones sobre si un texto contiene o no un estereotipo, lo que puede llevar a inconsistencias en los datos de entrenamiento. Esto, a su vez, puede afectar a la capacidad del modelo para generalizar y detectar estereotipos con precisión.

Falta de recursos y benchmarks

A diferencia del inglés, donde existen numerosos recursos y benchmarks para la detección de estereotipos y discurso de odio, el español cuenta con menos herramientas y datos disponibles [31]. Esto limita la capacidad de los investigadores para entrenar y

evaluar modelos de manera consistente. La creación de corpus anotados en español, que reflejen la diversidad dialectal y cultural del idioma, es una tarea costosa, pero esencial para el desarrollo de modelos robustos.

La falta de benchmarks estandarizados también dificulta la comparación de resultados entre diferentes estudios. Sin benchmarks claros, es complicado determinar qué enfoques son más efectivos y cómo se pueden mejorar los modelos existentes.

Requerimientos computacionales

El entrenamiento de modelos de lenguaje masivo requiere una infraestructura computacional significativa [32]. Estos modelos necesitan grandes cantidades de datos y poder de procesamiento para entrenar de manera efectiva. Por eso, muchas instituciones y equipos de investigación se encuentran con el acceso limitado a estas tecnologías.

Además, una vez entrenados, estos modelos también requieren recursos considerables para su implementación y uso en tiempo real. Esto puede ser un obstáculo para integrar los modelos de lenguaje masivo en aplicaciones prácticas, especialmente en contextos donde los recursos son limitados.

CAPÍTULO 4
Diseño

Esta sección tiene como objetivo explicar el proceso de diseño para la detección de estereotipos en textos, desde el preprocesamiento de los datos usando librerías como SpaCy [1] y el tokenizador de BERT, la elección de modelos de aprendizaje automático (ML, por su acrónimo en inglés) tradicionales y transformadores, así como las métricas de evaluación utilizadas para medir el rendimiento de los modelos.

4.1 Diseño experimental

4.1.1. Preprocesamiento de los datos

El preprocesamiento de los datos es un paso crucial antes de entrenar los modelos de aprendizaje automático. Este proceso mejora la calidad de los datos, elimina el ruido y asegura que los datos estén en un formato adecuado para el análisis. Sin un preprocesamiento adecuado, los modelos pueden producir resultados poco fiables y sesgados.

Métodos de preprocesamiento comunes incluyen:

- **Tokenización:** [34] Consiste en dividir el texto en unidades más pequeñas, como palabras o frases, conocidas como tokens. Esto es fundamental para que el modelo pueda analizar y aprender patrones a partir de las palabras individuales en el texto.
- **Eliminación de palabras vacías (stopwords):** Las stopwords son palabras comunes que suelen aparecer con mucha frecuencia pero no aportan nada significativo al análisis (por ejemplo, 'el', 'la', 'de'). Eliminarlas ayuda a reducir el ruido en los datos y mejora el rendimiento del modelo.
- **Lematización:** La lematización es el proceso de reducir las palabras a su forma base o raíz (lema). Por ejemplo, las palabras 'corriendo' y 'corre' se reducen a su lema 'correr'. Esto ayuda a normalizar las palabras y reduce la dimensionalidad del espacio de características.
- **Minúsculas:** Trata de convertir todo el texto a minúsculas para asegurar que las variaciones de capitalización no afecten al modelo, ya que "Casa" y "casa" deben considerarse equivalentes.

Por otro lado, la extracción de características o *feature extraction* [2] es una etapa esencial del preprocesamiento de texto en la cual el texto procesado se convierte en una representación numérica que puede ser utilizada por algoritmos de aprendizaje automático. Esta

[1] https://spacy.io/
[2] https://domino.ai/data-science-dictionary/feature-extraction

etapa es crucial ya que los modelos de aprendizaje automático no pueden procesar texto directamente sino que necesitan datos numéricos para funcionar.

Métodos comunes de extracción de características incluyen:

- **Bag of Words (BoW) [35]:** Esta técnica convierte el texto en un conjunto de palabras únicas presentes en el corpus, ignorando el orden pero contando la frecuencia de cada palabra. Aunque es sencillo y efectivo para ciertas tareas, BoW no considera el contexto de las palabras.
- **TF-IDF (Term Frequency-Inverse Document Frequency) [36]:** TF-IDF es una técnica que pondera las palabras en un documento basado en su frecuencia en el documento y su frecuencia inversa en el corpus completo. Esta ponderación ayuda a resaltar las palabras más informativas y a reducir la influencia de palabras comunes que aparecen en muchos documentos.
- **Word Embeddings [37]:** Los embeddings de palabras, como Word2Vec [38], GloVe [39] y FastText [40], convierten las palabras en vectores densos de dimensiones fijas. Estos vectores capturan las relaciones semánticas entre las palabras, permitiendo que el modelo entienda mejor el contexto y el significado.
- **Embeddings basados en Transformadores:** Los modelos de lenguaje como BERT [3], RoBERTa [4] y GPT-3.5 Turbo [5] generan embeddings contextuales para las palabras, lo que significa que el mismo término puede tener diferentes representaciones vectoriales dependiendo de su contexto. Estos embeddings se obtienen al alimentar el texto al modelo preentrenado, que luego produce representaciones con abundante información sobre el contexto.

Para los modelos de ML tradicionales, como Máquinas de Soporte Vectorial y Regresión Logística, el preprocesamiento incluye tareas como la eliminación de caracteres especiales, la conversión de texto a minúsculas, la eliminación de stop words, y la lematización o stemming. Estas operaciones se realizan para reducir la dimensionalidad del texto y mejorar la eficiencia del modelo. En este proyecto, hemos utilizado la librería SpaCy para llevar a cabo estas tareas. Además, para la vectorización, que es el proceso de convertir texto en una representación numérica que pueda ser utilizada por modelos de aprendizaje automático, se ha utilizado la técnica de TF-IDF, la cual refleja la importancia de una palabra en un documento en relación con una colección de documentos.

Por otro lado, los modelos basados en transformadores, como BETO, RoBERTa y GPT-3.5 Turbo, requieren un enfoque de preprocesamiento diferente. Estos modelos utilizan tokenizadores específicos que dividen el texto en subpalabras o tokens. El tokenizador de BERT, por ejemplo, convierte el texto en una secuencia de tokens que el modelo puede procesar. Además, los transformadores manejan el contexto de las palabras de manera más efectiva, por lo que el preprocesamiento incluye la segmentación en oraciones y la preservación de la estructura del texto tanto como sea posible.

El preprocesamiento de los datos varía significativamente entre los modelos de ML tradicionales y los transformadores. Mientras que los primeros se enfocan en simplifi-

[3] https://huggingface.co/docs/transformers/model_doc/bert
[4] https://huggingface.co/docs/transformers/model_doc/roberta
[5] https://platform.openai.com/docs/models/gpt-3-5-turbo

car y limpiar el texto, los transformadores requieren que se mantenga el contexto y la estructura del texto.

4.1.2. División del dataset

La división del dataset es una etapa fundamental en el proceso de desarrollo y evaluación de modelos de aprendizaje automático. Esta división asegura que el modelo pueda ser evaluado de manera objetiva y que su rendimiento generalice bien a datos no vistos anteriormente. El proceso consiste en dividir el dataset en dos conjuntos: uno de entrenamiento y otro de prueba. Esta división permite entrenar el modelo en un subconjunto de datos y evaluar su rendimiento en un conjunto separado, asegurando que el modelo no se ajuste en exceso a los datos de entrenamiento.

En la tarea DETESTS-Dis, se proporcionó un dataset de entrenamiento denominado *train.csv* que representaba el 80 % del total del corpus. Este dataset se utilizó para entrenar y evaluar el modelo con los datos de entrenamiento. Posteriormente, se proporcionó un dataset de prueba llamado *test.csv*, que representaba el 20 % restante del corpus y a partir del cual se obtuvieron las predicciones. Todos los datasets y scripts necesarios para realizar la tarea fueron proporcionados a los participantes a través de un repositorio de GitHub [6].

En este proyecto, en concreto, se aplicó una validación cruzada 5-fold sobre el conjunto de entrenamiento para asegurar la robustez y consistencia del modelo. En este método, el dataset de entrenamiento se dividió en 5 subconjuntos, entrenando y validando los modelos 5 veces, cada vez utilizando un subconjunto diferente como conjunto de validación y los 4 restantes como conjunto de entrenamiento. Este enfoque garantiza que la partición de los datos entre entrenamiento y prueba no sea aleatoria y a prevenir el sesgo de que los datos de entrenamiento y prueba sean de diferente dificultad, lo cual podría afectar la evaluación del rendimiento del modelo.

Posteriormente, para la fase de entrenamiento final del modelo, se utilizó todo el conjunto de datos de entrenamiento disponible en *train.csv*. Esto maximiza la cantidad de información que el modelo puede aprender antes de ser evaluado. Las predicciones se realizaron sobre el dataset *test.csv* que no contenía etiquetas, para asegurar una evaluación imparcial y realista del modelo.

Este proceso de división y evaluación garantiza que el modelo desarrollado sea robusto y tenga la capacidad de generalizar adecuadamente a nuevos datos, proporcionando resultados fiables y precisos.

4.2 Máquinas de Vectores de Soporte

Las Máquinas de Vectores de Soporte (SVM, por su acrónimo en inglés) son modelos de aprendizaje supervisado que se utilizan comúnmente para tareas de clasificación. Las

[6] https://github.com/clic-ub/DETESTS-Dis/tree/main Visitado el 20/03/2024.

SVM buscan el hiperplano que mejor separa las clases en el espacio de características, maximizando el margen entre las clases.

Características

- **Versatilidad:** Puede ser utilizado con kernels para problemas no lineales.
- **Robustez:** Eficaz en espacios de alta dimensionalidad y con datos ruidosos.

El objetivo de una SVM lineal es resolver el siguiente problema de optimización [41]:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad sujeto \quad a \quad y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1, \forall i$$

Donde:

- y_i son las etiquetas de las clases (-1 o 1);
- $\mathbf{x_i}$ son los vectores de características correspondientes a las instancias;
- w es el vector de pesos que define la orientación del hiperplano separador;
- $\mathbf{b_i}$ es el sesgo o término independiente que desplaza el hiperplano respecto al origen.

4.2.1. Preprocesamiento

Para el modelo SVM, se han realizado los siguientes pasos de preprocesamiento utilizando la librería *spaCy*:

Librería spaCy

spaCy es una biblioteca avanzada para el procesamiento del lenguaje natural en Python. Es especialmente conocida por su eficiencia y facilidad de uso, proporcionando herramientas robustas para tareas como el análisis morfosintáctico, etiquetado de partes del habla o *part of speech* (POS) [7], y lematización, entre otras.

- **Tokenización:** *spaCy* divide el texto en unidades más pequeñas conocidas como tokens. Esto permite que cada palabra sea procesada individualmente. La tokenización es esencial para que el modelo SVM pueda analizar y aprender patrones a partir de las palabras individuales en el texto.
- **Eliminación de palabras vacías (stopwords):** *spaCy* tiene una lista integrada de stopwords en español que se utiliza para eliminar estas palabras del texto. Esto ayuda a reducir el ruido en los datos y mejora el rendimiento del modelo al centrarse en las palabras más significativas.

[7] https://saturncloud.io/glossary/part-of-speech-tagging/#:~:text=Part%2Dof%2DSpeech%20(POS)%20tagging%20is%20the%20process,or%20a%20rule%2Dbased%20system.

- **Lematización:** *spaCy* proporciona una lematización precisa y eficiente, ayudando a normalizar las palabras y a reducir la dimensionalidad del espacio de características, lo cual es crucial para mejorar la precisión del modelo SVM.
- **Minúsculas:** La librería permite convertir todo el texto a minúsculas para asegurar que las variaciones de capitalización no afecten al modelo, ya que "Texto" y "texto" deben considerarse equivalentes. Esto asegura la consistencia en la representación de palabras.

El uso de *spaCy* para el preprocesamiento ayuda a que el texto sea limpiado y normalizado de manera eficiente antes de pasar a la fase de extracción de características y entrenamiento del modelo SVM. Este preprocesamiento es esencial para maximizar el rendimiento del modelo y garantizar que los datos sean lo más representativos y consistentes posible.

4.2.2. Extracción de características

Para el modelo SVM, se ha utilizado la técnica de extracción de características conocida como TF-IDF, implementada a través de la librería *scikit-learn*.

Librería scikit-learn

scikit-learn es una biblioteca de aprendizaje automático en Python que proporciona herramientas simples y eficientes para el análisis de datos y modelado predictivo. Es ampliamente utilizada en la comunidad de ciencia de datos y aprendizaje automático debido a su facilidad de uso y amplia gama de algoritmos implementados. En este proyecto, se utilizó específicamente el vectorizador *TfidfVectorizer* de *scikit-learn*.

- **TF-IDF:** El vectorizador *TfidfVectorizer* convierte una colección de documentos de texto en una matriz de características TF-IDF. TF-IDF es una técnica que refleja la importancia de una palabra en un documento en relación con una colección de documentos. Esta técnica combina la frecuencia de término (TF), que mide cuántas veces aparece una palabra en un documento, con la frecuencia inversa de documento (IDF), que mide cuán común o rara es una palabra en todos los documentos. El resultado es una representación numérica que pondera las palabras más significativas y disminuye el peso de las palabras más comunes.
- **Ajuste y transformación:** Primero, el vectorizador *TfidfVectorizer* se ajusta a los datos de entrenamiento para aprender el vocabulario y calcular las puntuaciones TF-IDF de cada palabra. Luego, el texto de entrenamiento se transforma en una matriz TF-IDF que se utiliza como entrada para el modelo SVM. De manera similar, el texto de prueba también se transforma utilizando el mismo vectorizador ajustado a los datos de entrenamiento, garantizando que las características de entrenamiento y prueba sean consistentes.
- **Reducción de dimensionalidad:** Al utilizar TF-IDF, se logra una reducción de dimensionalidad efectiva, ya que se descartan palabras irrelevantes o redundantes. Esto mejora la eficiencia computacional del modelo SVM y ayuda a evitar el sobreajuste, asegurando que el modelo se centre en las características más relevantes y distintivas del texto.

La utilización de *TfidfVectorizer* para la extracción de características permite que el modelo SVM se entrene y haga predicciones basadas en una representación numérica de los textos, capturando la importancia relativa de las palabras en diferentes contextos. Este enfoque es crucial para maximizar el rendimiento del modelo y garantizar que las predicciones sean precisas y relevantes.

4.2.3. Entrenamiento del modelo

Para el entrenamiento del modelo SVM, se siguió un enfoque estructurado utilizando las herramientas proporcionadas por la librería *scikit-learn*. La preparación de los datos y la configuración del modelo se realizaron de manera meticulosa para asegurar un rendimiento óptimo.

Configuración del modelo

Se utilizó el algoritmo de Máquina de Vectores de Soporte (SVM) con el clasificador *SVC* de *scikit-learn*. Este clasificador se configuró para habilitar la probabilidad en las predicciones, lo cual es útil para obtener una estimación más precisa de las clases y evaluar la confianza del modelo en sus predicciones.

Proceso de entrenamiento

Una vez preprocesados y transformados los textos mediante el vectorizador TF-IDF, se procedió a entrenar el modelo SVM. El conjunto de entrenamiento, transformado en una matriz de características TF-IDF, se utilizó como entrada para el modelo SVM junto con las etiquetas correspondientes, en este caso, la etiqueta *stereotype* para la tarea 1 y la etiqueta *implicit* para la tarea 2. El modelo se ajustó a estos datos, de manera que pudo aprender a distinguir las clases basándose en los patrones encontrados en las características textuales.

El entrenamiento se llevó a cabo en dos fases distintas para abordar las diferentes tareas:

1. **Detección de estereotipos**
   El modelo SVM fue entrenado para detectar estereotipos en el texto. Utilizando las características extraídas del texto preprocesado, el modelo aprendió a clasificar los textos distinguiendo entre textos que contenían estereotipos y textos que no.

2. **Clasificación de implicitud**
   En una segunda fase, se filtraron los textos que fueron clasificados como que contenían estereotipos de la primera fase. Luego, el modelo SVM se entrenó para clasificar aquellos textos que contenían estereotipos como implícitos o explícitos. Este modelo también utilizó la matriz de características TF-IDF, adaptada para esta tarea específica. El entrenamiento se realizó utilizando las etiquetas correspondientes para esta clasificación.

4.2.4. Evaluación y predicción

Después del entrenamiento, el modelo SVM se utilizó para hacer predicciones sobre el conjunto de datos de prueba. Las predicciones se realizaron sobre la matriz TF-IDF transformada del conjunto de prueba, asegurando que el modelo aplicara el conocimiento adquirido durante el entrenamiento para clasificar los nuevos textos. Las predicciones incluyeron la detección de estereotipos seguidas de su clasificación entre implicícitos o explícitos, y se almacenaron para su análisis posterior.

Este enfoque para el entrenamiento del modelo SVM garantiza que el modelo esté bien adaptado a las características específicas del texto en español, logrando una alta precisión y robustez en las tareas de clasificación asignadas.

4.3 Regresión Logística

La Regresión Logística es otro modelo de aprendizaje supervisado utilizado para tareas de clasificación binaria. Este modelo estima la probabilidad de que una instancia pertenezca a una clase particular utilizando una función logística.

Características

- **Probabilidad:** Proporciona probabilidades de pertenencia a clases.
- **Eficiencia:** Rápido de entrenar y ejecutar en grandes conjuntos de datos.

La fórmula de la regresión logística se define como [42]:

$$logit(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

Donde:

- p es la probabilidad de que la instancia pertenezca a la clase positiva;
- β_0 es el término independiente;
- $\beta_1, \beta_2, \ldots, \beta_n$ son los coeficientes de los predictores $x_1, x_2, \ldots, x_n$.

4.3.1. Preprocesamiento

Para el modelo de Regresión Logística, se han realizado los mismos pasos de preprocesamiento que para el modelo SVM utilizando la librería *spaCy*:

- **Tokenización:** El texto se divide en unidades más pequeñas conocidas como tokens.

- **Eliminación de palabras vacías (stopwords):** Se eliminan las palabras vacías para reducir el ruido en los datos.
- **Lematización:** Las palabras se normalizan para reducir la dimensionalidad del espacio de características.
- **Minúsculas:** Todo el texto se convierte a minúsculas para asegurar la consistencia en la representación de palabras.

El preprocesamiento, al igual que en SVM, ayuda a que el texto sea limpiado y normalizado de manera eficiente antes de pasar a la fase de extracción de características y entrenamiento del modelo de Regresión Logística.

4.3.2. Extracción de características

Para la Regresión Logística, también se ha utilizado la técnica de extracción de características conocida como TF-IDF, implementada a través de la librería *scikit-learn* y utilizando el vectorizador *TfidfVectorizer* de *scikit-learn*.

- **TF-IDF:** El vectorizador *TfidfVectorizer* convierte los documentos de texto en una matriz de características ponderando las palabras según su importancia.
- **Ajuste y transformación:** El vectorizador se ajusta a los datos de entrenamiento y transforma tanto el texto de entrenamiento como el de prueba en matrices TF-IDF consistentes.
- **Reducción de dimensionalidad:** TF-IDF ayuda a reducir la dimensionalidad, mejorando la eficiencia y evitando el sobreajuste.

4.3.3. Entrenamiento del modelo

De la misma manera que en el anterior modelo, para el entrenamiento del modelo de Regresión Logística, se siguió un enfoque estructurado utilizando las herramientas proporcionadas por la librería *scikit-learn*.

Configuración del modelo

Se utilizó el clasificador de Regresión Logística de *scikit-learn*, que se configura para modelar la probabilidad de que una instancia pertenezca a una clase particular. En nuestro caso, para la tarea de detección de estereotipos, la clase positiva es aquella que indica la presencia de estereotipos, mientras que la clase negativa indica su ausencia. Para la tarea de clasificación de implicitud, la clase positiva indica que un texto es implícito, mientras que la clase negativa indica que es explícito.

Proceso de entrenamiento

Después de preprocesar y convertir los textos con TF-IDF, se entrenó el modelo de Regresión Logística usando la matriz de características y las etiquetas del conjunto de entrenamiento. Esto permitió al modelo aprender a identificar los patrones necesarios para diferenciar entre clases.

De la misma forma, el entrenamiento se realizó en dos partes para cada una de las tareas del DETESTS-Dis:

1. **Detección de estereotipos**

   A partir de las características obtenidas tras el preprocesamiento, el modelo desarrolló la capacidad de distinguir entre textos que contenían estereotipos y aquellos que no los presentaban.

2. **Clasificación de implicitud**

   En una segunda fase, se filtraron los textos que fueron clasificados como contenedores estereotipos en la primera fase. Luego, se entrenó el modelo de Regresión Logística para diferenciar entre textos con estereotipos implícitos y explícitos. Para esta tarea específica, se usaron nuevamente las matrices TF-IDF y las etiquetas adecuadas para la clasificación.

4.3.4. Evaluación y predicción

Finalizado el entrenamiento, se usó el modelo de Regresión Logística para predecir en el conjunto de prueba, aplicando la matriz TF-IDF correspondiente. El modelo clasificó los textos para detectar estereotipos y determinar si eran implícitos o explícitos, almacenando los resultados para la posterior evaluación del modelo.

4.4 BETO

BETO [8] es un modelo de lenguaje masivo transformador preentrenado específicamente para el idioma español, basado en la arquitectura BERT [43]. Este modelo ha sido entrenado con una gran cantidad de texto en español, lo que le permite capturar las sutilezas y matices del idioma. BETO es capaz de entender el contexto y la semántica de los textos.

Características

- **Especializado en el español:** Capta y maneja con precisión las particularidades y estructuras gramaticales del idioma.
- **Contexto y semántica:** Utiliza la arquitectura de BERT para capturar el contexto bidireccional de las palabras en una oración, entendiendo mejor el significado de cada palabra basada en su entorno.

[8] https://github.com/dccuchile/beto

- **Flexibilidad:** Puede ser adaptado para diversas tareas de PLN como clasificación de texto, análisis de sentimientos, y detección de estereotipos.

Funcionamiento

BETO, al igual que BERT [44], utiliza la arquitectura de transformadores basada en la atención (Figura 4.1) para procesar el texto. La principal innovación de esta arquitectura es el mecanismo de atención, que permite al modelo considerar el contexto completo de una palabra analizando todas las palabras en la oración simultáneamente.

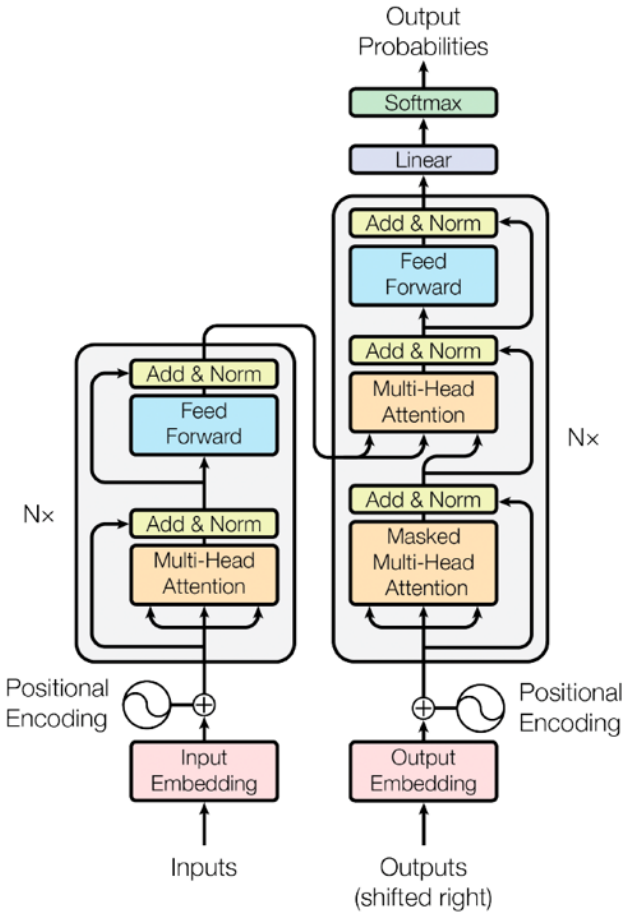

Figura 4.1: Arquitectura de transformadores basada en la atención. Imágen extraída de [9]

El mecanismo de atención se describe mediante la fórmula de atención escalada por puntos [45]:

$$Atención(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Donde:

- Q son las consultas;
- K son las claves;

- V son los valores;
- d_k es la dimensión de las claves.

El proceso de entrenamiento de BETO implica dos fases:

- **Preentrenamiento**

  - **Modelado de lenguaje enmascarado** (MLM, según su acrónimo en inglés): Se ocultan aleatoriamente algunas palabras del texto y el modelo intenta predecirlas basándose en el contexto.
  - **Predicción de la siguiente oración** (NSP, según su acrónimo en inglés): El modelo intenta predecir si dos oraciones consecutivas pertenecen al mismo contexto.

- **Fine-tuning**

  En esta fase, se ajusta el modelo ya preentrenado a una tarea específica, con un conjunto de datos específico para la tarea que se desea realizar, como la clasificación de texto, respuestas a preguntas, resumen de textos o, como en este caso, detección de estereotipos. Durante el ajuste fino, se actualizan todos los parámetros del modelo preentrenado.

4.4.1. Preprocesamiento

El preprocesamiento para el modelo BETO se realiza utilizando la librería *transformers* de Hugging Face [9]. A diferencia de los modelos tradicionales como SVM y Regresión Logística, donde se utilizan técnicas como la tokenización y lematización con *spaCy*, BETO emplea un tokenizador específico para modelos basados en transformadores.

- **Tokenización:** Se utiliza el *BertTokenizer* para dividir el texto en tokens. Este tokenizador maneja tanto la segmentación en palabras como la subtokenización, que consiste en dividir palabras raras o desconocidas en subcomponentes más pequeños, permitiendo que el modelo maneje de manera efectiva vocabularios extensos y preserve el contexto completo del texto.
- **Padding y truncation:** Los textos se rellenan o se truncan a una longitud fija (en este caso, 128 tokens) para garantizar que todos los ejemplos tengan la misma longitud. Esto es crucial para el procesamiento en lotes (batch processing), ya que permite que el modelo procese múltiples secuencias simultáneamente de manera eficiente. Al tener todas las secuencias de la misma longitud, se optimiza el uso de memoria y se acelera el entrenamiento del modelo.

[9] https://huggingface.co/dccuchile/bert-base-spanish-wwm-uncased

Este enfoque asegura que BETO procese los textos de una manera que maximice la retención del contexto y la estructura del idioma.

4.4.2. Extracción de características

Para BETO, la extracción de características se integra directamente en el proceso de tokenización y modelado mediante la librería *transformers*. Al tokenizar el texto, se generan secuencias de tokens que se convierten en vectores de entrada para el modelo. Este proceso es diferente de los enfoques tradicionales como TF-IDF, ya que las características contextuales y semánticas se capturan directamente a través del modelo preentrenado.

4.4.3. Entrenamiento del modelo

El entrenamiento del modelo BETO se lleva a cabo utilizando la librería *transformers* de Hugging Face y PyTorch. A continuación, se detallan los pasos principales del entrenamiento:

Configuración del modelo: Se utiliza *BertForSequenceClassification*, configurado para clasificación binaria con dos etiquetas.

Proceso de entrenamiento: Una vez tokenizados los textos, se crea un dataset personalizado que incluye tanto las secuencias de tokens como las etiquetas correspondientes. Este dataset se utiliza para entrenar el modelo en dos fases distintas, siguiendo el enfoque jerárquico binario:

1. **Primera fase: Detección de estereotipos**

   El modelo se entrena para identificar la presencia de estereotipos en el texto. Utilizando las características tokenizadas, el modelo aprende a clasificar los textos basándose en patrones contextuales y semánticos.

2. **Segunda fase: Clasificación de implicitud para textos con estereotipos**

   En esta fase, se filtran los textos que fueron clasificados como contenedores de estereotipos en la primera fase. Luego, el modelo se entrena para distinguir entre textos con estereotipos implícitos y explícitos. Se utiliza la misma estructura de dataset y tokenización, pero aplicando etiquetas para la implicitud del texto.

4.4.4. Evaluación y predicción

Después del entrenamiento, se utiliza el modelo BETO para predecir en el conjunto de datos de prueba. Las predicciones se realizan aplicando el modelo entrenado a los datos tokenizados del conjunto de prueba. Los resultados incluyen tanto la detección de estereotipos como la clasificación de implicitud y se almacenan para su análisis posterior.

4.5 RoBERTa

RoBERTa (Robustly optimized BERT approach) es otro modelo de lenguaje masivo basado en la arquitectura Transformer, en este caso, es una variante mejorada de BERT, optimizada para un preentrenamiento más robusto. Este modelo ajusta varios hiperparámetros y se entrena en un mayor volumen de datos, lo que le permite manejar grandes volúmenes de información y extraer características profundas del texto. RoBERTa mejora la precisión en diversas tareas de procesamiento de lenguaje natural (PLN), incluida la clasificación binaria.

Características:

- **Optimización del preentrenamiento:** Ajusta hiperparámetros clave como la longitud de secuencia y el tamaño del lote, y se entrena durante más pasos, lo que permite un mejor ajuste del modelo a los datos.
- **Manejo de grandes volúmenes de datos:** Entrenado con una gran cantidad de datos, lo que mejora su capacidad para generalizar y manejar diferentes contextos y dominios de texto.
- **Extracción de características profundas:** Su capacidad para capturar relaciones complejas dentro del texto lo hace ideal para tareas que requieren una comprensión profunda del lenguaje.

Funcionamiento:

RoBERTa, al igual que BETO, utiliza la arquitectura de transformadores basada en la atención para procesar el texto, permitiendo al modelo considerar el contexto completo de una palabra mediante el análisis simultáneo de todas las palabras en una oración.

El proceso de entrenamiento de RoBERTa incluye dos fases principales, similares a las de BETO, pero con algunas diferencias notables:

- Preentrenamiento:
  - *Modelado de lenguaje enmascarado* (MLM): Similar a BETO, oculta palabras aleatorias del texto y el modelo intenta predecirlas basándose en el contexto.
- Fine-tuning:
  Después del preentrenamiento, RoBERTa puede ser afinado en tareas específicas de PLN, como clasificación de texto, análisis de sentimientos y, en nuestro caso, detección de estereotipos en comentarios de texto en español.

4.5.1. Preprocesamiento

El preprocesamiento para RoBERTa se realiza utilizando la combinación de *spaCy* y la librería *transformers* de Hugging Face. A diferencia de los enfoques tradicionales como SVM y Regresión Logística, y de manera similar al modelo BETO, el preprocesamiento con RoBERTa incluye los siguientes pasos:

- **Tokenización y lematización:** Inicialmente, se usa *spaCy* para dividir el texto en tokens y lematizar las palabras. Este paso también incluye la eliminación de stopwords y puntuación, lo que ayuda a limpiar el texto. Este proceso es similar al utilizado en los modelos tradicionales.

- **Tokenización para RoBERTa:** Luego, se emplea el *RobertaTokenizer* para convertir el texto preprocesado en tokens específicos de RoBERTa. Al igual que el *BertTokenizer* utilizado con BETO, el *RobertaTokenizer* maneja tanto la segmentación en palabras como la subtokenización, fragmentando palabras poco comunes o desconocidas en partes más pequeñas.

- **Padding y truncation:** Se ajustan los textos a una longitud fija (en este caso, 512 tokens) ya sea rellenándolos o truncándolos, de modo que todos los ejemplos tengan la misma extensión. Esta técnica es similar a la utilizada en BETO, aunque la longitud de los tokens puede variar según las necesidades específicas del modelo.

4.5.2. Extracción de características

En RoBERTa, la extracción de características ocurre simultáneamente con la tokenización y el modelado utilizando la librería *transformers*. Además, las secuencias de tokens resultantes se transforman en vectores de entrada. Este enfoque es similar al utilizado por BETO, donde las características se integran directamente en el proceso de tokenización y no requieren técnicas adicionales como TF-IDF utilizadas en modelos tradicionales como SVM y Regresión Logística.

4.5.3. Entrenamiento del modelo

El entrenamiento del modelo RoBERTa, al igual que BETO, se lleva a cabo utilizando la librería *transformers* de Hugging Face y PyTorch. A continuación, se detallan los pasos principales del entrenamiento:

Configuración del modelo: En este caso, se utiliza *RobertaForSequenceClassification* para la clasificación binaria.

Proceso de entrenamiento: De la misma forma que BETO, se genera un conjunto de datos que contiene tanto las secuencias de tokens como sus respectivas etiquetas, el cual se utiliza para entrenar el modelo tanto para la detección de estereotipos como la clasificación de implicitud.

1. **Detección de estereotipos**

   Para la detección de estereotipos, este modelo pre-entrenado también se ajusta con fine-tuning usando los datos específicos del dataset *train*. Utilizando las características tokenizadas, el modelo aprende a clasificar los textos identificando patrones. Este proceso es similar al de BETO, aunque RoBERTa puede incluir ajustes adicionales en los hiperparámetros debido a sus optimizaciones específicas.

2. **Clasificación de implicitud**

   En esta segunda clasificación y después de haber filtrado el dataset para clasificar solo aquellos comentarios que contienen estereotipos, el modelo se entrena para distinguir entre estereotipos implícitos y explícitos. Este proceso consiste en una clasificación jerárquica binaria como así se enfocaba en la tarea DETESTS-Dis.

4.5.4. Evaluación y predicción

Finalmente, con el modelo entrenado, usamos RoBERTa para sacar las predicciones de estereotipos e implicitud en el conjunto de datos de prueba. Las predicciones acaban siendo guardadas en un fichero para su futura evaluación.

4.6 GPT-3.5 Turbo

GPT-3.5 Turbo (Generative Pre-trained Transformer) es un modelo de lenguaje masivo generativo de transformador preentrenado desarrollado por OpenAI [10]. Este modelo es conocido por su capacidad para generar texto coherente y relevante contextualmente, así como por su capacidad de aprendizaje en pocos disparos (few-shot learning), es decir, con pocas muestra como ejemplo.

Características

- **Tamaño:** Contiene miles de millones de parámetros que permiten capturar una amplia gama de conocimientos y patrones lingüísticos.
- **Versatilidad:** Es capaz de realizar una variedad de tareas de procesamiento de lenguaje natural, incluyendo traducción, resumen, generación de texto, y más.
- **Entrenamiento:** Fue entrenado con grandes cantidades de texto de internet, lo que le proporciona un vasto conocimiento general y una capacidad para generar texto coherente en diversos contextos.

Funcionamiento

El modelo GPT-3.5 Turbo utiliza la arquitectura Transformer (Figura 4.1), que es especialmente adecuada para tareas de PLN debido a su capacidad para manejar dependencias a largo plazo en el texto. A diferencia de modelos como BETO y RoBERTa, que se basan exclusivamente en la parte del encoder [46] del diagrama de la arquitectura Transformer, GPT-3.5 Turbo utiliza la parte del decoder. Este enfoque se caracteriza por una atención auto-regresiva, donde el modelo predice cada palabra del texto secuencialmente, basándose únicamente en las palabras anteriores. Esta metodología permite que GPT-3.5 Turbo genere texto de manera fluida y coherente, adaptándose al contexto proporcionado por las entradas anteriores.

[10] https://openai.com/

- **Entrada:** El modelo recibe una secuencia de texto como entrada, que puede incluir instrucciones, preguntas o cualquier otro tipo de contexto.
- **Procesamiento:** Utiliza múltiples capas de atención y mecanismos de *feed-forward* [11] para procesar la entrada y generar una representación interna del texto.
- **Salida:** Produce una secuencia de texto como salida, prediciendo la siguiente palabra en la secuencia basada en las anteriores hasta completar la generación del texto deseado.

Usos

GPT-3.5 Turbo se utiliza en una variedad de aplicaciones, tales como:

- **Generación de texto:** Creación de artículos, historias, poesía y otros contenidos creativos.
- **Asistentes virtuales:** Responder preguntas, proporcionar información y asistir en la realización de tareas.
- **Traducción automática:** Convertir texto de un idioma a otro con alta precisión.

El modelo GPT-3.5 Turbo representa un avance significativo en el campo del procesamiento de lenguaje natural, proporcionando herramientas poderosas para la automatización y mejora de la comprensión del lenguaje humano.

4.6.1. Preprocesamiento

El preprocesamiento para GPT-3.5 Turbo es bastante minimalista comparado con los otros modelos como BETO y RoBERTa. En lugar de realizar una tokenización y lematización extensiva, el enfoque se centra en limpiar el texto eliminando menciones y URLs, para lo cual se utiliza la biblioteca *re* [12] en Python. Este paso asegura que el texto de entrada esté limpio y listo para ser procesado por el modelo generativo.

4.6.2. Extracción de características

A diferencia de modelos tradicionales como SVM y Regresión Logística, que pueden requerir el uso de matrices de características como TF-IDF, GPT-3.5 Turbo procesa el texto utilizando tokenizadores que convierten el texto en secuencias de tokens sin la necesidad de técnicas de vectorización tradicionales. La extracción de características se integra directamente en el modelo mediante el uso de su capacidad de atención auto-regresiva, que analiza las relaciones contextuales en el texto durante la generación.

[11] https://www.turing.com/kb/mathematical-formulation-of-feed-forward-neural-network#ending-note

[12] https://docs.python.org/es/3/library/re.html

4.6.3. Entrenamiento del modelo

El modelo GPT-3.5 Turbo ya viene preentrenado, por lo que no se realiza un entrenamiento adicional tradicional como en BETO o RoBERTa. En cambio, se utiliza un enfoque basado en consultas directas (*prompts*) para guiar al modelo en la tarea deseada. Para la competición DETESTS-Dis, se presentaron los resultados del modelo GPT con *zero-shot learning*, es decir, sin dar ninguna muestra de datos como ejemplo en el prompt. No obstante, en un momento más avanzado del trabajo, se decidió experimentar también con el modelo usando la técnica de *few-shot learning* para aydarle a comprender mejor la tarea y poder analizar mejor el desempeño del modelo en diferentes escenarios.

Para la detección de estereotipos, se sigue un proceso específico:

- Con *zero-shot learning* se utiliza una muestra del conjunto de datos y se realizan preguntas directas al modelo sobre la presencia de estereotipos en la muestra proporcionada. El modelo genera una respuesta que se convierte en una predicción binaria.
- Se prepara un sistema de mensajes que incluyen instrucciones específicas al modelo, indicándole que identifique si un texto contiene estereotipos y que dé la respuesta en binario. En el caso del *few-shot learning*, se proporciona al modelo una serie de ejemplos previamente etiquetados que contienen tanto preguntas como respuestas, con el fin de contextualizar y guiar mejor al modelo.

En la segunda tarea (detección de implicitud), se emplea una estrategia de clasificación jerárquica binaria:

- Primero, se filtran los textos que contienen estereotipos utilizando las predicciones anteriormente generadas por el modelo.
- Luego, se hace una segunda consulta al modelo GPT-3.5 Turbo, esta vez pidiéndole que clasifique los textos identificados como implícitos o explícitos, aplicando de la misma forma los diferentes enfoques de uso del modelo, *zero-shot learning* y *few-shot learning* (proporcionando ejemplos etiquetados de textos explícitos e implícitos).

El proceso de configuración y ejemplos utilizados para *few-shot learning* se encuentran descritos en el Apéndice A.

4.6.4. Evaluación y predicción

Después de aplicar los métodos descritos, las predicciones se almacenan y analizan. El modelo genera respuestas textuales que se convierten en etiquetas binarias.

Este enfoque con GPT-3.5 Turbo destaca por su capacidad de manejar tareas complejas de PLN con un mínimo de preprocesamiento y su flexibilidad para adaptarse a diferentes tipos de consultas y contextos textuales.

4.7 Diseño de evaluación y métricas

Para evaluar los modelos y las tareas, se han utilizado scripts de plantilla proporcionados en el repositorio de GitHub de la tarea [13]. Estos scripts ayudan a formatear los resultados en la estructura correcta (JSON) y a realizar evaluaciones utilizando las métricas F1 y la métrica basada en la información de la clasificación jerárquica.

Primero, se transforma el conjunto de datos de prueba al formato necesario para la evaluación. Los scripts de evaluación consideran etiquetas duras (*hard labels*), que se utilizan para comparar las predicciones del modelo con el *gold standard* [47].

Para la tarea 1, se utiliza la métrica estándar de clasificación F1 para comparar las predicciones del modelo con el *gold standard*.

Para la tarea 2, se implementa una clasificación jerárquica binaria. En esta tarea, se utiliza la métrica ICM (*Information Contrast Model*) [48] que considera tanto la estructura jerárquica como la especificidad de la clase. Esta métrica se aplica a las etiquetas duras.

El script de evaluación realiza las siguientes funciones:

- Calcula las métricas de evaluación según el tipo de tarea y el tipo de etiqueta (dura en este caso).
- Genera informes detallados de evaluación que incluyen precisión, *recall*, F1, la métrica ICM y la ICM normalizada.

Estos scripts automatizan y estandarizan el proceso de evaluación, asegurando que los resultados sean comparables y reproducibles. Las métricas utilizadas permiten una evaluación detallada y exhaustiva de los modelos, capturando tanto el rendimiento en términos de precisión como la capacidad de manejar la estructura jerárquica de las clases.

Métricas utilizadas

- **Precisión**
  Es una métrica que indica la proporción de instancias correctamente clasificadas como positivas entre todas las instancias que fueron clasificadas como positivas por el modelo. Es una medida de la exactitud de las predicciones positivas del modelo. Matemáticamente, se define como:

$$Precisión = \frac{TP}{TP + FP}$$

  Donde:

  - TP (True Positives) son los verdaderos positivos;

[13] https://github.com/clic-ub/DETESTS-Dis/blob/main/evaluation.py

- FP (False Positives) son los falsos positivos.

- **Recall**
  También conocido como sensibilidad, mide la proporción de instancias correctamente clasificadas como positivas entre todas las instancias que son realmente positivas. Indica la capacidad del modelo para identificar todas las instancias positivas. Matemáticamente, se define como:

$$Recall = \frac{TP}{TP + FN}$$

  Donde:

  - TP (True Positives) son los verdaderos positivos;
  - FN (False Negatives) son los falsos negativos.

- **F1 Score:**
  La métrica F1 es la media armónica de la precisión y el *recall*. Es una métrica que combina ambas para proporcionar una única medida de rendimiento, especialmente útil cuando se necesita un equilibrio entre precisión y *recall*. La fórmula para el F1 score es:

$$F1 = 2 \cdot \frac{Precisión \cdot Recall}{Precisión + Recall}$$

  El F1 score es particularmente útil en situaciones donde las clases están desbalanceadas y se necesita un balance entre falsos positivos y falsos negativos.

- **Modelo de Contraste de la Información (ICM)**
  Esta es una métrica más compleja y avanzada que considera tanto la estructura jerárquica de las clases como la especificidad de cada clase en la evaluación de los modelos.

  La ICM se basa en la teoría de la información y mide la cantidad de información compartida entre las predicciones del modelo y las etiquetas verdaderas, considerando la estructura jerárquica de las categorías. La ICM se puede descomponer en varios componentes que reflejan diferentes aspectos del rendimiento del modelo:

  - **ICM** evalúa la consistencia de las predicciones con respecto a la estructura jerárquica de las clases.
  - **ICM Normalizada** proporciona una versión normalizada de la métrica para facilitar la comparación entre diferentes modelos y conjuntos de datos.

  La fórmula básica de la métrica ICM se puede expresar como:

$$ICM(A, B) = \alpha_1 \cdot IC(A) + \alpha_2 \cdot IC(B) - \beta \cdot IC(A \cup B)$$

Donde:

- A y B son conjuntos de características;
- $IC(X) = -\log(P(X))$ representa el contenido de información del conjunto de características X;
- α_1, α_2 y β son parámetros que satisfacen $\alpha_1, \alpha_2 < \beta < \alpha_1 + \alpha_2$.

En resumen, estas métricas proporcionan una evaluación detallada del rendimiento de los modelos en las tareas de clasificación de estereotipos y implicitud, considerando tanto la exactitud como la capacidad de los modelos para manejar la estructura jerárquica de las clases.

CAPÍTULO 5
Desarrollo

Este capítulo describe las decisiones finales del proceso de desarrollo del proyecto, incluyendo las tecnologías y lenguajes utilizados, así como los problemas encontrados durante el desarrollo y cómo se resolvieron. Se pretende ofrecer una visión integral de las herramientas y métodos empleados, proporcionando una guía del flujo de trabajo y las decisiones técnicas adoptadas.

5.1 Tecnologías y lenguajes utilizados

Para llevar a cabo este proyecto, se han utilizado diversas tecnologías y lenguajes de programación que facilitan el procesamiento y análisis de datos, así como la implementación y evaluación de modelos de aprendizaje automático. A continuación, se describen las principales tecnologías utilizadas:

- **Python** [1]
  Es un lenguaje de programación ampliamente utilizado en ciencia de datos y aprendizaje automático debido a su simplicidad y versatilidad. Proporciona una amplia gama de bibliotecas y herramientas que facilitan la manipulación de datos, la implementación de modelos y la automatización de tareas.

- **Google Colab** [2]
  Es una plataforma gratuita de Jupyter notebooks que permite ejecutar código Python en la nube. Es especialmente útil para el procesamiento de grandes volúmenes de datos y el entrenamiento de modelos de aprendizaje automático, ya que proporciona acceso a GPUs y TPUs de manera gratuita.

- **GitHub** [3]
  Es una plataforma de control de versiones y colaboración que permite gestionar y compartir el código del proyecto. Se utilizó un repositorio proporcionado por la tarea DETESTS-Dis como fuente de consultas que incluía scrips para la evaluación de los modelos y para el formato de los resultados. Además, es donde se subieron los resultados correctos en un dataset final para poder comparar los resultados.

- **pandas** [4]
  Es una biblioteca de Python para la manipulación y análisis de datos. Proporciona estructuras de datos flexibles y eficientes, como DataFrames, que facilitan la limpieza, transformación y análisis de grandes conjuntos de datos.

[1] https://docs.python.org/3/
[2] https://colab.google/
[3] https://github.com/
[4] https://pandas.pydata.org/

- **scikit-learn** [5]
  Es una biblioteca de aprendizaje automático en Python que ofrece herramientas simples y eficientes para el análisis de datos y la modelización predictiva. Se utilizó para la implementación de modelos tradicionales como SVM y regresión logística.

- **spaCy** [6]
  Es una biblioteca de procesamiento de lenguaje natural (PLN) en Python que proporciona herramientas para la tokenización, lematización, eliminación de stopwords y otras tareas de preprocesamiento de texto. Se utilizó para preparar los datos antes de entrenar los modelos.

- **json** [7]
  La biblioteca json en Python se utilizó para la manipulación de datos en formato JSON. Específicamente, se empleó para guardar y cargar los resultados de las predicciones en el formato requerido para la evaluación.

- **OpenAI** [8]
  La API de OpenAI se utilizó para interactuar con el modelo GPT-3.5 Turbo. Esta API permite realizar consultas al modelo y obtener respuestas generadas.

- **torch** [9]
  torch es una biblioteca de Python para el desarrollo y entrenamiento de modelos de aprendizaje profundo. Proporciona una interfaz flexible y eficiente para la creación de redes neuronales. Se utilizó junto con la biblioteca transformers para entrenar los modelos de lenguaje masivo BETO y RoBERTa.

- **transformers** [10]
  Esta biblioteca de Hugging Face proporciona herramientas y modelos preentrenados para tareas de PLN. Facilita la implementación y el ajuste fino de modelos de lenguaje como BETO y RoBERTa.

5.2 Problemas encontrados

En esta sección se detallan los principales problemas encontrados durante el desarrollo del proyecto y las soluciones adoptadas para resolverlos. Se abordan tanto los desafíos técnicos relacionados con el diseño como los problemas de gestión del proyecto.

- **Limitaciones de recursos**
  Al inicio del proyecto, se intentó ejecutar y entrenar los modelos en una máquina local. Sin embargo, esto presentó varios problemas significativos. El entrenamiento de modelos grandes como BETO y RoBERTa en una máquina local resultó ser extremadamente lento debido a la falta de potencia de cálculo y recursos limitados. Además, los tiempos de ejecución se volvieron impracticables para un desarrollo ágil.

[5] https://scikit-learn.org/stable/
[6] https://spacy.io/
[7] https://www.json.org/json-en.html
[8] https://openai.com/
[9] https://pytorch.org/docs/stable/library.html
[10] https://huggingface.co/docs/transformers/index

Para superar estas limitaciones, se decidió cambiar el entorno de ejecución a Google Colab. Google Colab proporciona acceso gratuito a GPUs y TPUs, lo que mejora significativamente la velocidad de entrenamiento y ejecución de los modelos. Esta migración no solo permitió entrenar los modelos de manera más eficiente, sino que también ofreció la flexibilidad de utilizar múltiples entornos de ejecución gratuitos para mayor optimización.

- **Uso incorrecto de modelos preentrenados**
  Durante el uso del modelo RoBERTa, se produjo un error debido a una confusión en la documentación. Inicialmente, se utilizó incorrectamente el modelo preentrenado BERT en lugar de RoBERTa. Este error se identificó al revisar los resultados inconsistentes y al compararlos con las expectativas del modelo RoBERTa.

  La solución a este problema fue cambiar al modelo correcto, utilizando *PlanTL-GOB-ES/roberta-base-bne*. Este ajuste permitió al modelo aprovechar plenamente las capacidades de RoBERTa, mejorando así la precisión y efectividad en las tareas de procesamiento de lenguaje natural en español.

- **Errores en clasificación jerárquica binaria**
  Otro contratiempo significativo se encontró en la implementación de la clasificación jerárquica binaria para la segunda tarea. Inicialmente, la clasificación de implicitud se realizó sobre todas las filas de las predicciones de 'stereotype', en lugar de filtrar primero aquellas que se habían predicho como contenedoras de un estereotipo. Esto llevó a resultados incorrectos y a una evaluación inexacta de la clasificación de implicitud ya que los modelos intentaban predecir la implicitud también en aquellos textos que directamente no contenían ningún estereotipo.

  Para resolver este problema, se ajustó el proceso de clasificación, de modo que primero se filtraran los comentarios predichos que contenían estereotipos. De esta manera, solo a estas filas se les aplicó la clasificación de implicitud.

CAPÍTULO 6
Marco experimental

En este capítulo se describen los experimentos realizados para evaluar el rendimiento de los distintos modelos utilizados en este proyecto. Se detallan los resultados obtenidos en cada fase de evaluación y se realiza una comparación entre los modelos para determinar cuál ofrece el mejor rendimiento en las tareas planteadas.

6.1 Resultados

En esta sección se presentan los resultados obtenidos por los modelos en las diferentes fases de evaluación. Se incluyen los resultados de la validación con el conjunto de entrenamiento utilizando validación cruzada (5-fold) [49] para la tarea de detección de estereotipos, así como los resultados de la evaluación con el conjunto de test para esa tarea y para la detección de implicitud. Todas las métricas reportadas estan basadas en la clase positiva de las etiquetas ya que la evaluación en la competición DETESTS-Dis se hizo de esta forma. La "clase positiva" es la categoría de mayor interés para nuestro análisis, en este caso es: comentarios con estereotipos.

A continuación, se muestra un ejemplo de cómo se representan los resultados de las predicciones realizadas por los modelos, siguiendo el formato JSON estándar utilizado en este estudio:

```
[
    {
        "test_case": "DETESTS-Dis",
        "id": "s_4277",
        "value": "NoStereotype"
    },
    {
        "test_case": "DETESTS-Dis",
        "id": "s_4278",
        "value": "NoStereotype"
    },
    {
        "test_case": "DETESTS-Dis",
        "id": "s_4279",
        "value": "Stereotype"
    }
]
```

Cabe recordar que, en la competición DETESTS-Dis, se presentó el modelo GPT utilizando *zero-shot learning*. Sin embargo, en fases posteriores del trabajo se decidió experimentar también con *few-shot learning*, proporcionando 1, 3 y 5 ejemplos del *train* set en

el prompt para evaluar el impacto en el desempeño del modelo en la detección de estereotipos y implicitud. Por ejemplo, con 1 ejemplo para la tarea 1, se dió un ejemplo de comentario con estereotipo y 1 ejemplo de comentario sin estereotipo. Lo mismo con 3 y 5 ejemplos así como para la segunda tarea.

6.1.1. Validación training dataset: Detección de estereotipos

Modelo	Métrica	Valor
Regresión Logística	F1	**0,46**
	Precisión	0,75
	Recall	0,35
SVM	F1	**0,61**
	Precisión	0,80
	Recall	0,50
GPT-3.5 Turbo (*zero-shot learning*)	F1	**0,51**
	Precisión	0,38
	Recall	0,79
BETO	F1	**0,50**
	Precisión	0,86
	Recall	0,31
RoBERTa	F1	**0,67**
	Precisión	0,70
	Recall	0,64

Cuadro 6.1: Resultados de las métricas obtenidas en la validación con el dataset de entrenamiento para la tarea 1: Detección de estereotipos.

En la Tabla 6.1 se presentan los resultados de la validación cruzada realizada con el conjunto de datos de entrenamiento para la detección de estereotipos. Se observa que el modelo de Regresión Logística logró una alta precisión, pero con un balance bajo en recall, lo que sugiere que, aunque este modelo es bueno para identificar correctamente los estereotipos cuando están presentes, pierde una cantidad significativa de instancias positivas. El modelo SVM mostró un mejor equilibrio entre precisión y recall, logrando una mayor F1, lo que indica una mejora en la capacidad del modelo para detectar estereotipos sin perder tantas instancias positivas en comparación con la Regresión Logística. Por otro lado, el modelo GPT-3.5 Turbo usando *zero-shot learning*, mostró un desempeño intermedio y, aunque tiene una precisión relativamente baja, es muy efectivo en recuperar la mayoría de las instancias positivas, lo que sugiere una capacidad notable para identificar estereotipos en contextos variados. Además, los modelos basados en transformadores específicos para el español, BETO y RoBERTa, presentaron resultados mixtos. BETO obtuvo una alta precisión, pero su recall fue significativamente bajo, lo que indica que, aunque es muy preciso, tiende a perder muchas instancias positivas. En contraste, RoBERTa mostró un mejor balance entre precisión y recall, con una F1 de 0,67, destacándose como el modelo más equilibrado en términos de identificación de estereotipos.

6.1.2. Evaluación test dataset: Detección de estereotipos

Modelo	Métrica	Valor
Regresión Logística	F1	**0,42**
	Precisión	0,80
	Recall	0,28
SVM	F1	**0,44**
	Precisión	0,77
	Recall	0,31
GPT-3.5 Turbo (*zero-shot learning*)	F1	**0,54**
	Precisión	0,51
	Recall	0,59
GPT-3.5 Turbo (1 ejemplo) (*few-shot learning*)	F1	**0,58**
	Precisión	0,55
	Recall	0,61
GPT-3.5 Turbo (3 ejemplos) (*few-shot learning*)	F1	**0,63**
	Precisión	0,58
	Recall	0,69
GPT-3.5 Turbo (5 ejemplos) (*few-shot learning*)	F1	**0,58**
	Precisión	0,64
	Recall	0,54
BETO	F1	**0,58**
	Precisión	0,80
	Recall	0,46
RoBERTa	F1	**0,62**
	Precisión	0,73
	Recall	0,54

Cuadro 6.2: Resultados de las métricas obtenidas en la evaluación con el dataset de test para la tarea 1: Detección de estereotipos.

En la evaluación con el conjunto de datos de test, se observó que los modelos basados en transformadores, en particular RoBERTa y BETO, mantuvieron su alto rendimiento, superando a los modelos tradicionales en términos de F1-score. RoBERTa, en particular, alcanzó un F1 de 0,62, mientras que BETO obtuvo un F1 de 0,58, lo que indica que estos modelos son más robustos y generalizan mejor cuando se aplican a datos no vistos previamente. Además, con un F1 de 0,54 en su configuración de *zero-shot learning*, GPT-3.5 Turbo no solo superó a la Regresión Logística y SVM, sino que también demostró una capacidad significativa para mantener un equilibrio entre precisión y recall. Esto resalta que, aunque en este caso GPT-3.5 Turbo no alcanzó la máxima precisión observada en RoBERTa o BETO, sus resultados son notablemente mejores que los de los enfoques tradicionales, subrayando su potencial como un modelo competitivo en esta tarea. Sin embargo, al introducir *few-shot learning*, el rendimiento de GPT-3.5 Turbo mejoró notablemente, alcanzando un F1 de 0,63 cuando se utilizaron tres ejemplos. Esto demuestra que el modelo GPT-3.5 Turbo puede beneficiarse considerablemente de la exposición a ejemplos adicionales, mejorando su capacidad para identificar estereotipos con mayor precisión y recall.

6.1.3. Resultados en la tarea 1: DESTESTS-Dis

Rank	Run	F1
0	BASELINE_gold_standard	1.000
1	Brigada Lenguaje_1	0.724
2	I2C-Huelva_1	0.712
3	I2C-Huelva_2	0.701
4	EUA_2	0.691
5	EUA_3	0.685
6	BASELINE_beto	0.663
7	EUA_1	0.653
8	UC3M-SAS_2	0.641
9	TaiDepZai999_UIT_AIC_1	0.630
10	TaiDepZai999_UIT_AIC_3	0.624
11	TaiDepZai999_UIT_AIC_2	0.608
12	UC3M-SAS_1	0.594
13	BASELINE_all_ones	0.589
14	VINE Bias Busters_1	0.581
15	VINE Bias Busters_2	0.558
16	VINE Bias Busters_3	0.514
17	I2C-Huelva_3	0.375
18	BASELINE_tfidf_svc	0.297
19	BASELINE_random_classifier	0.297
20	BASELINE_fast_text_svc	0.297
21	BASELINE_all_zeros	0.000

Cuadro 6.3: Task 1 with Hard Labels.

En la tarea 1 de detección de estereotipos con hard labels de la competición DETESTS-Dis [1], los resultados obtenidos por el equipo de este proyecto, VINE Bias Busters, se situaron en las posiciones 14, 15 y 16 del ranking. Específicamente, el run1 (BETO) ocupó la posición 14 con un F1-score de 0,581, el run2 (RoBERTa) quedó en la posición 15 con un F1-score de 0,558, y el run3 (GPT-3.5 Turbo) se ubicó en la posición 16 con un F1-score de 0,514. Estos resultados muestran un rendimiento moderado en comparación con los equipos mejor posicionados, como Brigada Lenguaje_1 que alcanzó un F1 de 0,724 en la primera posición. Otro punto a comentar es que, se observa que el F1 Score en el ranking para RoBERTa difiere del obtenido finalmente en este proyecto debido a un error a la hora de escribir el código para el modelo, el cual se encuentra explicado en el capítulo anterior, en concreto en la sección 5.2 Problemas encontrados.

Además, en el ranking se puede ver que otros modelos basados en transformadores, como el baseline de BETO y RoBERTa, quedaron también entre los primeros puestos, con un F1-score de 0,663 y 0,558, respectivamente. Esto sugiere que, aunque nuestros modelos mostraron un rendimiento sólido, todavía hay margen para mejorar y optimizar su capacidad de detección de estereotipos en futuras iteraciones.

Por otro lado, si se consideran los resultados obtenidos posteriormente con *few-shot learning* utilizando el modelo GPT-3.5 Turbo, se observa un rendimiento notablemente

[1] https://detests-dis.github.io/evaluation/

mejorado. Con la inclusión de 3 ejemplos de muestra, el modelo alcanzó un F1-score de 0,63, lo que lo situaría en la misma posición que el noveno lugar en el ranking oficial. Esto representa un aumento de unas 5 posiciones en comparación con el mejor modelo del equipo (BETO), demostrando que la técnica de *few-shot learning* puede mejorar significativamente la capacidad de detección de estereotipos.

6.1.4. Evaluación test dataset: Clasificación de implicitud

Modelo	Métrica	Valor
Regresión Logística	ICM	**-0,37**
	ICM Norm	0,36
	Precision Implicit	0,58
	Precision Explicit	0,51
	Recall Implicit	0,02
	Recall Explicit	0,44
SVM	ICM	**-0,37**
	ICM Norm	0,36
	Precision Implicit	0,58
	Precision Explicit	0,51
	Recall Implicit	0,01
	Recall Explicit	0,47
GPT-3.5 Turbo (*zero-shot learning*)	ICM	**-1,70**
	ICM Norm	0,0
	Precision Implicit	0,26
	Precision Explicit	0,10
	Recall Implicit	0,41
	Recall Explicit	0,38
GPT-3.5 Turbo (1 ejemplo) (*few-shot learning*)	ICM	**-1,69**
	ICM Norm	0
	Precision Implicit	0,29
	Precision Explicit	0,13
	Recall Implicit	0,31
	Recall Explicit	0,61
GPT-3.5 Turbo (3 ejemplos) (*few-shot learning*)	ICM	**-0,60**
	ICM Norm	0
	Precision Implicit	0,32
	Precision Explicit	0.15
	Recall Implicit	0,38
	Recall Explicit	0,67
GPT-3.5 Turbo (5 ejemplos) (*few-shot learning*)	ICM	**-1,73**
	ICM Norm	0
	Precision Implicit	0,33
	Precision Explicit	0,12
	Recall Implicit	0,31
	Recall Explicit	0,59
BETO	ICM	**-0.07**
	ICM Norm	0.47
	Precision Implicit	0.64
	Precision Explicit	0.48
	Recall Implicit	0.17
	Recall Explicit	0.64
RoBERTa	ICM	**-0.22**
	ICM Norm	0.41
	Precision Implicit	None
	Precision Explicit	0.37
	Recall Implicit	0.0
	Recall Explicit	0.73

Cuadro 6.4: Resultados de las métricas obtenidas en la evaluación con el dataset de test para la tarea 2: Clasificación de implicitud.

En la tarea de clasificación de implicitud, los modelos basados en transformadores nuevamente superaron a los modelos tradicionales en varias métricas clave. BETO y RoBERTa lograron los mejores resultados en lo que se refiere a la métrica ICM, que considera tanto la estructura jerárquica como la especificidad de la clase. BETO, en particular, se destacó con un ICM de -0,07 y un ICM Norm de 0,47, lo que lo posiciona como uno de los mejores modelos para esta tarea. Además, se observa una mejora significativa en el rendimiento de GPT-3.5 Turbo al aplicar *few-shot learning*. Con la inclusión de 3 ejemplos, el modelo alcanzó un ICM de -0,60, demostrando una mayor capacidad para captar la naturaleza implícita de los estereotipos. Aunque estos resultados no superan a BETO y RoBERTa, muestran que el uso de ejemplos puede aumentar la eficacia de GPT-3.5 Turbo en esta tarea.

Es importante aclarar que, aunque no se participó originalmente en la tarea 2, se decidió realizar la tarea a posteriori para evaluar el rendimiento de los modelos en este contexto. A pesar de no figurar en el ranking oficial, a continuación se muestra la tabla oficial con el ranking de los resultados para la tarea 2 de la competición DETESTS-Dis de manera que se puedan comparar los resultados de este trabajo con los del ranking para obtener una referencia sobre la eficacia de los modelos.

Rank	Run	ICM	ICM Norm
0	BASELINE_gold_standard	1.380	1.000
1	BASELINE_beto	0.126	0.546
2	EUA_2	0.065	0.524
3	EUA_3	0.061	0.522
4	EUA_1	0.045	0.516
5	Brigada Lenguaje_1	-0.240	0.413
6	BASELINE_tfidf_svc	-0.275	0.400
7	I2C-Huelva_1	-0.328	0.381
8	BASELINE_fast_text_svc	-0.470	0.328
9	BASELINE_all_zeros	-0.797	0.211
10	BASELINE_random_classifier	-1.056	0.117
11	BASELINE_all_ones	-1.210	0.060
12	I2C-Huelva_3	-1.263	0.042
13	I2C-Huelva_2	-1.403	0.000
14	UC3M-SAS_2	-2.103	0.000

Cuadro 6.5: Ranking de la Tarea 2 con Hard Labels.

En esta comparación, se evidencia que, aunque los modelos de este proyecto obtuvieron resultados competitivos, especialmente BETO, con un ICM de -0,07, todavía hay margen de mejora en comparación con los modelos mejor posicionados en el ranking oficial, como el BASELINE_gold_standard que obtuvo un ICM de 1,380. Esto sugiere que, aunque mis modelos son sólidos, su capacidad para manejar la clasificación jerárquica de estereotipos implícitos y explícitos podría beneficiarse de ajustes adicionales.

Por otro lado, el modelo GPT-3.5 Turbo tuvo dificultades significativas en esta tarea, obteniendo el ICM más bajo (-1,70) entre los modelos evaluados, lo que destaca las limitaciones de este modelo en comparación con los modelos entrenados específicamente en español. Sin embargo, si comparamos los resultados obtenidos posteriormente con *few-shot learning* utilizando 3 ejemplos, el modelo GPT-3.5 Turbo habría alcanzado un ICM del -0,60, lo que lo situaría en la novena posición del ranking. Este resultado subraya la

eficacia de *few-shot learning* para mejorar la capacidad de detección de estereotipos implícitos, posicionando a GPT-3.5 Turbo de manera más competitiva en la tarea.

6.2 Análisis de modelos

En esta sección se realiza una comparación y análisis de los resultados obtenidos por los distintos modelos en las diferentes clasificaciones. En el análisis se muestran las diferencias en el rendimiento de los modelos de aprendizaje automático tradicionales y los basados en transformadores, así como entre los modelos preentrenados en español y el modelo generativo GPT-3.5 Turbo.

Además, se llevará a cabo un análisis detallado de las matrices de confusión generadas para las predicciones tanto de estereotipos como de implicitud. La matriz de confusión muestra las siguientes métricas:

- **Verdaderos Positivos (TP, por su acrónimo en inglés):** Representa las instancias correctamente predichas como positivas.
- **Falsos Positivos (FP, por su acrónimo en inglés):** Indica las instancias incorrectamente predichas como positivas cuando en realidad son negativas.
- **Falsos Negativos (FN, por su acrónimo en inglés):** Representa las instancias incorrectamente predichas como negativas cuando en realidad son positivas.
- **Verdaderos Negativos (TN, por su acrónimo en inglés):** Indica las instancias correctamente predichas como negativas.

Estas métricas proporcionan una visión detallada del rendimiento del modelo y es importante destacar que, en este caso, las matrices de confusión están normalizadas. Esto significa que cada valor dentro de la matriz representa una proporción en lugar de un conteo absoluto, es decir, la normalización se realiza dividiendo cada fila por su suma total, lo que convierte los recuentos de la matriz en porcentajes relativos. Esto facilita la comparación entre modelos y revela patrones de predicción.

El análisis de estas matrices permitirá identificar las fortalezas y debilidades de cada modelo en términos de cómo manejan las clasificaciones de estereotipos e implicitud, proporcionando información crucial para la mejora y optimización del rendimiento del modelo en contextos específicos.

6.2.1. Análisis de matrices de confusión: Clasificación de estereotipos

A continuación se presentan las matrices de confusión de los modelos evaluados para la tarea de detección de estereotipos. Las matrices permiten visualizar cómo se distribuyen las predicciones correctas e incorrectas de cada modelo. En esta clasificación la clase positiva es la que contiene estereotipos y la clase negativa la que no.

Modelo SVM

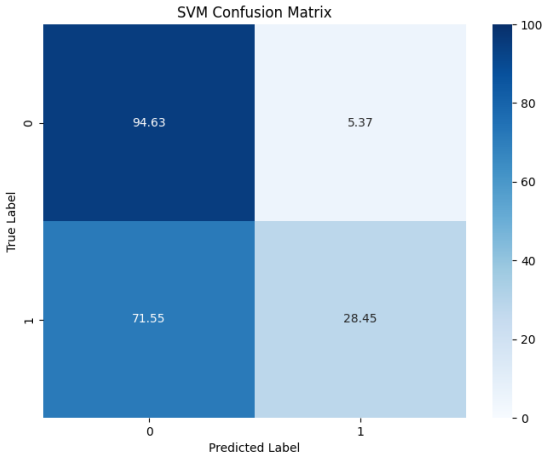

Figura 6.1: Matriz de confusión del modelo SVM

La matriz de confusión del modelo SVM muestra que tiene un alto porcentaje para la clase negativa (0), identificando correctamente el 94,63 % de los comentarios que no contienen estereotipos. Sin embargo, su capacidad para detectar estereotipos (clase positiva o 1) es limitada, con solo un 28,45 % de aciertos, indicando que falla en identificar una gran cantidad de comentarios con estereotipos.

Modelo RoBERTa

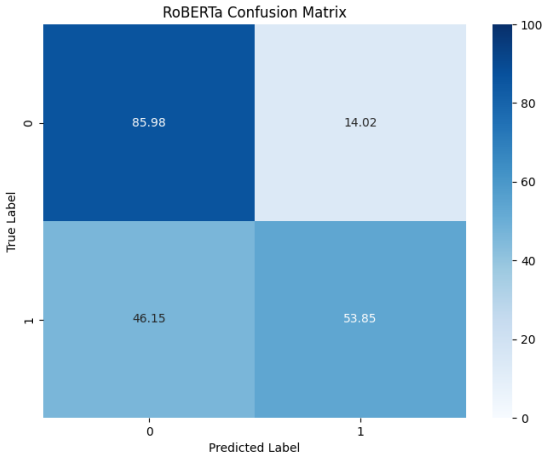

Figura 6.2: Matriz de confusión del modelo RoBERTa

El modelo RoBERTa muestra un mejor equilibrio entre los TN y los TP. La precisión para la clase negativa (0) es del 85,98 %, mientras que para la clase positiva (1) es del 53,85 %. Esto sugiere que RoBERTa tiene una mejor capacidad para identificar correctamente los comentarios que contienen estereotipos.

Modelo Regresión Logística

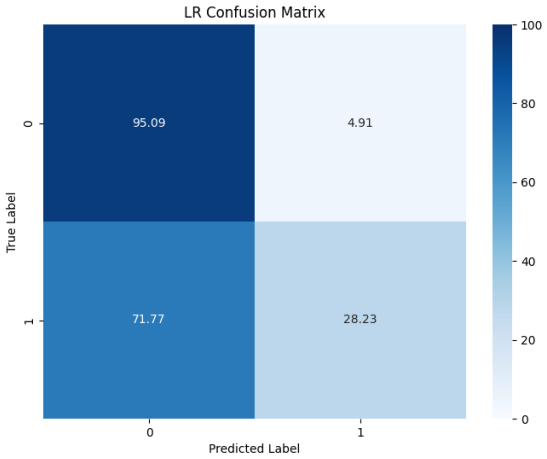

Figura 6.3: Matriz de confusión del modelo Regresión Logística

La matriz de confusión del modelo de Regresión Logística es similar a la de SVM, con una alta tasa de aciertos para la clase negativa (95,09 %), pero una baja capacidad para identificar estereotipos (28,23 %).

Modelo BETO

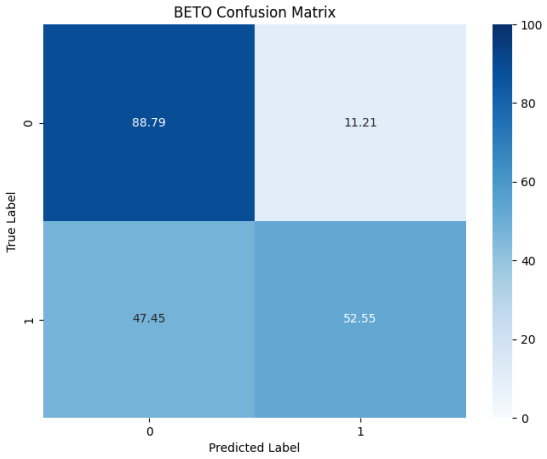

Figura 6.4: Matriz de confusión del modelo BETO

El modelo BETO también muestra un buen equilibrio, con una precisión del 88,79 % para la clase negativa y un 52,55 % para la clase positiva. Además, tiene una tasa baja en cuanto a los *FP* (11,21 %).

Modelo GPT 3.5 Turbo (*zero-shot learning*)

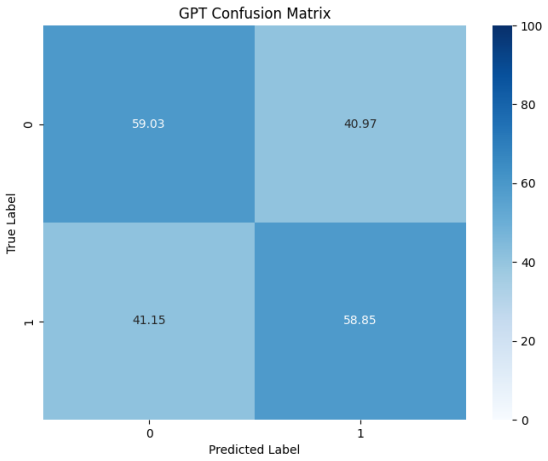

Figura 6.5: Matriz de confusión del modelo GPT-3.5 Turbo (*zero-shot learning*)

El modelo GPT-3.5 Turbo muestra un balance tanto en la clase negativa como la clase positiva, con un 59,03 % de precisión en ambos casos. Esto indica que el modelo tiene una capacidad equilibrada para predecir ambas clases, aunque aún hay una tasa significativa de errores (alrededor del 41 %) en ambos tipos de predicciones.

Modelo GPT 3.5 Turbo (*few-shot learning*)

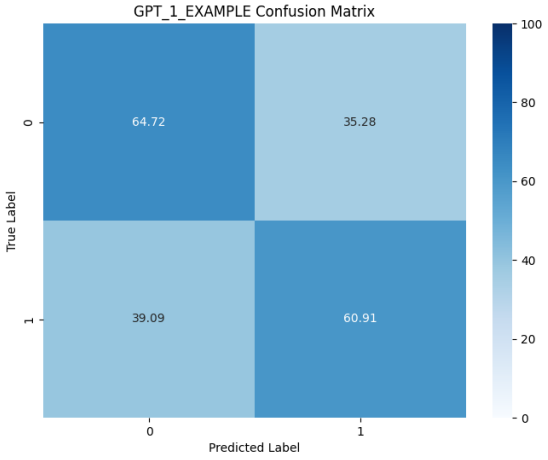

Figura 6.6: Matriz de confusión del modelo GPT-3.5 Turbo (*few-shot learning*) con 1 ejemplo

El modelo GPT-3.5 Turbo, utilizando *few-shot learning* con un ejemplo de texto que contiene estereotipos y otro que no, mejora su capacidad de detección en comparación con *zero-shot learning*, alcanzando una precisión del 64,72 % en la clase negativa y del 60,91 % en la clase positiva. Esto indica que la inclusión de un solo ejemplo puede reducir significativamente los errores de clasificación.

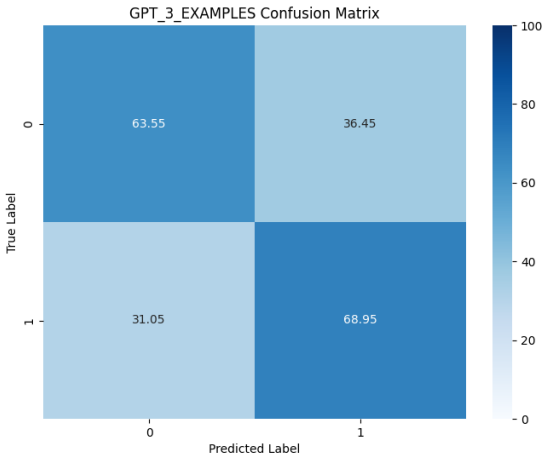

Figura 6.7: Matriz de confusión del modelo GPT-3.5 Turbo (*few-shot learning*) con 3 ejemplos

El modelo GPT-3.5 Turbo, utilizando *few-shot learning* con 3 ejemplos, muestra una mejora en su capacidad de detección de estereotipos en comparación con el uso de un solo ejemplo, logrando una precisión del 63,55 % en la clase negativa y del 68,95 % en la clase positiva. Esto demuestra que aumentar el número de ejemplos proporcionados permite al modelo capturar mejor las características de los textos estereotipados, reduciendo significativamente los errores de clasificación, en especial para la clase positiva.

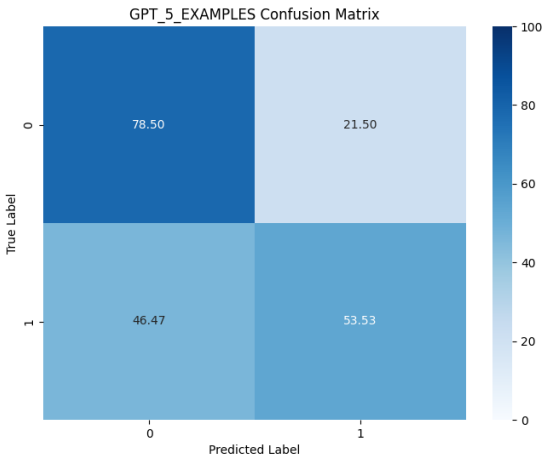

Figura 6.8: Matriz de confusión del modelo GPT-3.5 Turbo (*few-shot learning*) con 5 ejemplos

El modelo GPT-3.5 Turbo, utilizando *few-shot learning* con 5 ejemplos, muestra una precisión del 78,50 % en la clase negativa y del 53,53 % en la clase positiva. Aunque la precisión en la clase negativa es alta, se observa una disminución en la precisión de la clase positiva en comparación con el uso de menos ejemplos.

6.2.2. Análisis de matrices de confusión: Clasificación de implicitud

Para la tarea de clasificación de implicitud, se presentan las matrices de confusión que permiten evaluar cómo los modelos diferenciaron entre estereotipos implícitos y explíci-

tos. En esta clasificación la clase positiva es la que contiene estereotipos implícitos y la clase negativa la que contiene estereotipos explícitos.

Modelo SVM

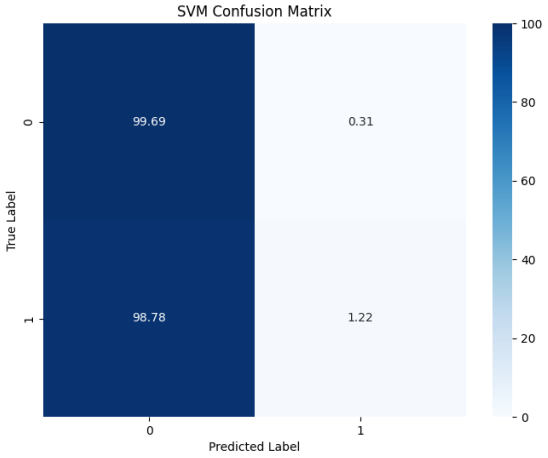

Figura 6.9: Matriz de confusión del modelo SVM para la clasificación de implicitud

La matriz de confusión del modelo SVM para la clasificación de implicitud muestra una precisión del 99,69 % para la clase negativa (0), pero una baja capacidad para identificar la clase positiva (1,22 %). Esto indica que el modelo tiene muchas dificultades para distinguir los estereotipos implícitos.

Modelo RoBERTa

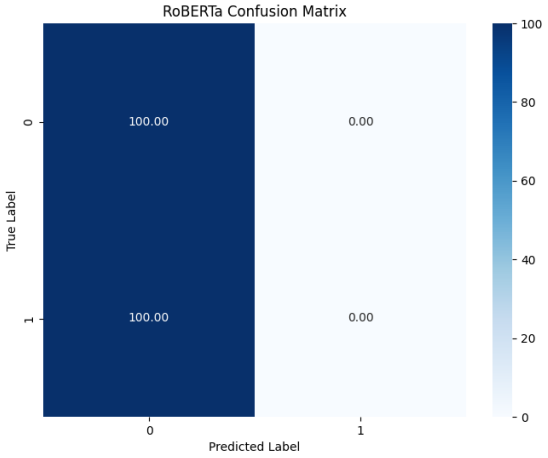

Figura 6.10: Matriz de confusión del modelo RoBERTa para la clasificación de implicitud

La matriz de confusión de RoBERTa muestra que el modelo clasifica incorrectamente todos los comentarios con estereotipos implícitos, con una precisión del 100 % para la clase negativa y del 0 % para la clase positiva, indicando un sesgo significativo y limitaciones en la generalización.

Modelo Regresión Logística

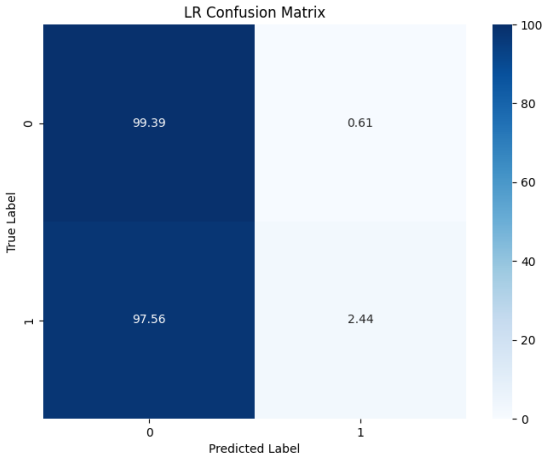

Figura 6.11: Matriz de confusión del modelo Regresión Logística para la clasificación de implicitud

La matriz de confusión de Regresión Logística, al igual que RoBERTa, muestra alta precisión para la clase negativa (99,39 %), pero falla en identificar correctamente los comentarios con estereotipos implícitos, con solo un 2,44 % de aciertos para la clase positiva, indicando una muy baja capacidad de detección de estereotipos implícitos.

Modelo BETO

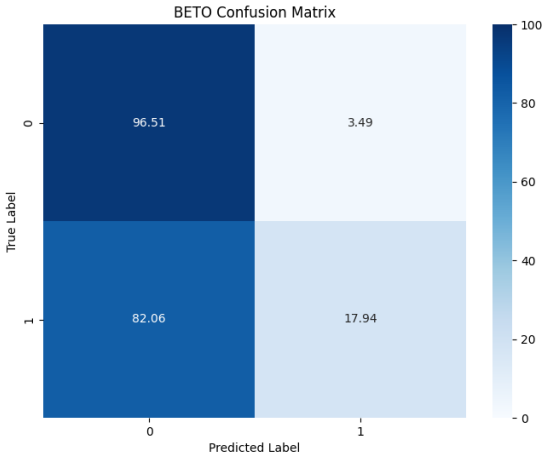

Figura 6.12: Matriz de confusión del modelo BETO para la clasificación de implicitud

El modelo BETO muestra una precisión del 96,51 % para la clase negativa y del 17,94 % para la clase positiva, pero con una alta tasa de *FN* (82,06 %), indicando altas dificultades para realizar la clasificación.

Modelo GPT 3.5 Turbo (*zero-shot learning*)

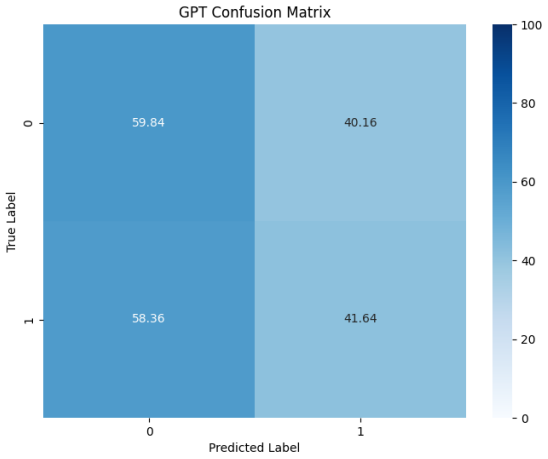

Figura 6.13: Matriz de confusión del modelo GPT-3.5 Turbo para la clasificación de implicitud

La matriz de confusión de GPT-3.5 Turbo muestra una precisión del 59,84 % para la clase negativa y del 41,64 % para la clase positiva, pero también evidencia una alta tasa de *FP* y *FN*, lo que indica dificultades para identificar correctamente los estereotipos implícitos y explícitos.

Modelo GPT 3.5 Turbo (*few-shot learning*)

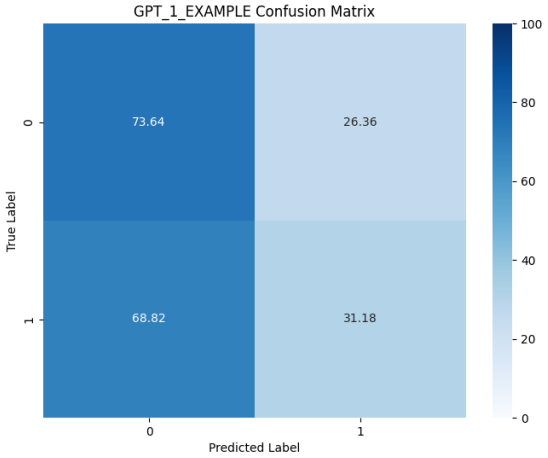

Figura 6.14: Matriz de confusión del modelo GPT-3.5 Turbo (*few-shot learning*) con 1 ejemplo para la clasificación de implicitud

El modelo GPT-3.5 Turbo, utilizando *few-shot learning* con 1 ejemplo para la clasificación de implicitud, muestra una precisión del 73,64 % en la clase negativa y del 31,18 % en la clase positiva. Aunque se observa una mejora en la identificación de la clase negativa en comparación con *zero-shot learning*, la precisión en la clase positiva sigue siendo baja, con un alto porcentaje de falsos negativos.

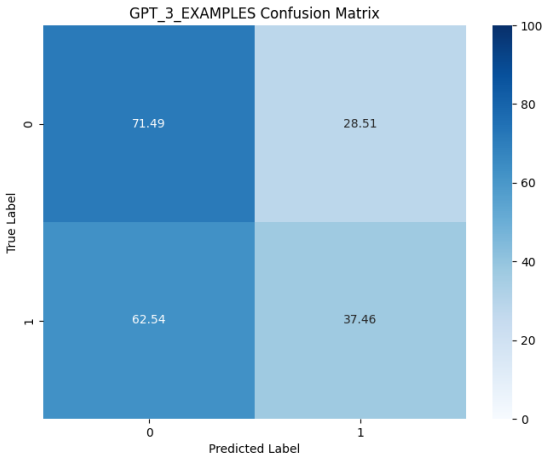

Figura 6.15: Matriz de confusión del modelo GPT-3.5 Turbo (*few-shot learning*) con 3 ejemplos para la clasificación de implicitud

El modelo GPT-3.5 Turbo, utilizando *few-shot learning* con 3 ejemplos para la clasificación de implicitud, logra una precisión del 71,49 % en la clase negativa y del 37,46 % en la clase positiva. Aunque hay una ligera reducción en la precisión de la clase negativa en comparación con el uso de un solo ejemplo, la precisión en la clase positiva muestra una mejora.

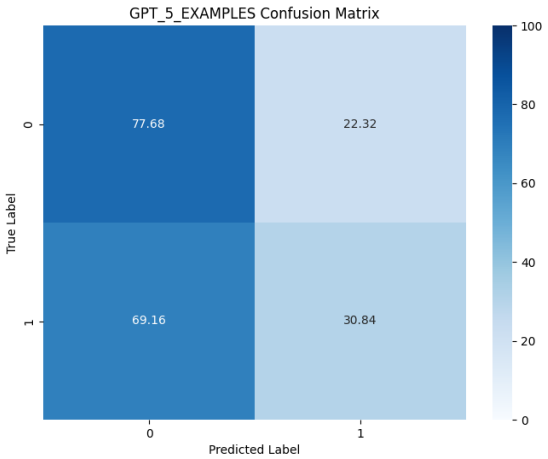

Figura 6.16: Matriz de confusión del modelo GPT-3.5 Turbo (*few-shot learning*) con 5 ejemplos para la clasificación de implicitud

Con 5 ejemplos en *few-shot learning*, el modelo GPT-3.5 Turbo logra una precisión del 77,68 % en la clase negativa, pero solo un 30,84 % en la clase positiva, por debajo de los resultados obtenidos con 0, 1 y 3 ejemplos.

6.2.3. Ejemplos de casos de clasificación erróneos

En el proceso de evaluación de los modelos, se han identificado varios ejemplos en los que los comentarios fueron clasificados incorrectamente por la mayoría de los mode-

los. Estos casos revelan ciertas limitaciones en los modelos al interpretar el contexto y la naturaleza del lenguaje en la clasificación de estereotipos, lo que subraya la complejidad de esta tarea.

"Estereotipos" clasificados incorrectamente como "No estereotipos"

ID del comentario	Comentario
s_4483	"No cabemos más ! Apago!"
d_3021_03	"En España bastante tenemos con la nuestra."

Cuadro 6.6: "Estereotipos" clasificados incorrectamente como "No estereotipos".

Estos comentarios fueron clasificados incorrectamente por todos los modelos presentados en este proyecto y la causa podría haber sido por la falta de información de contexto en los comentarios.

"No estereotipos" clasificados incorrectamente como "Estereotipos"

ID del comentario	Comentario
s_4554	"Para equilibrar en los países del islam debería de haber playas para nudistas, a ver cuando"
d_3042_01	"ahora vienen a defenderle el gorrino justiciero, la de las negritas y el otro fetichista de los delincuentes.... stuart?"

Cuadro 6.7: "No estereotipos" clasificados incorrectamente como "Estereotipos".

En este caso, las votaciones de los anotadores no fue unánime para ninguno de los dos comentarios, lo cual muestra la dificultad de estos comentarios y de identificar si contienen estereotipos. Así mismo, como en el caso anterior, todos los modelos clasificaron incorrectamente estos dos comentarios.

"Implícitos" clasificados incorrectamente como "Explícitos"

ID del comentario	Comentario
s_4356	"De allí igual no se escapan contagiados por el virus comunista chino"
s_4437	"De fuera vendrá quien de casa nos echará..."

Cuadro 6.8: "Implícitos" clasificados incorrectamente como "Explícitos".

Para estos comentarios, también hubo un error en la clasificación de la implicitud de los estereotipos por parte de todos los modelos exceptuando las 3 versiones de los modelos que usaron el enfoque *few-shot learning*.

"Explícitos" clasificados incorrectamente como "Implícitos"

ID del comentario	Comentario
s_4853	"Han venido para replicar sus costumbres en nuestro país y exigirnos respeto. Y nosotros, como buenos gilipollas, les concedemos eso ¿y más que nos pidiesen, porque... Pobrecitos e indefensos pagapensiones, ellos son así y hay que respetar sus costumbres."
d_2860_02	"No generan nada de valor, ni vienen a integrarse, ni respetan nuestras costumbres."

Cuadro 6.9: "Explícitos" clasificados incorrectamente como "Implícitos".

Finalmente, en estos ejemplos, el primer comentario fue clasificado incorrectamente por todos los modelos exceptuando el modelo RoBERTa (el cual no clasificó ningún comentario como "implícito") y las tres versiones del modelo GPT 3.5 Turbo con enfoque *few-shot learning*. El segundo comentario fue clasificado incorrectamente por todos los modelos exceptuando el modelo GPT con 3 ejemplos en *few-shot learning* y RoBERTa.

6.3 Comparación de modelos

El análisis de los diferentes modelos revela varias tendencias importantes. Los modelos tradicionales como Regresión Logística y SVM son efectivos para identificar textos sin estereotipos, logrando una alta precisión en la clase negativa. Sin embargo, estos modelos tienen una capacidad limitada para detectar textos con estereotipos, con una tasa baja de aciertos, especialmente en la clase positiva, lo que muestra una significativa incapacidad para identificar todos los estereotipos presentes.

Por otro lado, los modelos basados en transformadores, como RoBERTa y BETO, ofrecen un mejor equilibrio entre precisión y *recall*, destacándose en la detección de estereotipos con puntajes F1 superiores a los de los modelos tradicionales. Sin embargo, RoBERTa, a diferencia de BETO, no muestra capacidad para manejar la clasificación de estereotipos implícitos.

El modelo GPT-3.5 Turbo muestra un rendimiento más balanceado con una precisión moderada en ambas clases y una tasa de errores considerable en el escenario de *zero-shot learning*. Sin embargo, al utilizar *few-shot learning* con ejemplos adicionales, se observa una mejora significativa en su capacidad de detección de estereotipos. Con 3 ejemplos, el modelo alcanzó un F1-score de 0,63, superando a todos los modelos, tanto los tradicionales como los transformadores. A pesar de estas mejoras, GPT-3.5 Turbo todavía enfrenta desafíos en la reducción de falsos positivos y negativos, lo que sugiere que una selección y optimización cuidadosa de los ejemplos es crucial para mejorar su precisión y fiabilidad.

En términos generales, el transformador preentrenado en español, BETO, sigue demostrando un rendimiento superior en la clasificación de implicitud, reflejando una mayor robustez y capacidad de generalización en esta tarea.

CAPÍTULO 7
Conclusiones

El presente trabajo de fin de grado ha tenido como objetivo principal abordar la detección de estereotipos en textos en español utilizando métodos de aprendizaje automático y modelos de lenguaje masivos. A través de un enfoque exhaustivo, se han evaluado y comparado diversas técnicas, desde modelos tradicionales como SVM y Regresión Logística, hasta avanzados modelos de transformadores como BETO, RoBERTa, y GPT-3.5 Turbo. Los resultados obtenidos proporcionan una visión clara de la eficacia de cada enfoque y establecen una base sólida para futuras investigaciones en este área.

En primer lugar, los modelos tradicionales de aprendizaje automático, específicamente SVM y Regresión Logística, mostraron un rendimiento aceptable en la detección de estereotipos. Estos modelos se beneficiaron del preprocesamiento detallado que incluyó técnicas como la tokenización, eliminación de palabras vacías y lematización. Sin embargo, a pesar de su eficacia en la clasificación binaria de estereotipos, estos modelos demostraron limitaciones significativas al enfrentarse a la detección de estereotipos implícitos, donde la comprensión del contexto y las sutilezas del lenguaje juegan un papel crucial.

Por otro lado, los modelos de lenguaje masivo, en particular BETO, RoBERTa, y GPT-3.5 Turbo, mostraron una notable superioridad en la detección de estereotipos explícitos. Estos modelos, preentrenados en grandes volúmenes de datos y capaces de capturar relaciones contextuales complejas, ofrecieron una precisión y capacidad de generalización superiores. BETO, diseñado específicamente para manejar las peculiaridades del idioma español, logró un rendimiento excepcional en la identificación de estereotipos explícitos, destacándose en métricas como precisión, *recall* y F1-score. Sin embargo, RoBERTa fue menos efectivo en la detección de estereotipos implícitos. Por su parte, GPT-3.5 Turbo, utilizando *few-shot learning* con 3 ejemplos, superó a todos los modelos, incluidos BETO y RoBERTa, en la detección de estereotipos, lo que demuestra su capacidad de adaptarse rápidamente a nuevas tareas con un mínimo de datos de muestra. No obstante, en la clasificación de implicitud, BETO sigue mostrando un rendimiento superior.

Entre todos los modelos evaluados, BETO emergió como el más eficaz en la tarea de clasificación de estereotipos explícitos. Sin embargo, es importante señalar que, aunque BETO mostró un excelente rendimiento en la clasificación de estereotipos explícitos, la detección de estereotipos implícitos sigue siendo un desafío significativo para todos los modelos evaluados.

Para mejorar los modelos y abordar sus limitaciones actuales, se pueden considerar varias estrategias. En el caso de GPT-3.5 Turbo, la implementación de técnicas avanzadas de *few-shot learning* ha demostrado ser una vía prometedora, ya que al enseñarle ejemplos específicos, los resultados del modelo mejoran significativamente. La gran ventaja de GPT-3.5 Turbo es que no necesita una gran cantidad de datos para realizar la tarea con eficacia, lo cual es una ventaja en contextos donde los datos etiquetados son escasos y costosos de obtener. Esta característica lo hace especialmente útil para la detección de

estereotipos en escenarios donde la disponibilidad de datos etiquetados es limitada.

La solución presentada en este trabajo tiene como objetivo identificar qué modelos son más efectivos para la detección de estereotipos y así poder ser utilizados como herramientas de moderación de contenido online. Estos modelos pueden ser integrados en sistemas de moderación de plataformas de redes sociales, foros y sitios de noticias para identificar y filtrar automáticamente comentarios y publicaciones que contengan estereotipos, ayudando a mantener un entorno digital más seguro y respetuoso. Por ejemplo, los sistemas que utilizan estos modelos pueden marcar contenido sospechoso para revisión, bloquear automáticamente publicaciones ofensivas o enviar alertas a moderadores humanos para una intervención más rápida. Esto no solo protege a los usuarios de contenido dañino, sino que también promueve un discurso más inclusivo y equitativo.

Para mejorar aún más los resultados, se podría considerar un preprocesamiento de datos más exhaustivo, como la limpieza y normalización del texto, y en el caso de GPT-3.5 Turbo, seleccionar ejemplos de *few-shot learning* más representativos y diversos. Esto podría ayudar al modelo a captar mejor las sutilezas y variaciones en la expresión de estereotipos, tanto explícitos como implícitos. Además, la integración de enfoques de aprendizaje con desacuerdo (*learning with disagreement* [50]) podría ser explorada para mejorar la capacidad de los modelos para manejar la subjetividad y ambigüedad inherentes en la detección de estereotipos implícitos.

En conclusión, el presente trabajo ha demostrado que los modelos de lenguaje masivo, particularmente BETO, superan significativamente a los enfoques tradicionales en la detección de estereotipos explícitos en textos en español. Aunque GPT-3.5 Turbo con *few-shot learning* ha mostrado un gran potencial, BETO sigue siendo el modelo más robusto y efectivo en términos generales, considerando tanto la tarea de detección de estereotipos como la clasificación de implicitud. Sin embargo, la detección de estereotipos implícitos sigue siendo un área desafiante que requiere mejoras continuas y enfoques innovadores. Estas conclusiones no solo subrayan la importancia de utilizar tecnologías avanzadas de PLN para abordar problemas complejos de moderación de contenido, sino que también abren nuevas oportunidades para futuras investigaciones y mejoras en el campo. A medida que la tecnología continúa evolucionando, es esencial seguir explorando y perfeccionando estos modelos para crear un entorno digital más seguro, justo y equitativo.

Bibliografía

[1] Schmeisser W.S., Pastells P., Frenda S., Ariza A., Farrús M., Rosso P., Taulé M. (2024). "Overview of DETESTS-Dis at IberLEF 2024: DETEction and classification of racial STereotypes in Spanish - Learning with Disagreement" en *Procesamiento del Lenguaje Natural (SEPLN)*, num. 73.

[2] Catalá Martínez, R., Pérez, J. (2024). "Introducción a los Modelos de Lenguaje de Gran Tamaño (LLM)" en *PWC*, 23 de abril. https://www.pwc.es/es/newlaw-pulse/legaltech/introduccion-modelos-lenguaje-gran-tamano.html

[3] Martel, J. (2021). "Introducción al Procesamiento del Lenguaje Natural: de Turing a Bert, pasando por Watson" en *Intelligent*, 3 de febrero. https://itelligent.es/pln-turing-bert-watson/

[4] Nalda, V., (2020). "Machine Learning: Los orígenes y la evolución" en *FutureSpace*, 29 de septiembre. https://www.futurespace.es/machine-learning-los-origenes-y-la-evolucion/

[5] Fernández-Casal, R., Costa, J., Oviedo, M., "Capítulo 4 Máquinas de soporte vectorial" en *Aprendizaje Estadístico*. Universidad de Santiago de Compostela. Santiago de Compostela: Universidad de Santiago de Compostela. https://rubenfcasal.github.io/aprendizaje_estadistico/svm.html

[6] Universitat De València. *T. 9 – El modelo de regresión lineal*. http://ocw.uv.es/ciencias-de-la-salud/pruebas-1/1-3/t_09nuevo.pdf [Consulta: 25 de marzo 2024]

[7] Manning, C. D., Raghavan, P., Schütze, H. (2008). "Introduction to Information Retrieval" en *An Introduction to Information Retrieval*. Cambridge University Press. p. 155. https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

[8] Sasaki, Y. (2007). "The truth of the F-measure" en *University of Manchester, MIB-School of Computer Science*. https://people.cs.pitt.edu/~litman/courses/cs1671s20/F-measure-YS-26Oct07.pdf

[9] Amazon Web Services (AWS). *¿Qué son los transformadores en la inteligencia artificial?* https://aws.amazon.com/es/what-is/transformers-in-artificial-intelligence/ [Consulta: 30 de marzo 2024]

[10] Artem (2023). "Decoding Transformers' Superiority over RNNs in NLP Tasks" en *Hackernoon*, 28 de junio. https://hackernoon.com/decoding-transformers-superiority-over-rnns-in-nlp-tasks [Consulta: 2 de junio 2024]

[11] Malingan, N. (2024). "Understanding GPT-1, GPT-2, and GPT-3" en *Scaler*, 2 de mayo. https://www.scaler.com/topics/nlp/gpt-versions/ [Consulta: 2 de junio 2024]

[12] X. -S. Wei, H. -Y. Xu, Z. Yang, C. -L. Duan, Y. Peng. (2023). "Negatives Make a Positive: An Embarrassingly Simple Approach to Semi-Supervised Few-Shot Learning" en *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2024 (46), Issue: 4, pp.2091-2103. https://ieeexplore.ieee.org/document/10319790

[13] Efimov, V. (2024). "Large Language Models, GPT-3: Language Models are Few-Shot Learners" de *Towards Data Science*, 16 de febrero. https://towardsdatascience.com/large-language-models-gpt-3-language-models-are-few-shot-learners-6e1261a1b466 [Consulta: 2 de junio 2024]

[14] Peskine, Y., Korencic, D., Grubišic, I., Papotti, P., Troncy, R., Y Rosso, P. (2023). "Definitions Matter: Guiding GPT for Multi-label Classification" en *Findings of EMNLP-2023, Empirical Methods in Natural Language Processing*, pp. 4054–4063.

[15] Mnassri, K., Farahbakhsh, R., Crespi, M. (2024). "Multilingual Hate Speech Detection: A Semi-Supervised Generative Adversarial Approach" en *Entropy*, Vol. 2024 (26), pp. 344. https://doi.org/10.3390/e26040344

[16] Mnassri, K., Rajapaksha, P., Farahbakhsh, R., Crespi, N. (2023). "Hate Speech and Offensive Language Detection using an Emotion-aware Shared Encoder" en *IEEE International Conference on Communications (ICC)*, Roma, Italia, Francia. https://ieeexplore.ieee.org/document/10279690

[17] Chiu, K., Collins, A., Alexander, R. (2021). *Detecting Hate Speech with GPT-3* https://ar5iv.labs.arxiv.org/html/2103.12407

[18] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L. (2020). "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension" en *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880. https://aclanthology.org/2020.acl-main.703/

[19] Lan, Z., Chen, C., Goodman, S., Gimpel, K., Sharma, P., Soricut, R. (2020). "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations" en *International Conference on Learning Representations* https://arxiv.org/pdf/1909.11942

[20] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., Mccandlish, S., Radford, A., Sutskever, I., Amodei, D. (2020). "Language Models are Few-Shot Learners" en *ICLR Eighth International Conference On Learning Representations* https://arxiv.org/pdf/2005.14165

[21] Das, A., Nandy, S., Saha, R., Das, S., Saha, D. (2024). *Analysis and Detection of Multilingual Hate Speech Using Transformer Based Deep Learning*. https://arxiv.org/pdf/2401.11021v1 [Consulta: 30 de mayo de 2024]

[22] Chulvi B., Rosso P., Fernandez De La Hoz Zeitler K. (2024). "Monitoring Hate Speech against Immigrants in Social Media: A Taxonomy and a Guide to Detect it" en *Online Hate Speech. LabCom Books / Editorial IcesiLabCom Book*, pp. 193-220.

[23] Sánchez-Junquera J., Chulvi B., Rosso P., Ponzetto S. (2021). "How Do You Speak about Immigrants? Taxonomy and StereoImmigrants Dataset for Identifying Stereotypes about Immigrants." en *Applied Science*, 11(8), 3610. https://doi.org/10.3390/app11083610.

[24] Chulvi B., Molpeceres M., Rodrigo M., Toselli A., Rosso P. (2024). "Politicization of Immigration and Language Use in Political Elites: A Study of Spanish Parliamentary Speeches." en *Journal of Language and Social Psychology*, Volume 43, Issue 2, pp. 164 - 194. https://doi.org/10.1177/0261927X231175856

[25] Ariza-Casabona A., Schmeisser-Nieto1 W.S., Nofre M., Taulé M., Amigó E., Chulvi B., Rosso P. (2022). "Overview of DETESTS at IberLEF 2022: DETEction and classification of Racial STereotypes in Spanish." en *Procesamiento del Lenguaje Natural (SEPLN)*, num. 69, pp. 217-228,

[26] SEPLN *(Sociedad Española para el Procesamiento del Lenguaje Natural)* http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/6442 [Consulta: 10 de mayo de 2024]

[27] Glstyan, A., Cohen, P. R (2007). "Empirical comparison of "hard" and "soft" label propagation for relational classification" en *International Conference on Inductive Logic Programming*, p. 98–111, Springer, Berlin, Heidelberg.

[28] Frenda S., Patti V., Rosso P. (2023). "Killing me Softly: Creative and Cognitive Aspects of Implicitness in Abusive Language Online." en *Natural Language Engineering (JNLE)*, 29(6): 1516 - 1537

[29] Maina, H., Alonso Alemany, L., Ivetta, G., Rajngewerc, M., Busaniche, B., Benotti, L. (2024). "Exploring Stereotypes and Biases in Language Technologies in Latin America" en *Communications of the ACM*, 11 de julio. https://cacm.acm.org/latin-america-regional-special-section/exploring-stereotypes-and-biases-in-language-technologies-in-latin-america/ [Consulta: 3 de septiembre 2024]

[30] Hershcovich, D., Frank, S., Lent, H., De Lhoneux, M., Abdou, M., Brandl, S., Bugliarello, E., Cabello Piqueras, L., Chalkidis, I., Cui, R., Fierro, C., Margatina, K., Rust, P., Y Søgaard, A. (2022). "Challenges and Strategies in Cross-Cultural NLP" en *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, Vol. 2022 (1), pp.6997–7013. https://aclanthology.org/2022.acl-long.482/

[31] Amigó, E., Carrillo-De-Albornoz, J., Fernández, A., Gonzalo, J., Marco, G., Morante, R., Plaza, L., Pedrosa, J. (2024). "A Web Portal about the State of the Art of NLP Tasks in Spanish" en *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, p. 2028–2038, Torino, Italia. ELRA e ICCL. https://aclanthology.org/2024.lrec-main.183/

[32] C. Thompson, N., Greenewald, K., Lee, K., F. Manso, G. (2020). "The Computational Limits Of Deep Learning" en *MIT Initiative On The Digital Economy Research Brief*, Vol. 2020 (4) https://ide.mit.edu/wp-content/uploads/2020/09/RBN.Thompson.pdf

[33] Zhou, V. (2019). "A Simple Explanation of the Softmax Function" en *Victorzhou*, 22 de julio. https://victorzhou.com/blog/softmax/ [Consulta: 15 de agosto 2024]

[34] Abhishek, J., (2024). "All about Tokenization, Stop words, Stemming and Lemmatization in NLP", 2 de febrero. https://medium.com/@abhishekjainindore24/all-about-tokenization-stop-words-stemming-and-lemmatization-in-nlp-1620ffaf0f87 [Consulta: 20 de marzo 2024]

[35] Murel, J., Kavlakoglu, E. (2024). "What is bag of words?" en *IBM*, 19 de enero. https://www.ibm.com/topics/bag-of-words [Consulta: 12 de junio de 2024]

[36] Simha, A. (2021). "Understanding TF-IDF for Machine Learning" en *CapitalOne*, 6 de octubre. https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/ [Consulta: 12 de junio de 2024]

[37] Walder, L. (2018). "A Brief History of Word Embeddings" en *Gavagai*, 20 de septiembre. https://www.gavagai.io/text-analytics/a-brief-history-of-word-embeddings/ [Consulta: 12 de junio de 2024]

[38] Daniel (2022). "Word2vec : NLP y Word Embedding" en *DataScientest*, 17 de mayo. https://datascientest.com/es/nlp-word-embedding-word2vec-es [Consulta: 12 de junio de 2024]

[39] Pennington, J., Socher, R., D. Manning, C. (2014). "GloVe: Global Vectors for Word Representation" en *The Stanford NLP Group*, agosto. https://nlp.stanford.edu/projects/glove/#:~:text=GloVe%20is%20an%20unsupervised%20learning,Getting%20started%20(Code%20download) [Consulta: 12 de junio de 2024]

[40] Navarra, S. (2024). "¿Qué es FastText y cómo funciona?" en *Keepcoding*, 12 de abril. https://keepcoding.io/blog/que-es-fasttext-y-como-funciona/#:~:text=FastText%20es%20un%20modelo%20de,obtener%20representaciones%20vectoriales%20de%20palabras. [Consulta: 12 de junio de 2024]

[41] Wikipedia contributors. Support vector machine. *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Support_vector_machine. [Consulta: 1 de julio 2024]

[42] Wikipedia contributors. Logistic regression. *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Logistic_regression [Consulta: 1 de julio 2024]

[43] Garagiola, N. (2022). *Mejorando reconocimiento de entidades nombradas del Español mediante la especialización de BETO* https://rdu.unc.edu.ar/bitstream/handle/11086/28964/Garagiola%2C%20Nazareno.%20Mejorando%20reconocimiento%20de%20entidades%20nombradas%20del%20espa%C3%B1ol%20mediante%20la%20especializaci%C3%B3n%20de%20BETO.pdf?sequence=1&isAllowed=y [Consulta: 30 de julio de 2024]

[44] Contribuidores de GeeksForGeeks (2024). "Explicación del modelo BERT – PNL" en *GeeksForGeeks*. https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/ [Consulta: 3 de agosto de 2024]

[45] Cristina, S. (2023). "The Transformer Attention Mechanism" en *Machine Learning Mastery*, 6 de enero. https://machinelearningmastery.com/the-transformer-attention-mechanism/ [Consulta: 1 de julio 2024]

[46] Raschka, S. (2023). "Understanding Encoder And Decoder LLMs" en *Ahead of AI*, 17 de junio. https://magazine.sebastianraschka.com/p/understanding-encoder-and-decoder [Consulta: 20 de agosto de 2024]

[47] Petkova, G. (2021). "The Gold Standard – The Key to Information Extraction and Data Quality Control" en *Ontotext*, 27 de mayo. https://www.ontotext.com/blog/gold-standard-key-to-information-extration-data-quality-control/ [Consulta: 30 de julio de 2024]

[48] Amigó, E., D. Delgado, A. (2022). "Evaluating Extreme Hierarchical Multi-label Classification" en *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, Vol. 2022 (1: Long Papers), p. 5809–5819, Dublin, Ireland. https://aclanthology.org/2022.acl-long.399/

[49] Sotaquirá, M. (2023). *Validación cruzada y k-fold cross-validation* en *CodificandoBits*, 31 de julio. https://www.codificandobits.com/blog/validacion-cruzada-k-fold-cross-validation/ [Consulta: 20 de abril 2024]

[50] N. Uma, A., Fornaciari, T., Hovy, D., Paun, S., Plank, B., Poesio, M. (2021). "Learning from Disagreement: A Survey" en *JAIR Journal of Artificial Intelligence Research*, Vol. 72 (2021). https://www.jair.org/index.php/jair/article/view/12752

=1

APÉNDICE A

Detección de estereotipos usando GPT Few-shot learning

En el apartado 4.6.3 se menciona que para el modelo GPT 3.5 Turbo se han experimentado diferentes enfoques para abordar la tarea de detección de estereotipos. El enfoque que usa *few-shot learning* permite al modelo aprender y adaptarse a la tarea proporcionando unos pocos ejemplos de textos con etiqueta indicando si contienen estereotipos o si no. De esta manera se le ayuda al modelo a comprender mejor la tarea de clasificación.

El proceso se divide en dos partes principales: definir ejemplos de *few-shot* y aplicar al *prompt* estos ejemplos para predecir si un texto contiene estereotipos o no. A continuación se presenta el código utilizado para este fin:

```
few_shot_examples = [
    {"role": "system", "content": "You are an AI trained to identify stereotypes in
    Spanish social media texts answering only with '0' if it does not contain any
    stereotypes and '1' if it does."},

    {"role": "user", "content": "Does the following text contain any stereotypes?
    \n\n'Desgraciadamente, y visto lo visto, al final vamos a tener que aplicar lo que
    hace los israelíes en sus fronteras.'"},

    {"role": "assistant", "content": "1"},  # La respuesta esperada es '1' porque
    contiene estereotipos

    {"role": "user", "content": "Does the following text contain any stereotypes?\n\n
    'Hay que enseñar que la magia no existe.'"},

    {"role": "assistant", "content": "0"}  # La respuesta esperada es '0' porque no
    contiene estereotipos
]

for idx, row in df.iterrows():
    text_c= row['text']

    response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages = few_shot_examples + [
            {"role": "user", "content": f"Does the following text contain any
            stereotypes?\n\n'{text_c}'"}
        ]
    )

    resultado = response.choices[0].message.content
    df.at[idx, 'stereotype'] = resultado
```

En el código mostrado:

- Se definen ejemplos de *few-shot* que proporcionan contexto al modelo sobre cómo clasificar los textos. El primer ejemplo contiene estereotipos y se espera una respuesta '1', mientras que el segundo no contiene estereotipos y se espera una respuesta '0'.
- Cada comentario cargado en el *Data Frame (df)* es analizado por GPT junto con los ejemplos proporcionados.
- Las respuestas del modelo son los resultados de la clasificación y son almacenados en el *DataFrame* en la columna 'stereotype', indicando con '0' o '1' si el texto contiene estereotipos.

APÉNDICE B
Objetivos de desarrollo sostenible (ODS)

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**		X		
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**		X		
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**				X
ODS 10. **Reducción de las desigualdades.**	X			
ODS 11. **Ciudades y comunidades sostenibles.**				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**	X			
ODS 17. **Alianzas para lograr objetivos.**				X

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Este trabajo de fin de grado está estrechamente relacionado con varios Objetivos de Desarrollo Sostenible (ODS) de la ONU, en particular con los ODS 3, 5, 10 y 16. A continuación, se detalla cómo el proyecto contribuye a cada uno de estos objetivos.

ODS 3: Salud y bienestar

La salud y el bienestar no se limitan únicamente a la ausencia de enfermedades físicas, sino que también abarcan la salud mental y el bienestar psicológico. En el entorno digital actual, los estereotipos y los discursos de odio en las redes sociales representan una gran amenaza para la salud mental de las personas. La exposición constante a comentarios negativos y discriminatorios puede conducir a problemas como ansiedad, depresión y baja autoestima, especialmente entre grupos vulnerables como minorías étnicas, personas LGBTQ+, y otros colectivos marginados.

Este trabajo aborda esta problemática mediante el desarrollo y evaluación de modelos de aprendizaje automático para la detección de estereotipos en comentarios online. Al identificar y mitigar estos comentarios dañinos, el proyecto contribuye a crear un entorno digital más saludable. Los beneficios para la salud mental pueden ser significativos, ya que la reducción de la exposición a contenidos negativos puede ayudar a disminuir los problemas psicológicos entre los usuarios de redes sociales.

Además, un entorno libre de discursos de odio y estereotipos es fundamental para que las personas puedan expresar sus opiniones y compartir información sin temor a represalias. El proyecto contribuye a este objetivo mediante la implementación de tecnologías avanzadas que pueden identificar y filtrar contenidos tóxicos, promoviendo así un espacio digital más seguro y acogedor.

ODS 5: Igualdad de género

La desigualdad de género es una problemática que afecta de manera particularmente aguda a las mujeres inmigrantes, quienes no solo enfrentan los desafíos de ser migrantes, sino también la discriminación de género. Estos estigmas perpetúan la desigualdad, limitando sus oportunidades y contribuyendo a un entorno hostil, tanto en la vida real como en el entorno digital.

En muchas plataformas online, los comentarios sexistas y xenófobos se dirigen con frecuencia a mujeres inmigrantes, reforzando tanto los roles de género tradicionales como los prejuicios hacia la inmigración. Esto no solo afecta su participación en las conversaciones digitales, sino que también refuerza las barreras que enfrentan en su integración y

desarrollo en la sociedad de acogida.

El entrenamiento de modelos para detectar estereotipos ayuda a identificar y mitigar estos comentarios sexistas en las redes sociales. Al detectar y eliminar estos estereotipos, se promueve un discurso más igualitario y respetuoso, que es esencial para avanzar hacia la igualdad de género en todos los contextos, incluyendo a las mujeres inmigrantes.

Además, al crear un entorno online libre de estereotipos de género y xenofobia, se fomenta la participación activa de las mujeres inmigrantes en el espacio digital, sin temor a ser objeto de discriminación o acoso. Este proyecto contribuye a este objetivo al investigar y desarrollar tecnologías que protejan mejor a estos grupos vulnerables de comentarios dañinos, promoviendo así una participación más equitativa, inclusiva y diversa en el entorno digital.

ODS 10: Reducción de las desigualdades

La desigualdad social a menudo aumenta debido a los prejuicios que se comparten en este tipo de plataformas. Estos prejuicios pueden ser por razón de ingresos, sexo, edad, discapacidad, orientación sexual, clase, etnia, religión, o raza como es el caso que se estudia en este proyecto. Estos prejuicios pueden reforzar las barreras existentes y crear nuevas formas de discriminación.

Al identificar y tratar este tipo de comentarios, el proyecto busca reducir las desigualdades promoviendo una representación más justa y equitativa de todos los grupos sociales en el entorno digital. Esto es fundamental para combatir la discriminación sistémica y fomentar la inclusión social.

Al eliminar los estereotipos y promover un discurso inclusivo en las redes sociales, el TFG contribuye a crear un entorno más acogedor para todos. Este enfoque no solo ayuda a reducir las desigualdades existentes, sino que también fomenta una mayor comprensión y aceptación de la diversidad.

ODS 16: Paz, justicia e instituciones sólidas

La paz y la justicia no son solo objetivos a nivel estatal o internacional, también son esenciales en el ámbito digital. Los discursos de odio en las redes sociales pueden fomentar la división y la violencia, creando un entorno hostil y polarizado. El TFG contribuye a la paz digital al desarrollar herramientas que detectan y mitigan estos discursos perjudiciales.

Al promover un discurso más respetuoso y equitativo, el proyecto ayuda a construir un entorno digital más justo. Esto es especialmente importante en un mundo cada vez más interconectado, donde las interacciones online pueden tener un impacto significativo en

la cohesión social y la estabilidad.

Las instituciones digitales, incluidas las plataformas de redes sociales, tienen la responsabilidad de garantizar que sus entornos sean seguros y justos para todos los usuarios. El trabajo sigue este objetivo al proporcionar tecnologías avanzadas que pueden ser implementadas por estas plataformas para monitorear y gestionar los contenidos generados por los usuarios.

De la misma manera, el respeto y la protección de los derechos humanos en el entorno digital son esenciales para una mantener la paz. Los estereotipos representan una violación de estos derechos, ya que perpetúan la discriminación y el prejuicio.

Un entorno digital libre de odio es fundamental para fomentar la participación cívica y el diálogo constructivo. Al eliminar las barreras que impiden la participación de ciertos grupos, todas las voces pueden ser escuchadas y respetadas.

Conclusión

La labor realizada en este proyecto no solo tiene un impacto técnico en el campo del procesamiento del lenguaje natural y el aprendizaje automático, sino que también tiene implicaciones sociales profundas. Al abordar la discriminación en el entorno digital, el TFG avanza hacia la creación de un espacio online más seguro, inclusivo y justo para todos. Esto es esencial para conseguir cumplir con los Objetivos de Desarrollo Sostenible y para la construcción de una sociedad donde todas las personas puedan vivir con dignidad y respeto.