



Universidad Nacional de La Matanza
Departamento de Ingeniería e Investigaciones Tecnológicas

Proyecto:

Aplicación de Inteligencia Artificial

Profesores:

Dr. Ierache, Jorge

Dr. Becerra Martín

Ing. Sanz Diego

Integrantes:

DNI	APELLIDO	NOMBRE	EMAIL
40137584	Forestiero	Camila Julieta	cforestiero@alumno.unlam.edu.ar
43034043	Agasi	Alejo Agustín	aagasi@alumno.unlam.edu.ar

Índice

Dominio elegido.....	3
Problema identificado.....	4
Casos de uso.....	5
Caso de uso 1: Identificar memes ofensivos.....	5
Caso de uso 2: Capturar imágenes desde una extensión.....	6
Caso de uso 3: Visualizar galería de memes procesados.....	6
Caso de uso 4: Actualizar la galería de memes procesados.....	6
Diagrama de casos de uso.....	7
Wireframes.....	8
Tareas identificadas.....	11
Investigación del Estado del Arte.....	12
Síntesis de papers elegidos.....	12
Multimodal Meme Dataset (MultiOFF) for Identifying Offensive Content in Image and Text	
Resumen del Artículo.....	14
Problema.....	14
Solución.....	14
Resumen de la Solución Propuesta.....	14
Descripción de los Datasets Utilizados.....	14
Preparación de Datos.....	14
Algoritmos/Arquitecturas/Modelos Utilizados y Configuración de Hiperparámetros..	15
Resultados.....	16
Evaluación.....	16
Comparación con Otras Soluciones.....	16
Vilio: State-of-the-art Visio-Linguistic Models applied to Hateful Memes.....	17
Resumen del artículo.....	17
Problema.....	17
Solución.....	17
Resumen de la solución propuesta.....	17
Descripción del dataset utilizado.....	17
Preparación de datos.....	17
Algoritmos/Arquitecturas/Modelos y Configuración de Hiperparámetros.....	18
Resultados.....	18
Evaluación.....	18
Comparación con otras soluciones.....	19
Detecting Hate Speech in Memes Using Multimodal Deep Learning Approaches:	
Prize-winning solution to Hateful Memes Challenge.....	20
Resumen del Artículo.....	20
Problema.....	20
Solución.....	20
Resumen de la Solución Propuesta.....	20
Descripción de los Datasets Utilizados.....	21

Preparación de Datos.....	21
Algoritmos/Arquitecturas/Modelos Utilizados y Configuración de Hiperparámetros..	21
Resultados.....	23
Evaluación y Comparación con otras soluciones.....	23
Reproducción del Paper Multimodal Meme Dataset (MultiOFF) for Identifying Offensive Content in Image and Text.....	25
Resumen de la Prueba a Realizar.....	25
Preparación de Datos.....	25
Algoritmos y Configuración de Hiperparámetros.....	25
Resultados Obtenidos.....	26
Link a Colab.....	26
Diseño de prueba de concepto.....	27
Esquema de Pipeline Diseñado.....	27
Modelos Elegidos.....	30
Datasets involucrados.....	30
Test y Evaluaciones de Modelos.....	31
Link al Notebook.....	33
Aplicación.....	34
Diseño de Arquitectura.....	34
Componentes Principales.....	34
Interacción entre Componentes.....	35
Modelado UML.....	36
Diagrama de componentes.....	36
Diagrama de secuencia.....	37
Implementación.....	38
Capturas de pantalla del sistema.....	38
Presentación final.....	41
Hoja de Ruta.....	41
Capturas de la Presentación Final.....	42
Link a Video de Presentación.....	61
Referencias.....	62

Dominio elegido

Según la Real Academia Española (RAE), un meme se define como una "imagen, video o texto, por lo general distorsionado con fines caricaturescos, que se difunde principalmente a través de Internet."

En el contexto actual, los memes han evolucionado para convertirse en una forma popular de comunicación en línea, a menudo utilizados para transmitir mensajes humorísticos, satíricos o, en ocasiones, ofensivos. La proliferación de contenido generado por los usuarios ha llevado a la necesidad de desarrollar herramientas que permitan identificar y filtrar aquellos memes que contienen lenguaje o imágenes ofensivas.

En consecuencia, este proyecto se desarrolla en el ámbito del análisis y moderación de contenido multimodal en entornos digitales, con un enfoque específico en la detección de memes ofensivos. Para abordar este desafío, el dominio combina dos áreas clave de la inteligencia artificial: la visión por computadora, que permite analizar las imágenes, y el procesamiento del lenguaje natural (NLP), que se encarga de interpretar el texto presente en los memes. Esta intersección es fundamental para capturar el contexto completo de los memes, dado que el contenido ofensivo suele depender tanto de la imagen como del texto.

La capacidad de detectar memes ofensivos no solo es relevante desde una perspectiva técnica, sino que también tiene implicaciones significativas en la promoción de un entorno en línea más seguro y respetuoso. La implementación de esta tecnología puede ayudar a mitigar la propagación de discursos de odio y otros tipos de contenido dañino, promoviendo así un uso más saludable de las redes sociales y otras plataformas de comunicación digital.

Problema identificado

El problema principal que aborda este proyecto es la detección automática de memes ofensivos mediante un enfoque multimodal [1], es decir, un enfoque que utiliza simultáneamente datos visuales (imágenes) y datos textuales (texto incrustado en las imágenes).

En relación a ello, los memes ofensivos suelen ser difíciles de identificar debido a varias razones:

1. **Complejidad semántica:** El texto dentro de un meme puede ser irónico, sarcástico o usar lenguaje figurativo, lo que dificulta que los enfoques simples de análisis de texto puedan captar el significado real.
2. **Contexto visual:** La imagen dentro del meme puede incluir elementos que, combinados con el texto, cambian completamente el significado. Por ejemplo, un texto aparentemente inocente puede volverse ofensivo cuando se superpone a una imagen que contiene connotaciones históricas, políticas o sociales. Este tipo de complejidad requiere que el sistema no solo "lea" el texto, sino que también interprete los componentes visuales de la imagen.
3. **Naturaleza multimodal:** La verdadera dificultad del problema reside en que un meme ofensivo no siempre puede ser detectado analizando únicamente el texto o solo la imagen. El mensaje ofensivo generalmente emerge de la interacción entre ambos componentes. Esto hace que los modelos puramente unidimensionales (como los que analizan solo imágenes o solo texto) sean insuficientes para este tipo de tarea.

De esta manera, detectar contenido ofensivo es importante no solo para evitar la difusión de mensajes de odio, sino también para proteger a los usuarios y crear un ambiente digital más inclusivo y seguro. Los memes ofensivos pueden perpetuar estereotipos de raza, género, religión u orientación sexual, y son una herramienta común en el acoso en línea.

Casos de uso

Caso de uso 1: Identificar memes ofensivos

- **Actor:** Usuario (a través de la extensión de navegador)
- **Objetivo:** El objetivo del usuario es enviar una imagen a la aplicación para que esta determine si el meme es ofensivo o no.
- **Flujo normal:**
 1. Iniciar Extensión:
 - El usuario abre la extensión del navegador.
 2. Capturar Imagen:
 - El usuario captura una imagen del meme que desea analizar mediante un botón.
 3. Preparación y Envío de la Imagen:
 - La extensión convierte la imagen a formato base64.
 - La extensión envía la imagen en base64 al servidor Flask [2] mediante una solicitud POST.
 4. Recepción y Procesamiento en el Servidor:
 - El servidor Flask recibe la imagen.
 - Flask genera un hash único de la imagen.
 - Flask verifica si el hash de la imagen ya ha sido procesado anteriormente.
 5. Procesamiento de Imagen Nueva:
 - *[Condición: La imagen no ha sido procesada previamente]*
 - Flask procesa la imagen utilizando un algoritmo para identificar si es ofensiva.
 - Flask almacena el hash de la imagen y el resultado del análisis en memoria.
 - Flask responde a la extensión con el resultado del análisis (ofensivo, no ofensivo).
 6. Mostrar Resultado al Usuario:
 - La extensión de navegador muestra el resultado al usuario.
- **Flujo Alternativo:**
 - Paso 5.1: Imagen Ya Procesada
 - *[Condición: La imagen ha sido procesada previamente]*
 - Flask recupera el resultado almacenado asociado al hash de la imagen (sin necesidad de volver a procesarla).
 - Flask responde a la extensión con el resultado previamente almacenado.
 - El flujo continúa en el paso 6 del flujo normal.

Caso de uso 2: Capturar imágenes desde una extensión

- **Actor:** Usuario (a través de la extensión de navegador)
- **Objetivo:** El objetivo es capturar una imagen desde la extensión del navegador y enviarla al servidor Flask [2] para su posterior procesamiento.
- **Flujo normal:**
 1. El usuario abre la extensión de navegador en su dispositivo.
 2. El usuario selecciona una imagen del contenido visible en la página web y oprime el botón Capture Image.
 3. La extensión convierte la imagen seleccionada en base64.
 4. La extensión envía la imagen en formato base64 al servidor Flask a través de una solicitud POST.
 5. Flask recibe la imagen para su posterior procesamiento.
 6. La extensión de navegador espera la respuesta del servidor Flask con el resultado del análisis de la imagen.

Caso de uso 3: Visualizar galería de memes procesados

- **Actor:** Usuario (a través de la interfaz Gradio)
- **Objetivo:** Permitir al usuario visualizar una galería de memes previamente procesados, junto con sus resultados.
- **Flujo normal:**
 1. El usuario accede a la interfaz Gradio [3].
 2. El usuario oprime el botón Refresh.
 3. Gradio envía una solicitud al backend Flask [2] para obtener la lista de memes procesados y sus resultados almacenados en memoria.
 4. Flask responde con la lista de imágenes y los resultados correspondientes.
 5. Gradio muestra las imágenes y sus resultados en una galería visual.
 6. El usuario puede actualizar la galería si es necesario para ver nuevos memes que hayan sido procesados.

Caso de uso 4: Actualizar la galería de memes procesados

- **Actor:** Usuario (a través de la interfaz Gradio)
- **Objetivo:** Permitir al usuario actualizar la galería de memes en Gradio [3] para mostrar los últimos memes procesados.
- **Flujo normal:**
 1. El usuario hace clic en el botón "Refresh" en la interfaz Gradio.
 2. Gradio envía una solicitud al backend Flask [2] para obtener las últimas imágenes y resultados almacenados en memoria.
 3. Flask devuelve la lista de memes más recientes procesados, con sus resultados.
 4. La galería en Gradio se actualiza con la información más reciente.
 5. El usuario visualiza los nuevos memes procesados.

Diagrama de casos de uso

En la *Figura 1* se pueden observar los casos de uso del sistema detector de memes ofensivos.

El caso de uso **Identificar memes ofensivos** tiene por objetivo enviar una imagen a la aplicación para que esta determine si el meme es ofensivo o no. **Capturar imágenes desde una extensión** consiste en capturar una imagen desde la extensión del navegador y enviarla al servidor Flask [2] para su posterior procesamiento. **Visualizar galería de memes procesados** permite al usuario visualizar una galería de memes previamente procesados, junto con sus resultados. **Actualizar la galería de memes procesados** permite al usuario actualizar la galería de memes en Gradio [3] para mostrar los últimos memes procesados.

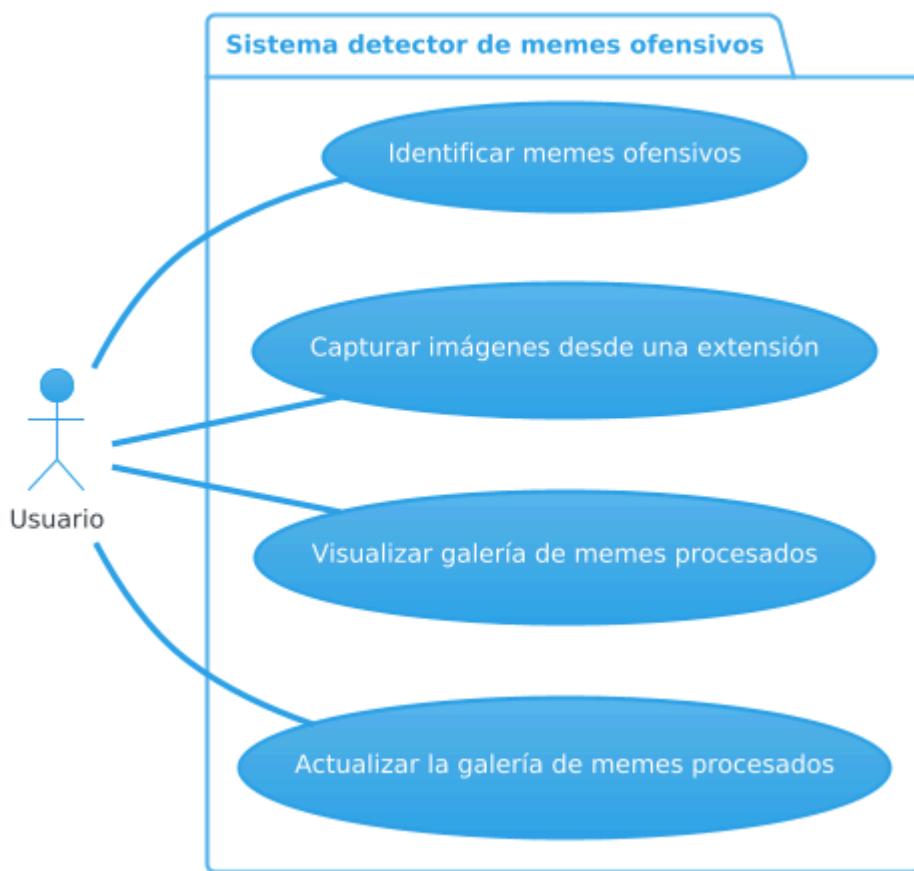


Figura 1: Diagrama de casos de uso de sistema detector de memes ofensivos

Wireframes

En la *Figura 2* puede apreciarse la extensión del navegador que permite capturar la imagen de un meme y analizarla.

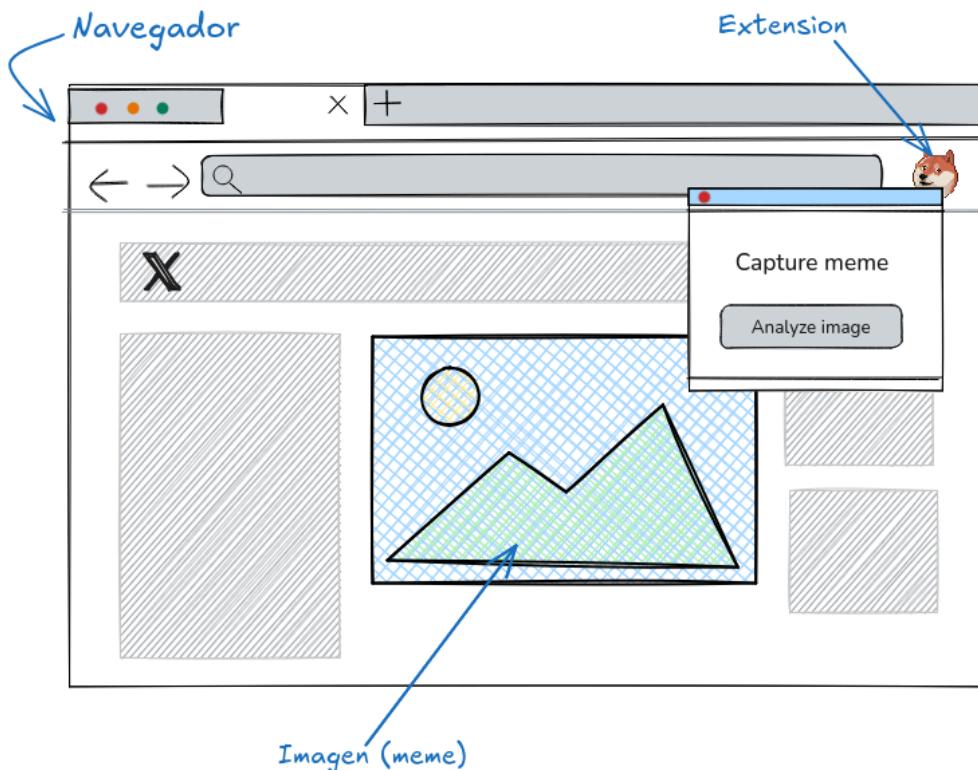


Figura 2: Wireframe mostrando la extensión del navegador

En la *Figura 3* vemos la notificación obtenida una vez hecho el análisis del meme, siendo el resultado en este caso un meme no ofensivo.

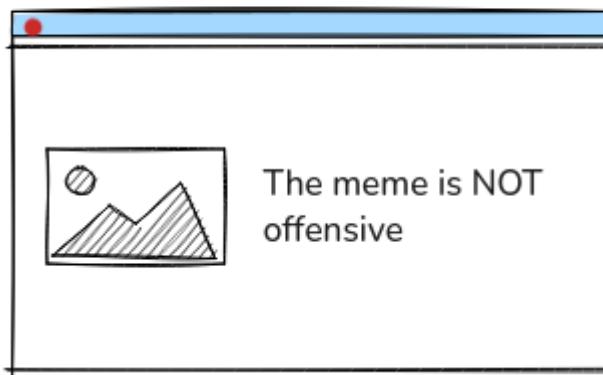


Figura 3: Wireframe de notificación de meme no ofensivo

En la *Figura 4* también vemos la notificación obtenida una vez hecho el análisis del meme, pero siendo el resultado en este caso un meme ofensivo.

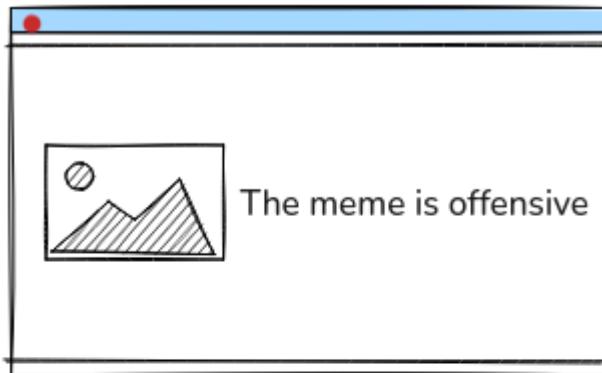


Figura 4: Wireframe de notificación de meme ofensivo

En la *Figura 5* se muestra el caso donde se quiere analizar una imagen que ya fue previamente analizada, ofreciendo la posibilidad de volver a hacerlo.

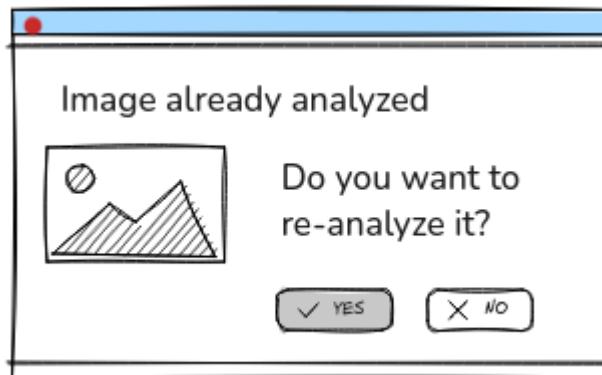


Figura 5: Wireframe de notificación para poder volver a analizar la imagen

Por último, en la *Figura 6* se visualiza la galería con todas las imágenes que fueron analizadas y el resultado asociado a cada una de ellas.

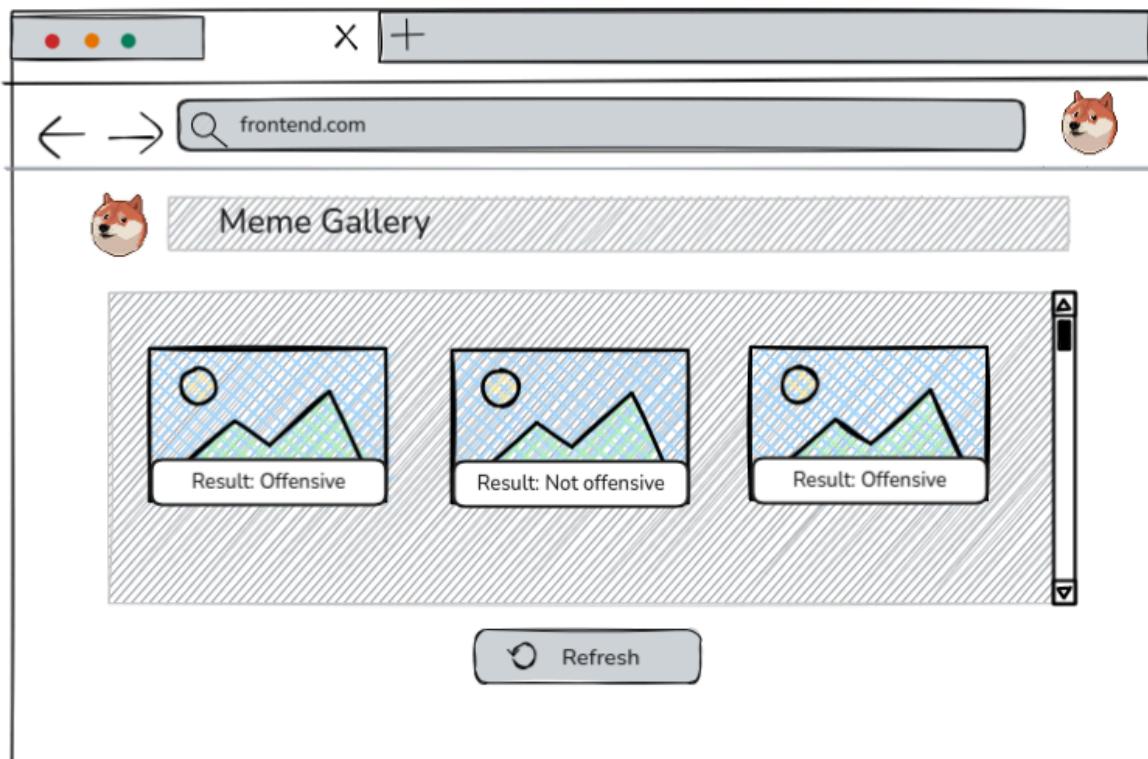


Figura 6: Wireframe de visualización de la galería de imágenes

Tareas identificadas

1. Extracción de Texto de Imágenes mediante OCR (Reconocimiento Óptico de Caracteres)

El OCR es una tecnología que permite convertir texto presente en imágenes en texto digital editable y procesable. Esta técnica es esencial cuando se trabaja con imágenes que contienen texto incrustado, ya que posibilita la transformación de información visual en datos textuales que pueden ser analizados.

En el contexto de nuestra aplicación para la identificación de memes ofensivos, el OCR se utiliza para extraer el texto contenido en los memes. Al convertir el texto de la imagen en formato digital, podemos aplicar técnicas de Procesamiento de Lenguaje Natural (NLP) para evaluar el contenido y determinar si es ofensivo.

2. Clasificación de Texto (Text Classification)

La clasificación de texto es una tarea fundamental en el Procesamiento de Lenguaje Natural que consiste en asignar categorías o etiquetas predefinidas a fragmentos de texto basándose en su contenido. Esta técnica utiliza algoritmos y modelos de aprendizaje automático para analizar patrones lingüísticos, identificando características clave que permiten clasificar el texto de manera precisa.

Una vez extraído el texto de los memes mediante OCR, nuestra aplicación utiliza la clasificación de texto para determinar si el contenido es "ofensivo" o "no ofensivo". El algoritmo analiza el texto en busca de palabras clave, expresiones y contextos que puedan indicar lenguaje inapropiado, discriminatorio o que incite al odio.

3. Reconocimiento de Imágenes (Image Recognition)

El reconocimiento de imágenes es una disciplina de la visión por computadora que se enfoca en analizar y comprender el contenido visual de una imagen. Utiliza técnicas avanzadas de aprendizaje automático, incluyendo redes neuronales convolucionales, para identificar y clasificar objetos, patrones, símbolos y otras características visuales presentes en las imágenes.

En nuestra aplicación, el reconocimiento de imágenes se implementa para analizar elementos visuales de los memes que podrían ser ofensivos. Esto incluye la identificación de símbolos asociados con discursos de odio, gestos inapropiados, imágenes violentas o cualquier otro contenido visual que pueda ser considerado inadecuado. Al integrar el reconocimiento de imágenes, la aplicación no solo se limita al análisis del texto, sino que también evalúa el componente visual del meme. Esto es crucial, ya que algunos memes pueden transmitir mensajes ofensivos exclusivamente a través de imágenes o mediante la combinación de texto e imagen. Al considerar ambos aspectos, la aplicación logra una detección más completa y efectiva de contenido ofensivo.

Investigación del Estado del Arte

Síntesis de papers elegidos

En esta sección se enumeran y explican brevemente los features del estado del arte y de nuestro proyecto.

Identificación de memes ofensivos

Este feature implica el análisis y clasificación de memes para determinar si contienen contenido ofensivo o no.

Integración de datos textuales y visuales

La capacidad de procesar tanto el texto incluido en el meme como sus características visuales. Esta combinación permite que los modelos tengan un contexto más completo, mejorando la precisión en la clasificación.

Uso de modelos multimodales

Se refiere a la utilización de modelos que combinan datos visuales y textuales. Por ejemplo, procesar una imagen junto con el texto extraído de la misma mejora la comprensión global del meme.

Ensamblado de modelos

El ensamblado de modelos consiste en combinar varios modelos de machine learning o deep learning para mejorar el rendimiento en tareas específicas.

Técnicas de Data Augmentation

Estas técnicas aumentan la cantidad de datos disponibles para entrenar un modelo mediante la creación de variaciones sintéticas de los datos existentes, como rotación de imágenes, modificaciones en el brillo, o cambio de texto.

Uso de OCR para extracción de texto

El OCR (Reconocimiento Óptico de Caracteres) permite extraer texto directamente de las imágenes.

Captura de imagen en navegador

Una funcionalidad que permite al usuario capturar memes directamente desde el navegador, facilitando la integración y recolección de datos.

Notificación de resultados en navegador

Se refiere a la posibilidad de que el sistema notifique los resultados directamente al navegador, mejorando la interacción con el usuario.

Visualización en galería de memes analizados

Este feature permite al usuario visualizar de manera organizada todos los memes que ya han sido analizados. Es útil para tener un historial o referencia visual, y está diseñado específicamente en nuestro proyecto.

Funcionalidad	Multimodal Meme Dataset (MultiOFF)	Vilio	Detecting Hate Speech in Memes	Nuestro Proyecto
Identificación de memes ofensivos	✗	✗	✗	✗
Integración de datos textuales y visuales	✗	✗	✗	✗
Uso de modelos multimodales	✗	✗	✗	✗
Ensamblado de modelos		✗	✗	
Técnicas de Data Augmentation			✗	✗
Uso de OCR para extracción de texto		✗		✗
Captura de imagen en navegador				✗
Notificación de resultados en navegador				✗
Visualización en galería de memes analizados				✗

Tabla 1: Tabla comparativa de las funcionalidades ofrecidas en cada paper y en nuestro proyecto

En nuestro proyecto, como muestra la *Tabla 1*, nos diferenciamos por:

- Captura de imagen en navegador: A través de una extensión de navegador, se podrá capturar la imagen del meme de la página web, facilitando así la recopilación de memes para su análisis.
- Notificación de resultados en navegador: A través de la misma extensión, se recibirá una notificación en el navegador con la clasificación resultante del meme analizado.
- Visualización en galería de memes analizados: Permite a los usuarios ver fácilmente todos los memes analizados hasta el momento y sus correspondientes clasificaciones.

Multimodal Meme Dataset (MultiOFF) for Identifying Offensive Content in Image and Text

Resumen del Artículo

El artículo [4] presenta la creación del dataset MultiOFF, un conjunto de datos multimodales diseñado para identificar contenido ofensivo en memes, utilizando tanto texto como imágenes. Los memes combinan modalidades de texto e imagen, lo que dificulta la clasificación de contenido ofensivo. Se desarrolló un clasificador multimodal usando una técnica de fusión temprana que combina ambas modalidades y se compara con enfoques basados solo en texto o solo en imagen, mostrando mejoras en precisión, recall y F-score.

Problema

El problema que aborda el artículo es la detección de contenido ofensivo en memes, que es difícil de clasificar debido a su naturaleza multimodal y al uso frecuente de humor o sarcasmo implícito. Anteriormente, la detección de este tipo de contenido se había centrado sólo en modalidades individuales (texto o imagen), lo que resultaba insuficiente para captar el contexto completo de los memes.

Solución

Resumen de la Solución Propuesta

El equipo creó el dataset MultiOFF basado en memes de las elecciones presidenciales de EE. UU. de 2016 y desarrolló un clasificador multimodal que utiliza una técnica de fusión temprana para combinar el análisis de texto e imágenes. Este clasificador multimodal supera a los modelos basados únicamente en texto o imagen en la detección de contenido ofensivo.

Descripción de los Datasets Utilizados

El dataset MultiOFF contiene 743 memes anotados como ofensivos o no ofensivos. Este conjunto de datos contiene las URL de las imágenes y el texto incrustado en ellas. Los memes fueron obtenidos de sitios de redes sociales como Reddit, Facebook, Twitter e Instagram, con énfasis en el contenido relacionado con las elecciones presidenciales de 2016 en EE. UU..

Preparación de Datos

El conjunto de datos original incluía características no relacionadas, como un timestamp (fecha de publicación), un enlace (URL de la publicación), autor, red, me gusta o votos positivos. Se eliminaron aquellas que no cumplían con el objetivo de la investigación, es decir, solo se utilizó el enlace URL y el texto (título) del conjunto de datos existente.

Se eliminaron stopwords y símbolos no alfanuméricos, se convirtió todo el texto a minúsculas y se mantuvieron sólo aquellos memes con texto de hasta 20 oraciones. Finalmente, se procedió a la vectorización del texto.

En cuanto a las imágenes, las URLs fueron verificadas y las imágenes se descargaron localmente para entrenar los clasificadores. Además, fueron transformadas en vectores entrenables.

Por último, como había más memes no ofensivos que ofensivos, el conjunto de datos se balanceó mediante el uso de diferentes pesos de clase durante el entrenamiento del clasificador.

Algoritmos/Arquitecturas/Modelos Utilizados y Configuración de Hiperparámetros

Se utilizaron diferentes modelos para clasificar los memes, por lo que se procederá a detallar las características principales de cada uno:

1. Stacked LSTM + VGG16:

- Modelo de Texto (Stacked LSTM): Usa embeddings preentrenados de GloVe (50 dimensiones). Dos capas de LSTM apiladas con 32 unidades cada una, procesan secuencias de texto.
- Modelo de Imagen (VGG16): VGG16 preentrenado en ImageNet, con capas congeladas, y una capa de GlobalAveragePooling2D.
- Modelo Combinado: Fusiona las salidas del LSTM y VGG16. La salida final es una capa densa con activación sigmoid. Se utiliza el optimizador Adam ($lr=0.001$) y la función de pérdida Binary Cross-entropy. Class weights: (1: 1.4 para ofensivo, 0: 1.0). Epochs: 7. Batch size: 32 para entrenamiento, 1 para validación y 1 para prueba.

2. BiLSTM + VGG16:

- Modelo de Texto (BiLSTM): Usa embeddings de GloVe (50 dimensiones). Una capa de BiLSTM (LSTM bidireccional) con 32 unidades que captura tanto el contexto anterior como posterior.
- Modelo de Imagen (VGG16): VGG16 preentrenado con capas congeladas y una capa de GlobalAveragePooling2D.
- Modelo Combinado: Combina las salidas del BiLSTM y VGG16. La salida es una capa densa con activación sigmoid. Se utiliza el optimizador Adam ($lr=0.001$) y la función de pérdida Binary Cross-entropy. Pesos de clase: (1: 1.4, 0: 1.0). Epochs: 7. Batch size: 32 para entrenamiento, 1 para validación y 1 para prueba.

3. CNN para Texto + VGG16:

- Modelo de Texto (CNN para texto): Usa embeddings preentrenados de GloVe (50 dimensiones). Varias capas de convolución 1D seguidas de max pooling extraen características de texto.
- Modelo de Imagen (VGG16): VGG16 congelado con salida GlobalAveragePooling2D.
- Modelo Combinado: Fusiona las salidas de la CNN para texto y el modelo de imagen VGG16. La salida es una capa densa con activación sigmoid. Se utiliza el optimizador Adam ($lr=0.001$) y la función de pérdida Binary Cross-entropy. Pesos de clase: (1: 1.4, 0: 1.0). Epochs: 7. Batch size: 32 para entrenamiento, 1 para validación y 1 para prueba.

4. Regresión Logística (LR), Naive Bayes (NB) y DNN (Deep Neural Network):

Estos modelos fueron usados para la predicción de memes ofensivos considerando solo el texto de los memes.

Puede destacarse respecto a DNN que, la capa de embeddings tiene entradas de 100 dimensiones y genera embeddings de 50 dimensiones, la salida es una capa densa con activación sigmoid para la clasificación, y usa Adam como optimizador y la función de pérdida Binary Cross-entropy para clasificación binaria.

Resultados

Evaluación

Se evaluaron los modelos utilizando precisión, recall y F-score. El modelo de fusión temprana, que combina texto e imagen, mostró un recall más alto en la detección de memes ofensivos en comparación con los modelos que utilizan solo texto o imagen. Sin embargo, aún es discutible si la precisión de un enfoque multimodal de este tipo es confiable (el máximo logrado fue 0.40).

Comparación con Otras Soluciones

Los resultados mostraron que los modelos multimodales superaron a los enfoques basados únicamente en texto o imagen en términos de precisión y recall. El modelo de CNN en combinación con VGG16 ofreció la mejor capacidad para recuperar memes ofensivos.

Cabe aclarar que el clasificador de texto muestra una precisión cercana al clasificador multimodal y, a veces, mejor. Por otra parte, si bien el clasificador de imágenes tiene una menor probabilidad de identificar y retener memes ofensivos por sí solo, el clasificador multimodal muestra mejoras en la retención de memes ofensivos. Por lo tanto, esto sugiere que hay más posibilidades de mejorar la precisión al aumentar el peso de las características textuales al combinarlo con elementos visuales del meme.

Vilio: State-of-the-art Visio-Linguistic Models applied to Hateful Memes

Resumen del artículo

Este artículo presenta Vilio [5], una plataforma que implementa múltiples modelos visio-lingüísticos de última generación para abordar el problema de la detección de memes ofensivos. Los memes, que combinan texto e imágenes, presentan un desafío único en cuanto a la interpretación del mensaje ofensivo, ya que tanto el texto como la imagen deben ser considerados simultáneamente. La solución propuesta por Vilio optimiza modelos como VisualBERT, UNITER, OSCAR y ERNIE-ViL, utilizando un proceso de tres etapas: preparación de datos, modelado y ensamblado de modelos. Estos esfuerzos lograron el segundo lugar entre 3 mil participantes en el Hateful Memes Challenge de Facebook.

Problema

El problema principal es la detección automática de memes ofensivos en redes sociales, lo que implica procesar tanto el texto como la imagen para determinar si un meme es perjudicial. A diferencia de las tareas convencionales de clasificación de texto o imágenes, los memes requieren una interpretación visio-lingüística, donde ni el texto ni la imagen son suficientes por sí solos para capturar el mensaje. El Hateful Memes Dataset, creado por Facebook AI, presenta confusiones intencionadas para garantizar que los modelos comprendan tanto el contexto visual como el textual.

Solución

Resumen de la solución propuesta

Vilio emplea un conjunto de 12 modelos visio-lingüísticos que integran redes de transformadores para procesar texto e imagen simultáneamente. El enfoque incluye tres etapas clave: Preparación, Modelado y Ensamblado.

Descripción del dataset utilizado

El Hateful Memes Dataset incluye 8,500 memes para entrenamiento, 500 para validación y 1,000 para prueba. Los memes contienen texto superpuesto a las imágenes, y los datos de texto también se proporcionan en formato JSON para facilitar el procesamiento. El conjunto de datos incluye confusiones donde se intercambian elementos textuales o visuales para evaluar si los modelos pueden interpretar correctamente la interacción entre ambos.

Preparación de datos

Se utilizaron características visuales extraídas con Detectron2, utilizando diferentes configuraciones de regiones de interés (Rois). Estas características visuales y los textos extraídos a través de OCR se utilizaron como entrada para los modelos. La combinación de múltiples configuraciones de características visuales y el preentrenamiento de los modelos en datasets como VisualGenome permitieron mejorar el rendimiento de la solución.

Algoritmos/Arquitecturas/Modelos y Configuración de Hiperparámetros

Se implementaron varios modelos de transformadores visio-lingüísticos, detallados individualmente a continuación, que demostraron ser efectivos en la tarea de procesamiento multimodal. Estos modelos procesan tanto la imagen como el texto de manera conjunta o en flujos separados, utilizando capas de atención para capturar las interacciones entre las dos modalidades.

1. ERNIE-VIL: este modelo basado en VilBERT y el transformador ERNIE, es preentrenado con labels del tipo “verdad fundamental”, ya que el Hateful Memes Dataset no cuenta con este tipo de labels se extrajo un set de features para utilizarse como falsa “verdad fundamental”.

2. UNITER y OSCAR: se realizaron actualizaciones en ambos modelos, como una función de activación actualizada y cálculos de embeddings. OSCAR también está preentrenado específicamente para la tarea de memes ofensivos utilizando Image-Text Matching (ITM) y Masked Language Modelling (MLM). Se utilizó el clasificador de LXMERT con activación GeLU. Los pesos preentrenados de OSCAR y UNITER están basados en el transformador BERT.

3. VisualBERT: al igual que OSCAR, el modelo VisualBERT está preentrenado específicamente en la tarea utilizando MLM. Mientras que el VisualBERT original usa los mismos identificadores de tipo de token para la entrada visual y de lenguaje, la implementación de Vilbert crea un tipo de token visual separado. Por lo tanto, los pesos de tipo de token se reinician y se vuelven a entrenar desde cero. Esto mejoró el modelo en un 1.2% absoluto en la métrica AUROC(Area Bajo la Curva Característica Operativa del Receptor).

Los modelos se optimizaron utilizando el optimizador Adam con una tasa de aprendizaje de 1e-5 y una función de pérdida de entropía cruzada binaria. El entrenamiento se llevó a cabo en lotes de tamaño 8, y los modelos fueron entrenados durante 5 épocas, con técnicas adicionales como Stochastic Weight Averaging para mejorar la estabilidad y el rendimiento.

Resultados

Evaluación

La evaluación se realizó principalmente utilizando la métrica AUROC, que mide la capacidad del modelo para clasificar correctamente los memes ofensivos. Vilbert alcanzó un AUROC de 82.52 en el conjunto de pruebas, lo que se acercó mucho al rendimiento humano (82.65).

Comparación con otras soluciones

Comparado con los modelos individuales como VisualBERT (AUROC de 71.33), UNITER (78.65) y ERNIE-ViL Large (80.59), el ensamblado de Vilio logró cerrar la brecha entre el rendimiento de los modelos basados en transformadores y el rendimiento humano en la detección de memes ofensivos. Esta combinación de múltiples modelos, junto con técnicas de ensamblado, permitió a Vilio mejorar significativamente su rendimiento en comparación con soluciones previas y ofrecer una solución más robusta.

Detecting Hate Speech in Memes Using Multimodal Deep Learning Approaches: Prize-winning solution to Hateful Memes Challenge

Resumen del Artículo

El paper [\[6\]](#) describe una solución para detectar discursos de odio en memes, enfocándose en datos multimodales. Los autores utilizaron un modelo preentrenado VisualBERT (un modelo basado en BERT para tareas de visión y lenguaje) y aplicaron técnicas de aprendizaje en conjunto (ensemble learning) para mejorar el rendimiento.

Problema

Los memes, aunque a menudo son humorísticos, pueden ser utilizados para difundir discursos de odio, especialmente cuando se combinan imágenes y texto. Detectar este tipo de contenido es un desafío debido a su naturaleza multimodal.

Solución

Resumen de la Solución Propuesta

La solución consiste en el modelo VisualBERT, un BERT multimodal para la combinación de visión y lenguaje. Se describe cómo se llevó a cabo la solución para detectar los memes de odio. La metodología incluye:

- **Expansión del conjunto de datos:** Se amplió el conjunto de entrenamiento con más memes, utilizando tanto muestras no vistas en otros conjuntos como un análisis cuidadoso del conjunto de datos Memotion, agregando más ejemplos relevantes.
- **Codificación de imágenes:** Se utilizó un modelo especializado para extraer características visuales (Mask-RCNN) y se integraron en el mismo espacio que las características textuales, permitiendo que el modelo procese conjuntamente las imágenes y el texto de los memes.
- **Entrenamiento del modelo:** Se usó el modelo VisualBERT preentrenado en datos de imágenes y texto, que luego fue ajustado con los datos del desafío de memes de odio. La salida del modelo se utilizó para clasificar si los memes eran de odio o no.
- **Aprendizaje en conjunto:** Se combinaron varios modelos entrenados utilizando una técnica de votación mayoritaria para mejorar la precisión del modelo final. Este enfoque ayuda a balancear las debilidades de los modelos individuales, logrando un mejor rendimiento que cualquiera de los modelos por separado.

Descripción de los Datasets Utilizados

El conjunto de datos de memes de odio que ofrece Meta no está creado para entrenar modelos desde cero, sino para ajustar y probar modelos multimodales preentrenados a gran escala. Por lo tanto, el tamaño del conjunto de datos (10K imágenes) es pequeño en comparación con conjuntos de datos como Visual Genome (108K), COCO (330K) y Conceptual Captions (3.3M).

Preparación de Datos

Expansión del Conjunto de Datos

Más datos proporcionan un aprendizaje estable y mejores puntuaciones. Por lo tanto, se buscaron fuentes de datos adicionales con el fin de aumentar el tamaño del conjunto de datos, y como resultado, ampliamos los datos de entrenamiento con 428 memes adicionales.

El conjunto de datos se divide en tres partes: un conjunto de entrenamiento de 8,500 muestras, un conjunto de desarrollo de 500 muestras y un conjunto de prueba de 1,000 muestras. Además de este conjunto de prueba "visto", se ha publicado un nuevo conjunto de pruebas de 2,000 muestras, donde los ganadores se determinan de acuerdo con su rendimiento en este conjunto de prueba "no visto".

Algoritmos/Arquitecturas/Modelos Utilizados y Configuración de Hiperparámetros

Codificación de Imágenes

Se describe cómo procesaron las imágenes de los memes para extraer características visuales que luego pudieran ser combinadas con el texto. Se utiliza un modelo llamado Mask-RCNN, que es conocido por su capacidad para detectar objetos y sus regiones dentro de una imagen.

1. Extracción de características visuales:

- Cada imagen es dividida en 100 regiones o "cuadros" que representan diferentes partes de la imagen. A cada una de estas regiones se le extraen características que tienen una dimensión de 2048, lo que significa que cada región se representa con un vector de 2048 números que describen la información visual que contiene.

2. Modelo Mask-RCNN y ResNeXT-152:

- El modelo Mask-RCNN es una red neuronal entrenada para detectar objetos y regiones en imágenes. El modelo que utiliza está basado en ResNeXT-152, una variante avanzada de redes neuronales para el procesamiento de imágenes. Mask-RCNN está entrenado en un conjunto de datos llamado Visual Genome, que contiene imágenes anotadas con información visual detallada (por ejemplo, qué objetos aparecen en la imagen y sus relaciones).

3. Proyección de las características visuales:

- Las características visuales de estas 100 regiones se proyectan en el espacio de las incrustaciones textuales. Esto es importante porque el modelo final necesita trabajar tanto con información visual (imágenes) como textual (palabras), y para hacerlo de manera eficiente, las características de ambos tipos de datos deben estar representadas en un espacio similar.
- Para proyectar estas características, el modelo aprende pesos que transforman el vector de 2048 dimensiones de cada región de la imagen en un vector de 768 dimensiones, que es el tamaño de las incrustaciones textuales que se utilizarán más adelante en el modelo.

4. Transformación hacia el modelo final:

- Despues de proyectar las características visuales en el espacio de las características textuales, estas se pasan por las capas del transformador (parte del modelo VisualBERT) para que el modelo pueda procesar conjuntamente las imágenes y el texto.

En la *Figura 7*, se muestra cómo las regiones de la imagen y el lenguaje se combinan con un transformador para permitir que la autoatención descubra alineaciones implícitas entre el lenguaje y la visión.

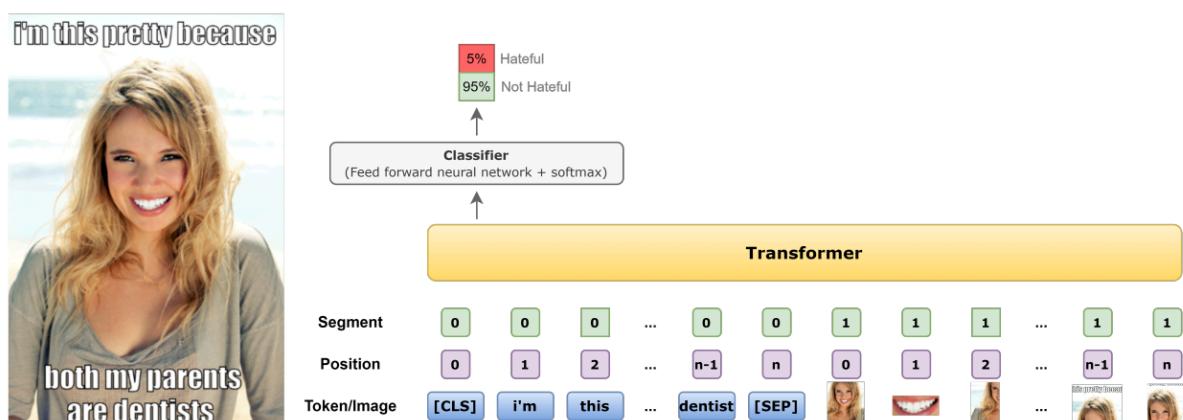


Figura 7: Ejemplo de meme extraido del conjunto de datos (a la izquierda) y una ilustración de la arquitectura del transformador multimodal (a la derecha).

Entrenamiento

- **Preentrenamiento:** VisualBERT se entrenó originalmente en el conjunto de datos de subtítulos de imágenes COCO, se observa que el modelo preentrenado en Conceptual Captions logra puntuaciones notablemente mejores. Por lo tanto, se llevó a cabo la investigación con este último modelo, proporcionado por MMF: un marco para la investigación multimodal de visión y lenguaje de Facebook AI Research (FAIR).

- **Fine-tuning:** Se ajustó el modelo VisualBERT preentrenado en el conjunto de entrenamiento agregado y lo evaluamos en el conjunto de desarrollo no visto.
- **Clasificación:** Se utilizó la primera salida de la capa final como entrada para una capa de clasificación. La misma hace una predicción final (ofensivo o no ofensivo) utilizando una función softmax para convertir las predicciones en probabilidades. Durante el entrenamiento, el modelo se ajusta usando la pérdida de entropía cruzada binaria, que mide cuán acertadas son sus predicciones en comparación con las etiquetas reales, y el modelo aprende a mejorar en base a eso.

Ensemble Learning

El ensemble learning o aprendizaje en conjunto es combinar las predicciones de múltiples modelos base para mejorar la generalización y la robustez en comparación con un solo modelo. Específicamente, se utilizó la técnica de Votación Mayoritaria (también conocida como Hard Voting or Voting Classifier), que combina diferentes clasificadores y utiliza una votación mayoritaria para predecir las etiquetas de las clases. El clasificador resultante suele ser útil para una variedad de modelos igualmente bien entrenados, ya que equilibra sus debilidades individuales. En consecuencia, logra un mejor rendimiento que cualquier modelo individual utilizado en el conjunto.

Resultados

Evaluación y Comparación con otras soluciones

Se construyó una búsqueda de hiperparámetros que resultó en múltiples modelos con diferentes puntuaciones de AUROC en el conjunto de desarrollo no visto. Después de ordenarlos por la puntuación de AUROC, se seleccionaron los 27 mejores modelos para el aprendizaje en conjunto, como puede apreciarse en la *Tabla 2*. Luego, se recopilan las predicciones de cada uno de los modelos y se aplica la técnica de votación mayoritaria: la clase de un punto de datos se determina por la clase más votada. Además, para calcular AUROC, se debe determinar la probabilidad de que un punto de datos sea asignado a una clase: si la clase más votada es 1 (de odio), entonces la probabilidad es la máxima entre todos los 27 modelos y mínima si es clase 0 (no de odio).

ID	Validation	
	Acc.	AUROC
1	70.93	75.21
2	69.63	75.16
3	70.74	75.02
...
25	70.56	73.76
26	70.93	73.75
27	69.81	73.68

Tabla 2: Rendimiento de los modelos en conjunto derivados de VisualBERT CC

La técnica de votación mayoritaria (ensemble learning) mejoró tanto la métrica AUROC como la precisión en un **2.5%**.

Esta técnica funciona combinando varios modelos, cada uno especializado en diferentes tipos de detección de discurso de odio. Algunos modelos pueden ser muy buenos detectando odio hacia un grupo particular (por ejemplo, mujeres), pero no tan buenos detectando odio hacia otro grupo (por ejemplo, religión). Al combinar los modelos con votación mayoritaria, el sistema aprovecha lo mejor de cada uno, compensando sus debilidades individuales. Esto genera un modelo final más robusto.

Reproducción del Paper Multimodal Meme Dataset (MultiOFF) for Identifying Offensive Content in Image and Text

Resumen de la Prueba a Realizar

El modelo entrenado clasifica memes como ofensivos o no ofensivos. La tarea consiste en procesar el texto y la imagen de cada meme de forma conjunta, utilizando LSTM para el análisis del texto y VGG16 para el análisis de las imágenes. El modelo fusiona ambos modos ("early fusion") antes de realizar la predicción.

Preparación de Datos

- Se descarga el dataset, que incluye tanto texto como imágenes etiquetadas.
- Para el texto, se realiza limpieza convirtiendo todo a minúsculas, eliminando símbolos no alfanuméricos y stopwords, y aplicando lematización.
- Se convierte el texto a secuencias de enteros (se vectoriza) y nos aseguramos que todas las secuencias tengan la misma longitud (100 palabras).
- Las imágenes se preprocesan (se redimensionan, se normalizan y se convierten a tensores) para adaptarse al formato esperado por VGG16 (224 x 224 px).
- Se convierten las etiquetas categóricas de 'label' en valores numéricos.

Algoritmos y Configuración de Hiperparámetros

Por un lado, se utiliza el modelo **LSTM** con 128 *unidades* para procesar secuencias de texto, capturar dependencias a lo largo de las palabras y retener información sobre la estructura de la oración. Previamente, se define una capa de *embedding* con una dimensión de 100 (*EMBEDDING_DIM* = 100) para convertir las palabras en vectores densos antes de pasarlas a la LSTM.

Por otro lado, el modelo de imágenes está basado en la arquitectura **VGG16**, que ha sido preentrenado con el dataset *ImageNet*. Se utiliza este modelo sin las capas superiores (capa *include_top=False*), y se congela el entrenamiento de todas las capas (es decir, las capas no se actualizan durante el entrenamiento). Después de pasar la imagen por las capas convolucionales de VGG16, se utiliza una capa de *GlobalAveragePooling2D* para reducir la dimensionalidad de la salida.

Luego, se lleva a cabo la fusión de las modalidades ("Early Fusion") mediante una capa de *Concatenate*, lo que permite integrar la información de ambas modalidades para hacer una clasificación conjunta.

Además, se utiliza una capa *Densa* (fully connected) de 256 *neuronas con activación ReLU*, para combinar las representaciones y aprender patrones más complejos. Se incluye una capa de *Dropout* con una *tasa del 50%*, que aleatoriamente desactiva la mitad de las neuronas durante el entrenamiento para evitar el sobreajuste. Y la última capa es una *Densa con una sola neurona y activación sigmoidal* para realizar la clasificación binaria.

Otros de los hiperparámetros establecidos son el uso del *optimizador Adam* (Adaptive Moment Estimation) para ajustar los pesos del modelo, y el de la función de pérdida *Binary Cross-entropy* ya que se trata de un problema de clasificación binaria.

Para finalizar, se establecen para el entrenamiento del modelo, *batch_size* = 32 (controla cuántas muestras de entrenamiento procesa el modelo antes de actualizar sus pesos) y *epochs* = 10 (número de veces que el modelo pasa por el conjunto de datos completo durante el entrenamiento).

Resultados Obtenidos

- Test accuracy: 0.55
- Precision: 0.44
- Recall: 0.60
- F1-Score: 0.51

Vemos que el test accuracy está apenas por encima del azar (50% en un problema binario), por lo que el modelo no está capturando patrones significativos de manera efectiva.

Además, el recall es mayor que la precisión, lo que sugiere que el modelo está identificando muchos de los casos ofensivos (verdaderos positivos), pero a costa de generar más falsos positivos.

Link a Colab

El notebook original se encuentra en [este enlace](#).

Diseño de prueba de concepto

Esquema de Pipeline Diseñado

En la *Figura 8* se observa cómo se organiza el proceso de análisis de memes en forma de pipeline. Comienza en el Inicio, con la captura del meme, que permite al usuario seleccionar un meme desde el navegador y enviarlo al sistema. Luego, pasa por una etapa de procesamiento de imagen y texto, donde se utiliza OCR para extraer el texto del meme y se analizan las características visuales, integrando ambos tipos de datos para un análisis multimodal. Posteriormente, el sistema obtiene el resultado del análisis, como la detección de contenido ofensivo. Este resultado se comunica al usuario mediante la funcionalidad de mostrar notificación en el navegador. Adicionalmente, el pipeline incluye una rama que permite refrescar la pantalla para actualizar los resultados y una etapa final para mostrar resultados en una galería, proporcionando al usuario una vista organizada de todos los memes procesados y sus respectivas clasificaciones.

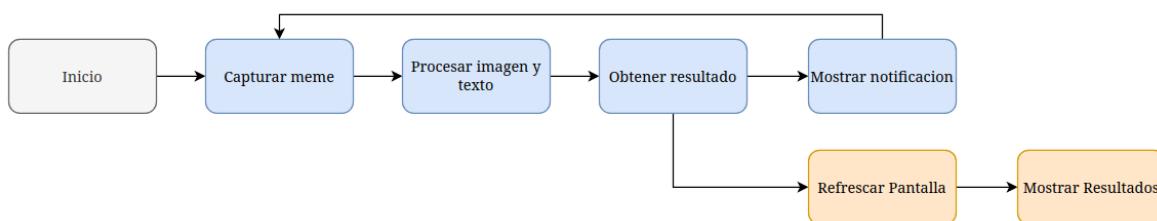


Figura 8: Esquema de pipeline diseñado

1. Capturar meme:

Fuente: Extensión de navegador (acción del usuario).

Acciones:

- El usuario hace clic en un botón de la extensión mientras navega en una web, por ejemplo X (Twitter).
- La extensión capture la imagen del meme de la página activa.
- La imagen se codifica en formato base64 y se envía como una solicitud POST al backend.

Salida: Datos de la imagen codificada enviados al backend de Flask [2] para su procesamiento.

2. Procesar imagen y texto:

Fuente: Backend de Flask [\[2\]](#).

Acciones:

- Procesamiento de Imagen:
 - El backend decodifica la imagen recibida y la procesa usando un modelo pre-entrenado como ResNet para extraer características visuales relevantes.
- Extracción de Texto:
 - Utiliza EasyOCR para extraer el texto presente dentro de la imagen.
- Análisis de Texto:
 - Procesa el texto extraído mediante el modelo DistilBERT para generar embeddings que representan su significado semántico.

Salida: Las características visuales extraídas de la imagen y los embeddings textuales se combinan en un solo conjunto de características para una representación multimodal.

3. Obtener resultado:

Fuente: Backend de Flask [\[2\]](#).

Acciones:

- Generación de Predicción:
 - El conjunto de características combinadas se introduce en una serie de capas adicionales que procesan esta información, obteniendo finalmente una probabilidad.
 - Aplica un umbral sobre esta probabilidad (por ejemplo, ≥ 0.1) para clasificar el meme como ofensivo o no ofensivo..
- Preparación del Resultado:
 - Prepara una respuesta en formato JSON que contiene el resultado ("ofensivo" o "no ofensivo").

Salida: Respuesta guardada en memoria y enviada a la extensión.

4. Mostrar notificación:

Fuente: Extensión de navegador.

Acciones:

- Recepción y Visualización:
 - La extensión recibe la respuesta JSON desde el backend.
 - Desencadena una notificación en el navegador del usuario.
 - La notificación muestra un mensaje (por ejemplo, "El meme es ofensivo").
- Interacción con el usuario:
 - La notificación permite al usuario entender rápidamente el resultado del análisis sin tener que salir de la página actual.

Salida: Notificación en el navegador.

5. Refrescar pantalla:

Fuente: App de Gradio [\[3\]](#).

Acciones:

- Obtención de Datos:
 - Cuando el usuario haga clic en "Refrescar", la app de Gradio solicita los resultados de análisis desde el backend.

Salida: Resultados del análisis.

6. Mostrar resultados:

Fuente: App de Gradio.

Acciones:

- Muestra cada meme procesado en una vista de galería, mostrando la imagen y el resultado de la clasificación.

Salida: Interfaz de usuario para visualizar los resultados del análisis.

Modelos Elegidos

Para la prueba de concepto, se han seleccionado dos modelos para conformar el modelo multimodal de detección de memes ofensivos: **DistilBERT** [7] para el componente textual, y **ResNet** [8] para el componente visual.

DistilBERT (Distilled Bidirectional Encoder Representations from Transformers) es una versión más ligera de BERT [7], optimizada para ser más rápida y menos costosa en términos de recursos computacionales, manteniendo un rendimiento competitivo. Es útil para la clasificación de secuencias de texto, como es el caso de los memes, donde es crucial entender tanto el significado como el contexto del lenguaje.

ResNet (Residual Networks) es un modelo de convolución profunda que introdujo las conexiones residuales o "skip connections", las cuales permiten entrenar redes muy profundas sin los problemas de degradación que enfrentaban las redes tradicionales. ResNet es excelente para la clasificación y análisis de imágenes.

De esta manera, la fusión temprana [9] de ambos modelos permite aprovechar la información cruzada entre ambas modalidades antes de realizar la predicción final.

Datasets involucrados

Se utilizó el dataset **Hateful Memes** [9, 10] desarrollado por el equipo de Facebook AI Research (FAIR). Este está compuesto de memes que requieren que el modelo "entienda" la interacción entre el texto y la imagen para capturar el odio. Es por ello que incluye ejemplos que deliberadamente están diseñados para confundir a los modelos que se enfocan solo en el texto o solo en la imagen.

En cuanto a su estructura, debemos destacar que cada meme posee un ID, una imagen asociada (path), el texto del mismo y está etiquetado de manera binaria como ofensivo (1) o no ofensivo (0). Las etiquetas fueron definidas por humanos que evaluaron el contenido.

El dataset contiene alrededor de 10,000 memes, los cuales están divididos en aproximadamente 8,500 ejemplos de entrenamiento, 500 de validación y 1,000 para pruebas.

Cabe resaltar que el dataset de entrenamiento estaba desbalanceado inicialmente, conteniendo 5481 memes no ofensivos y 3019 memes ofensivos, por lo que se procedió a realizar una aumentación de datos, obteniendo finalmente 5481 memes no ofensivos y 8058 memes ofensivos. Ahora se sigue teniendo un dataset desbalanceado pero a favor de la clase ofensiva, lo cual es de interés para darle más peso a dicha clase, permitiendo que el modelo tenga más sensibilidad hacia los memes de odio.

Test y Evaluaciones de Modelos

Luego de probar y evaluar una gran variedad de configuraciones de los hiperparámetros de los modelos, en busca de las mejores métricas posibles, hemos arribado a las siguientes conclusiones:

Durante el entrenamiento y validación, notamos que cuanto más epochs se utilizaban, si bien el Training Loss (error que comete el modelo en los datos de entrenamiento) disminuía gradualmente, el Validation Loss (error que comete el modelo en los datos de validación) aumentaba, lo cual es una clara muestra de Overfitting. Por ello, teniendo como referencia las métricas utilizadas, se concluyó que 3 epochs eran suficientes para lograr una generalización aceptable, ya que más epochs solo empeoraba la capacidad de generalización del modelo.

Así es que en la *Figura 9* puede apreciarse que el Validation Loss es muy alto y las métricas mejoran levemente, siendo por ello que se decidió que 3 epochs era la elección más sensata.

```

Epoch 1, Training Loss: 0.5530286707696555
Epoch 1, Validation Loss: 1.0628119511529803
Accuracy: 0.562
Precision: 0.535
Recall: 0.8663967611336032
AUC-ROC: 0.5656094477604775

-----
Epoch 2, Training Loss: 0.3276994440093232
Epoch 2, Validation Loss: 1.3938104594126344
Accuracy: 0.584
Precision: 0.5611285266457681
Recall: 0.7246963562753036
AUC-ROC: 0.5856683362404186

-----
Epoch 3, Training Loss: 0.1977994177433296
Epoch 3, Validation Loss: 1.7544608805328608
Accuracy: 0.598
Precision: 0.5871212121212122
Recall: 0.6275303643724697
AUC-ROC: 0.598350162423389

```

Figura 9: Visualización de epochs y métricas durante el entrenamiento del modelo.

También, se evaluaron las probabilidades obtenidas de las predicciones del modelo y se visualizó que:

- Cuando se tenía la certeza de que un meme era no ofensivo, se le asignaba una probabilidad extremadamente baja, tal y como muestra la *Figura 10*.

```

Probabilidad predicha: 1.0191192814090755e-06, Valor verdadero: 0
Probabilidad predicha: 1.4794416074437322e-06, Valor verdadero: 0
Probabilidad predicha: 1.8007673361353227e-06, Valor verdadero: 0
Probabilidad predicha: 2.433605686746887e-06, Valor verdadero: 0
Probabilidad predicha: 2.5550975806254428e-06, Valor verdadero: 0
Probabilidad predicha: 4.175439698883565e-06, Valor verdadero: 0

```

Figura 10: Visualización de probabilidades bajas predichas por el modelo

- Cuando se tenía la certeza de que un meme era ofensivo, se le asignaba una probabilidad alta, tal y como muestra la *Figura 11*.

```
Probabilidad predicha: 0.9977707862854004, Valor verdadero: 1
Probabilidad predicha: 0.9980311989784241, Valor verdadero: 1
Probabilidad predicha: 0.9982140064239502, Valor verdadero: 1
Probabilidad predicha: 0.9983419179916382, Valor verdadero: 1
Probabilidad predicha: 0.9984951019287109, Valor verdadero: 1
Probabilidad predicha: 0.9986270666122437, Valor verdadero: 1
```

Figura 11: Visualización de probabilidades altas predichas por el modelo

- Por ello, se procedió a analizar las probabilidades intermedias y se observó que estas estaban en torno a 0.05, por lo que se adoptó un threshold ligeramente menor de 0.02 como criterio para clasificar los memes ofensivos. Esta decisión se tomó para reflejar la distribución observada de las probabilidades y mejorar la sensibilidad del modelo en la detección de memes ofensivos, priorizando el recall en detrimento de la precisión (puede observarse en la *Figura 12* que algunos memes serían clasificados como ofensivos aunque en realidad no lo son).

```
Probabilidad predicha: 0.052461735904216766, Valor verdadero: 1
Probabilidad predicha: 0.05345312878489494, Valor verdadero: 0
Probabilidad predicha: 0.05435502156615257, Valor verdadero: 1
Probabilidad predicha: 0.054431889206171036, Valor verdadero: 0
Probabilidad predicha: 0.056152015924453735, Valor verdadero: 1
Probabilidad predicha: 0.056733425706624985, Valor verdadero: 1
```

Figura 12: Visualización de probabilidades intermedias predichas por el modelo

Las métricas mencionadas anteriormente que fueron utilizadas para testear el modelo multimodal son:

- **Accuracy:** Es el porcentaje de predicciones correctas entre el total de muestras.
- **Precision:** Mide la proporción de verdaderos positivos (predicciones correctas de ofensivos) entre todos los ejemplos predichos como ofensivos.
- **Recall:** Es la proporción de verdaderos positivos entre todos los ejemplos que son realmente ofensivos.
- **AUC-ROC:** Mide la capacidad del modelo para distinguir entre las clases ofensiva y no ofensiva.

En nuestro caso se prioriza el recall, ya que buscamos que el modelo sea capaz de identificar la mayor cantidad posible de memes ofensivos, minimizando falsos negativos. Y además, se buscó llegar a un accuracy similar al obtenido por otros modelos multimodales.

A pesar de que el accuracy obtenido por nuestro modelo (*Figura 13*) no es demasiado bueno, podemos observar que este superó de manera sutil el de los otros modelos (*Figura 14*) [9], a la vez de la importancia que se le dio al recall, aunque en detrimento de la precisión. Por lo tanto, se detectan mayormente los memes ofensivos pero con el riesgo de clasificar como tal un meme que en realidad no es ofensivo.

Accuracy: 0.653
Precision: 0.6269982238010657
Recall: 0.7204081632653061
AUC-ROC: 0.6543217286914766

Figura 13: Métricas de nuestro modelo multimodal en la fase de testing.

TYPE	MODEL	TEST	
		Acc.	AUROC
	Human	84.70	82.65
Unimodal	Image-grid	52.00	52.63
	Image-region	52.13	55.92
	Text BERT	59.20	65.08
Multimodal <i>(Unimodal pretraining)</i>	Late fusion	59.66	64.75
	Concat BERT	59.13	65.79
	MMBT-grid	60.06	67.92
	MMBT-region	60.23	70.73
	ViLBERT	62.30	70.45
	Visual BERT	63.20	71.33
Multimodal <i>(Multimodal pretraining)</i>	ViLBERT CC	61.10	70.03
	Visual BERT COCO	64.73	71.41

Figura 14: Métricas de otros modelos.

Link al Notebook

El notebook se encuentra en este [enlace](#).

Aplicación

Diseño de Arquitectura

Componentes Principales

a) Extensión de Navegador:

La extensión es el punto de entrada para los usuarios. Permite capturar imágenes (en formato base64) y enviarlas al backend para su procesamiento. Su principal responsabilidad es interactuar con el servidor Flask [2] para enviar las imágenes y recibir los resultados del procesamiento de los memes. Está realizada para Google Chrome para más simplicidad en su desarrollo.

b) Backend Flask:

El servidor Flask [2] actúa como el core de la aplicación, gestionando el procesamiento de las imágenes recibidas desde la extensión. Tiene varias funciones:

- Recepción de imágenes: Flask recibe las imágenes en formato base64 desde la extensión.
- Verificación del hash: El backend genera un hash único para cada imagen, lo que permite identificar si una imagen ha sido procesada previamente.
- Procesamiento de imágenes: Si la imagen es nueva o si se solicita un re-análisis, Flask procesa la imagen. Se utiliza una lógica de procesamiento con el modelo detallado anteriormente para la moderación de memes.
- Almacenamiento en memoria: Las imágenes y los resultados se almacenan en un diccionario en memoria, permitiendo que los resultados estén disponibles para futuras consultas.

c) Interfaz de Gradio:

Gradio [3] proporciona una interfaz gráfica que permite a los usuarios visualizar las imágenes procesadas y sus resultados en forma de galería. La interfaz de Gradio se actualiza mediante solicitudes al backend Flask [2] para recuperar las imágenes y los resultados almacenados. Los usuarios pueden interactuar con la galería y refrescar para ver las imágenes procesadas.

Interacción entre Componentes

Extensión de Navegador - Backend Flask:

La interacción entre la extensión de navegador y el backend Flask [2] comienza cuando el usuario captura una imagen a través de la extensión. La imagen, en formato base64, es enviada al servidor Flask mediante una solicitud HTTP POST dirigida al endpoint [/process-meme](#).

Al recibir la imagen, Flask primero genera un hash único de la imagen para verificar si ya ha sido procesada previamente. Si la imagen ya existe, Flask recupera el resultado almacenado en memoria y lo devuelve a la extensión en formato JSON. Si la imagen no ha sido procesada antes, Flask ejecuta un procesamiento de la imagen. Después del procesamiento, el backend almacena tanto la imagen como el resultado en un diccionario en memoria, utilizando el hash como clave, y devuelve el resultado a la extensión del navegador.

Backend Flask - Gradio:

Gradio [3] interactúa con Flask [2] para obtener y mostrar los resultados de las imágenes procesadas. La interfaz de Gradio está diseñada para actuar como una galería de memes, donde se presentan tanto las imágenes como los resultados del proceso.

Cuando los usuarios desean actualizar la galería, la interfaz envía una solicitud al backend Flask para recuperar la lista de imágenes y sus respectivos resultados, que han sido almacenados en el diccionario en memoria. Flask responde con la información más reciente de las imágenes procesadas y sus resultados, que son presentados visualmente en la interfaz de Gradio. Los usuarios pueden refrescar la galería para obtener los datos más recientes y visualizar las imágenes.

Modelado UML

Se realizaron algunos diagramas para representar los componentes principales de la aplicación y su comportamiento.

Diagrama de componentes

En la *Figura 15* se ilustra la interacción entre los tres módulos principales explicados en la sección anterior del presente documento.

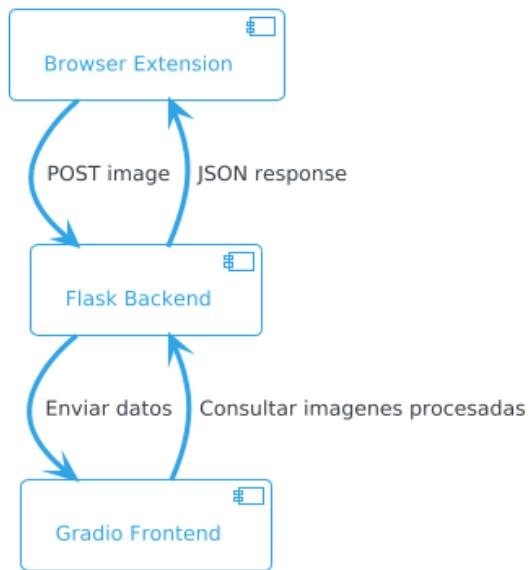


Figura 15: Diagrama de componentes del sistema detector de memes ofensivos

Diagrama de secuencia

En la *Figura 16* se muestra el flujo de procesamiento cuando la extensión de navegador envía una imagen al backend. Primero, el backend verifica si la imagen ya fue procesada; de no ser así, procede a analizarla y clasificarla como ofensiva o no ofensiva. Luego, el backend recupera y envía las imágenes procesadas y sus resultados al frontend de Gradio [3] para ser visualizados en una galería.

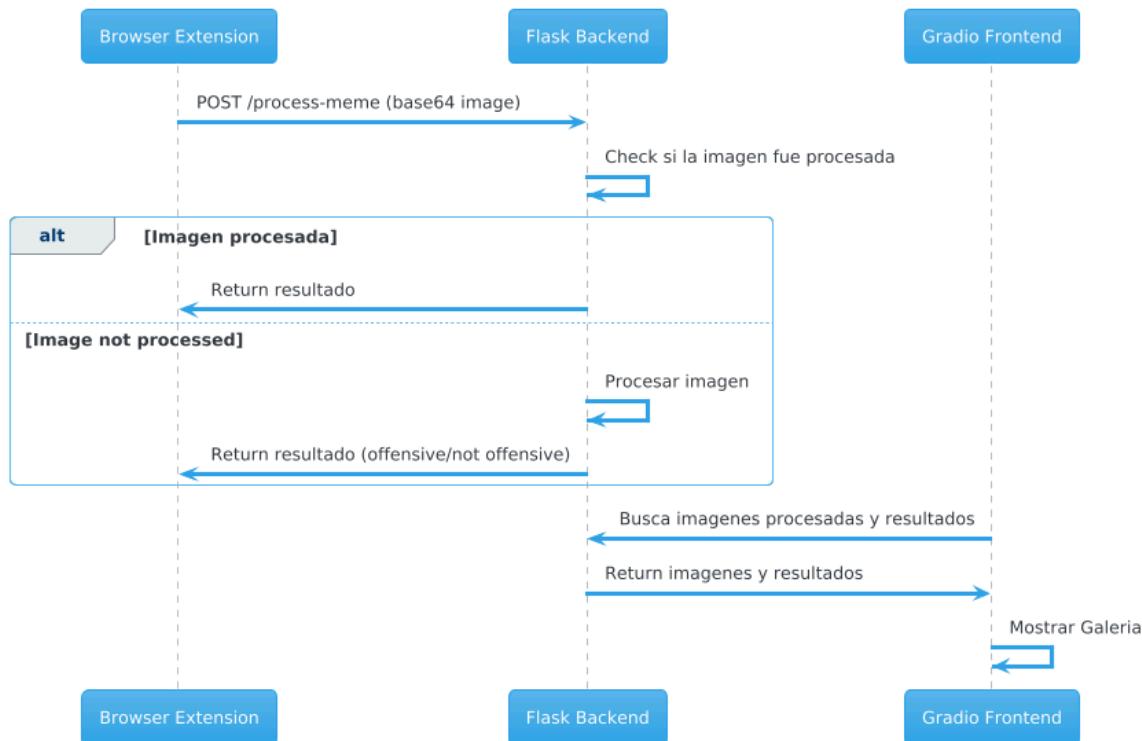


Figura 16: Diagrama de secuencia del sistema detector de memes ofensivos

Implementación

Capturas de pantalla del sistema

La Figura 17 muestra la extensión para Google Chrome diseñada para capturar la imagen de un meme y realizar su análisis.

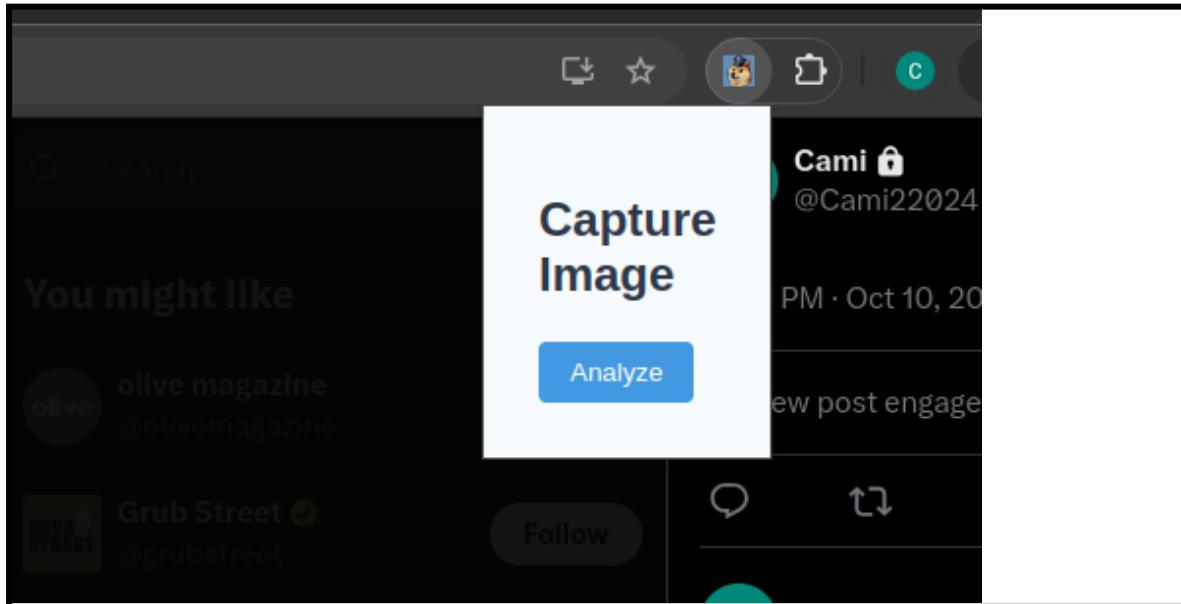


Figura 17: Extensión para Google Chrome

La Figura 18 presenta la notificación generada tras el análisis del meme, indicando en este caso que se trata de un meme ofensivo.



Figura 18: Notificación de meme clasificado como ofensivo

La *Figura 19* presenta la notificación generada tras el análisis del meme, indicando en este caso que se trata de un meme no ofensivo.

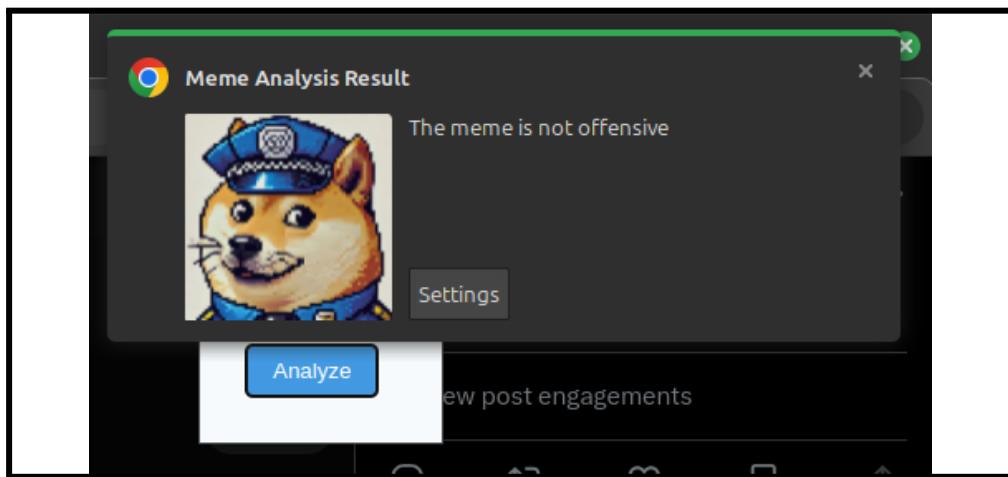


Figura 19: Notificación de meme clasificado como no ofensivo

La *Figura 20* muestra el caso en el que se intenta analizar un meme que ya ha sido previamente procesado, brindando la opción de realizar el análisis nuevamente.

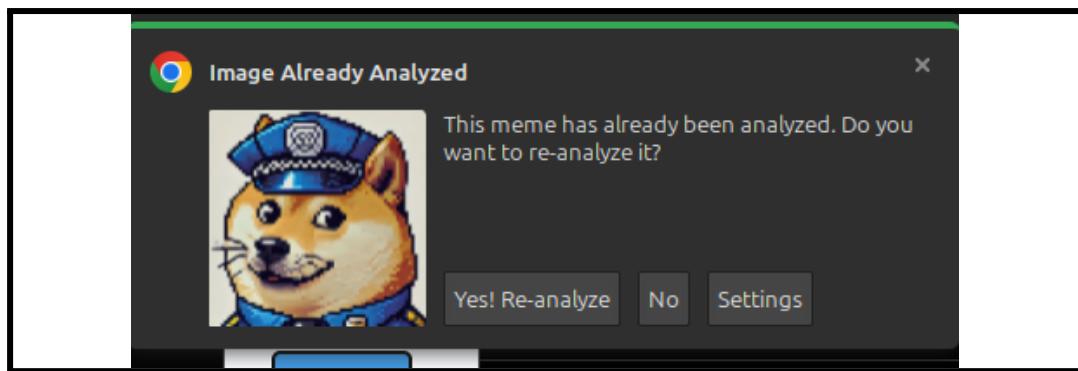


Figura 20: Notificación de meme ya analizado anteriormente

La Figura 21 muestra la galería con todas las imágenes que han sido analizadas, junto con el resultado correspondiente a cada una de ellas.

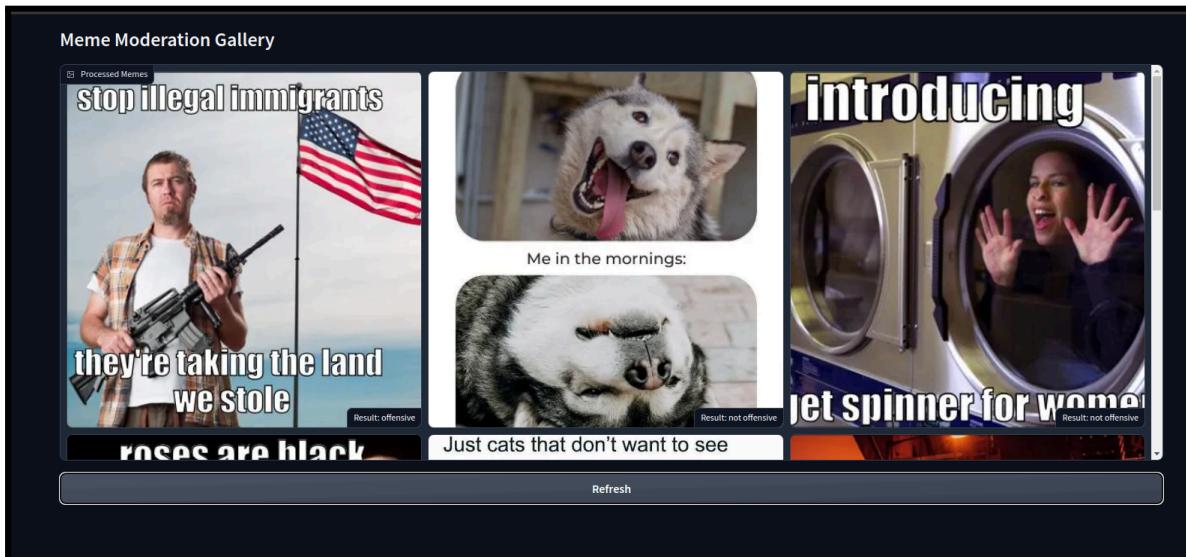


Figura 21: Galería de imágenes de Gradio

Presentación final

Hoja de Ruta



Figura 22: Hoja de ruta de la presentación

Capturas de la Presentación Final



Figura 23: Título del Proyecto.



Figura 24: Miembros del Equipo.



Figura 25: Temas de la presentación.



Figura 26: Dominio elegido.



Figura 27: Definición de meme.



Figura 28: Contexto actual y Enfoque del proyecto.



Figura 29: Importancia del proyecto.



Figura 30: Problema.



Figura 31: Desafíos en la detección de memes ofensivos.



Figura 32: Solución propuesta.

*Figura 33: Aspectos de la solución propuesta.**Figura 34: Casos de uso identificados.*

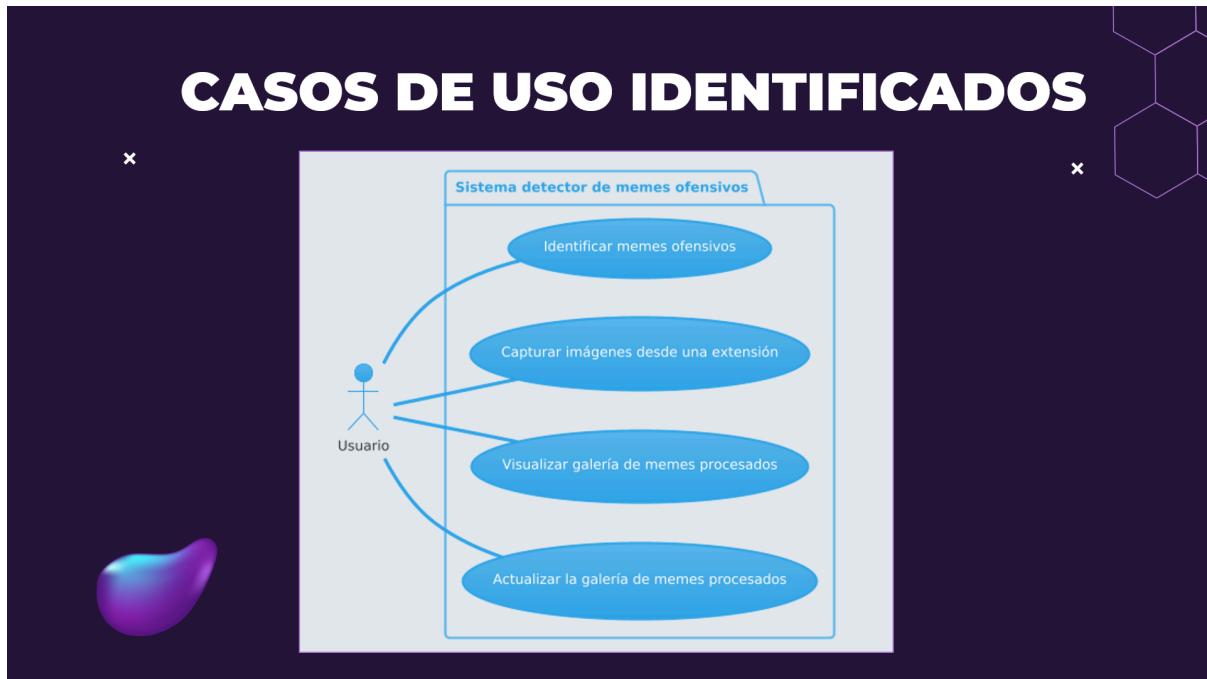


Figura 35: Diagrama de casos de uso.

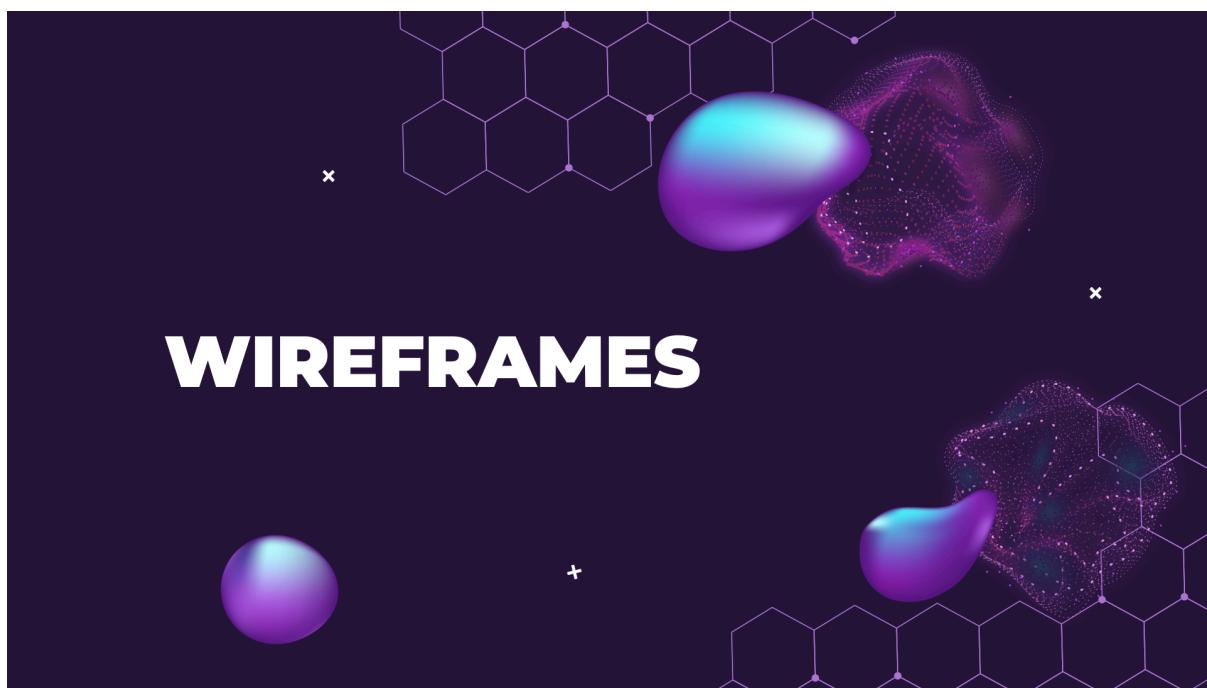


Figura 36: Wireframes.



Figura 37: Wireframe de la extensión.



Figura 38: Wireframes de los tipos de notificaciones.



Figura 39: Wireframe de la web de visualización de imágenes.



Figura 40: Tareas identificadas.



Figura 41: Estado del Arte.

Multimodal Meme Dataset (MultiOFF) for Identifying Offensive Content in Image and Text

DATASET

MultiOFF

MODELO MULTIMODAL

- Stacked LSTM + VGG16
- BiLSTM + VGG16
- CNN para Texto + VGG16
- LR, NB Y DNN (Unimodal)

RESULTADOS

CNN + VGG16 ofreció la mejor capacidad para recuperar memes ofensivos

Figura 42: Primer paper.

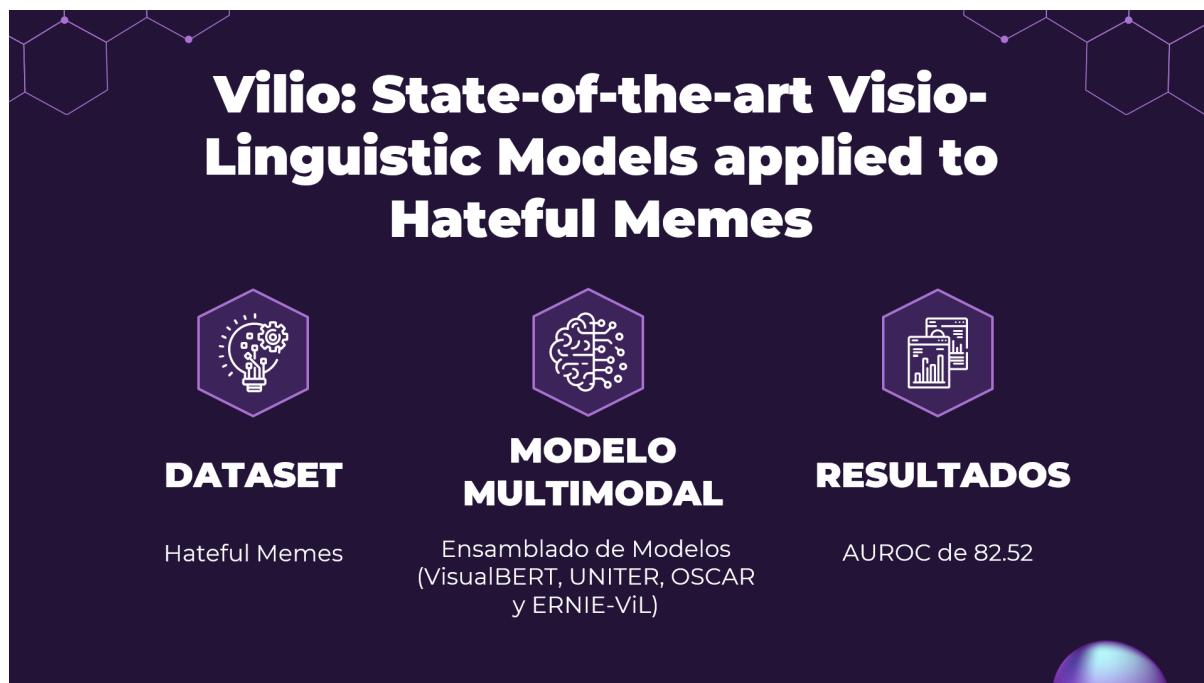


Figura 43: Segundo paper.

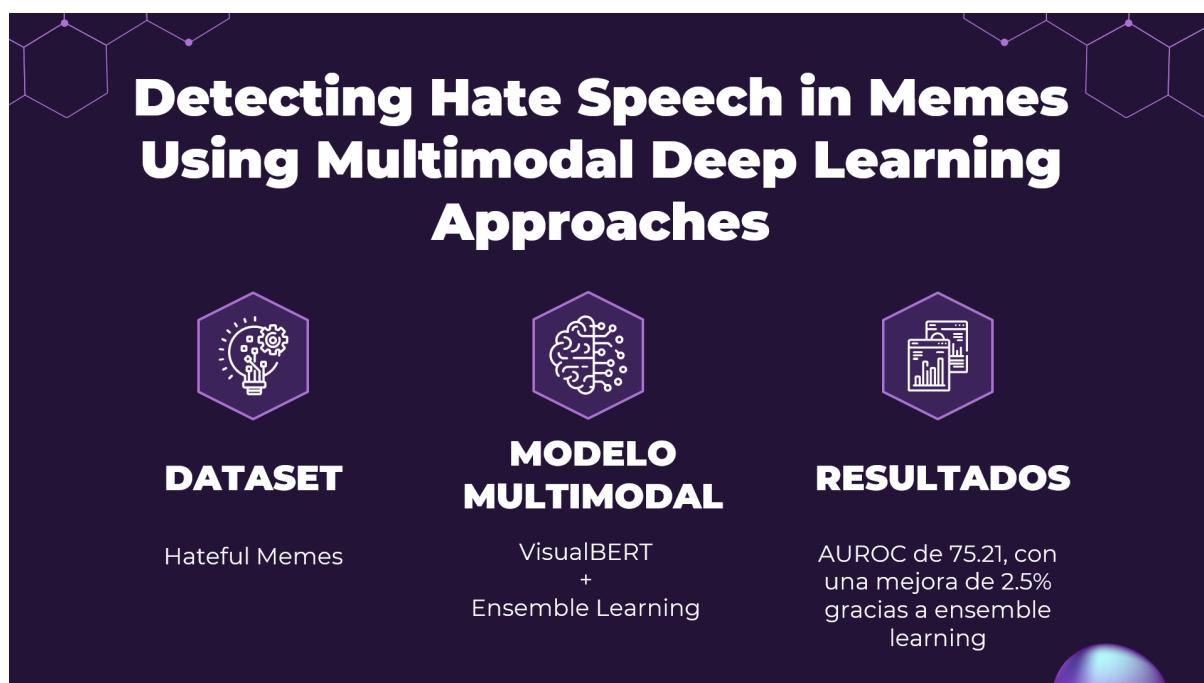


Figura 44: Tercer paper.

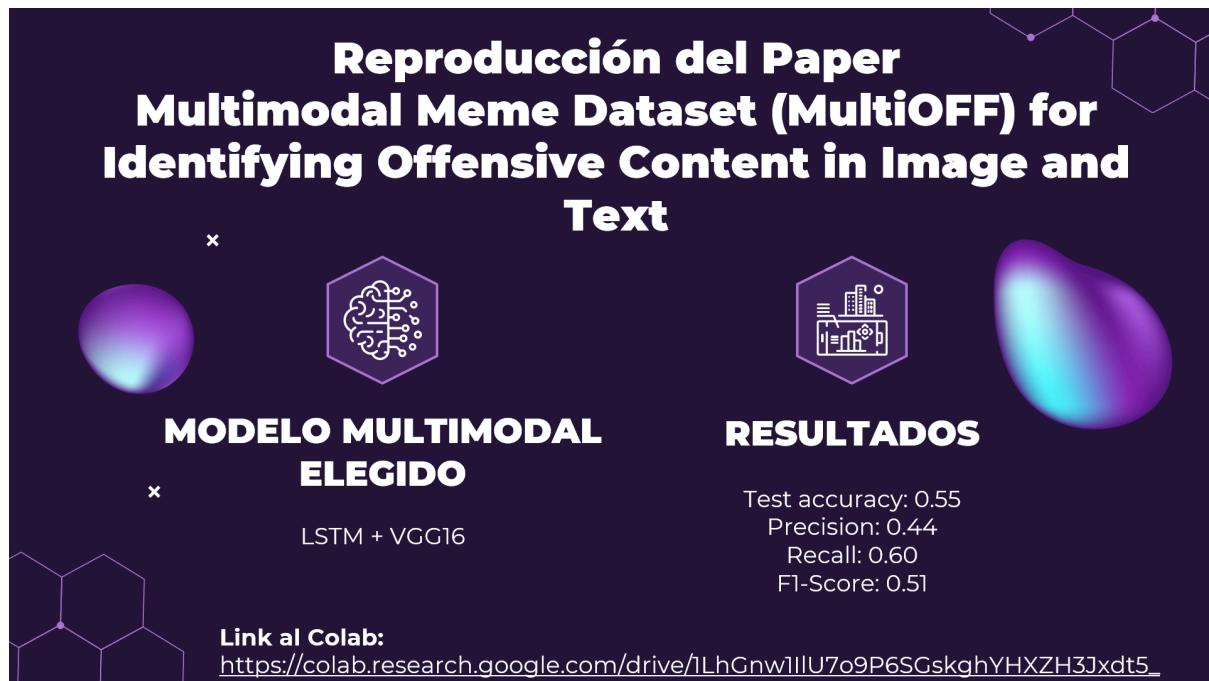


Figura 45: Reproducción del primer paper.



Figura 46: Prueba de concepto.



Figura 47: Esquema de Pipeline.



Figura 48: Modelos elegidos.

*Figura 49: Dataset.**Figura 50: Training Loss y Validation Loss en epochs.*

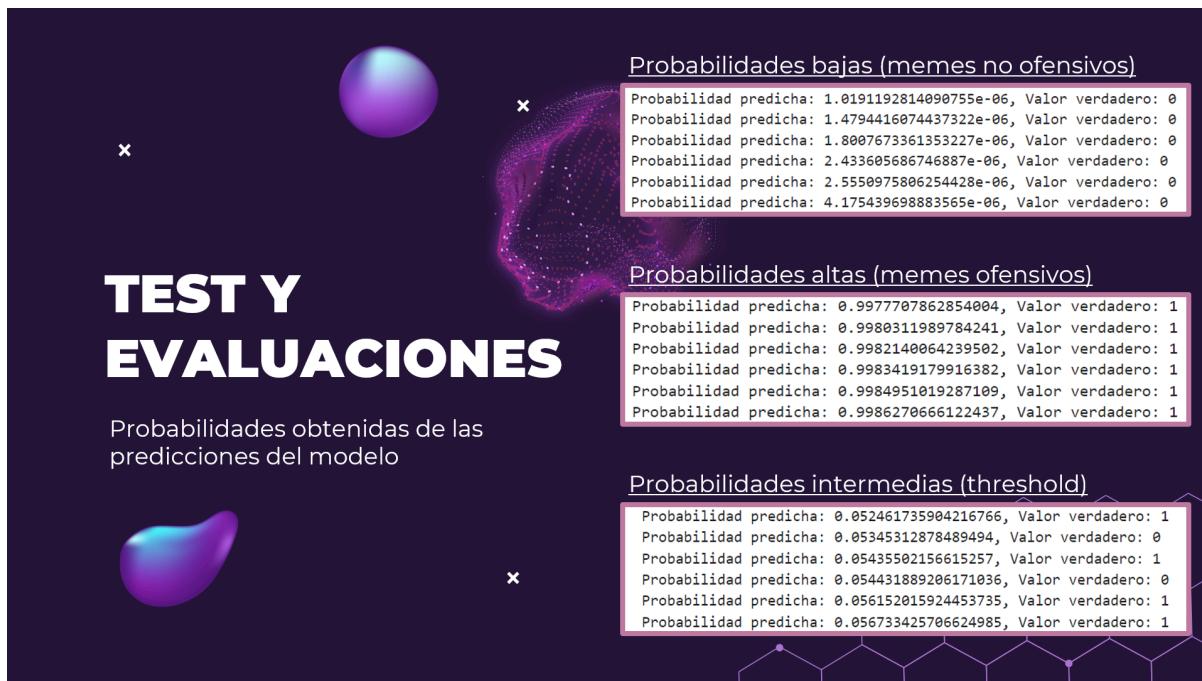


Figura 51: Probabilidades obtenidas de las predicciones del modelo.

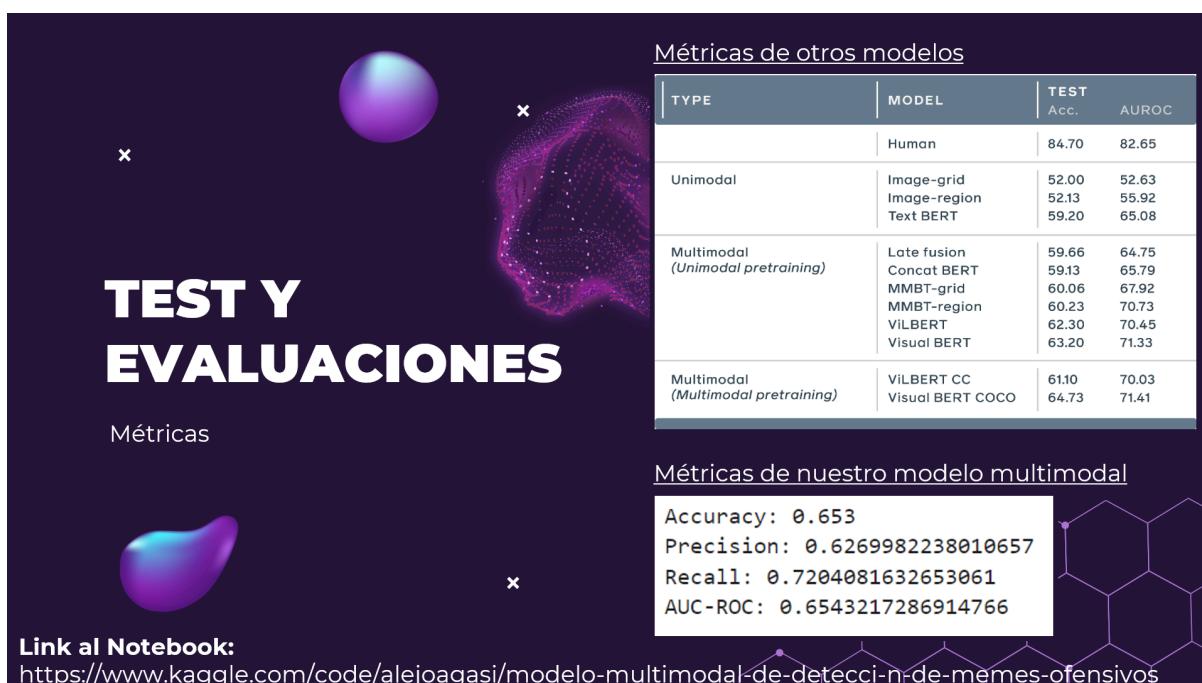


Figura 52: Métricas.



Figura 53: Aplicación.



Figura 54: Objetivo de la aplicación.

Proveer una solución que permita capturar memes, analizarlos automáticamente y mostrar los resultados de forma intuitiva para los usuarios, utilizando una combinación de una extensión de navegador, un backend Flask y una interfaz de Gradio.

*Figura 55: Componentes principales.**Figura 56: Componente - Extensión.*

COMPONENTES PRINCIPALES

* Backend Flask

Actúa como el núcleo de la aplicación, gestionando el procesamiento de las imágenes.

- ◆ Recibe las imágenes desde la extensión.
- ◆ Genera un hash único para cada imagen para identificar si ha sido procesada previamente.
- ◆ Si la imagen es nueva o si se solicita un re-análisis, procesa la imagen usando el modelo.
- ◆ Almacena las imágenes y sus resultados en un diccionario para futuras consultas.

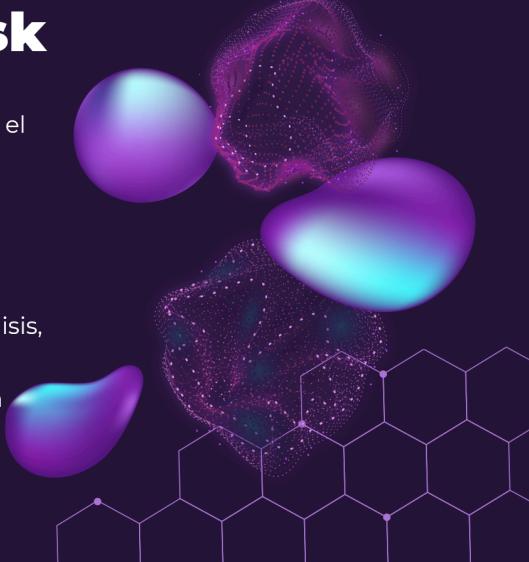


Figura 57: Componente - Backend Flask.

COMPONENTES PRINCIPALES

* Interfaz de Gradio

Proporciona una interfaz gráfica para visualizar las imágenes procesadas y sus resultados.

- ◆ Muestra las imágenes y sus resultados en forma de galería.
- ◆ Permite a los usuarios actualizar la galería para ver los resultados más recientes.



Figura 58: Componente - Interfaz de Gradio.

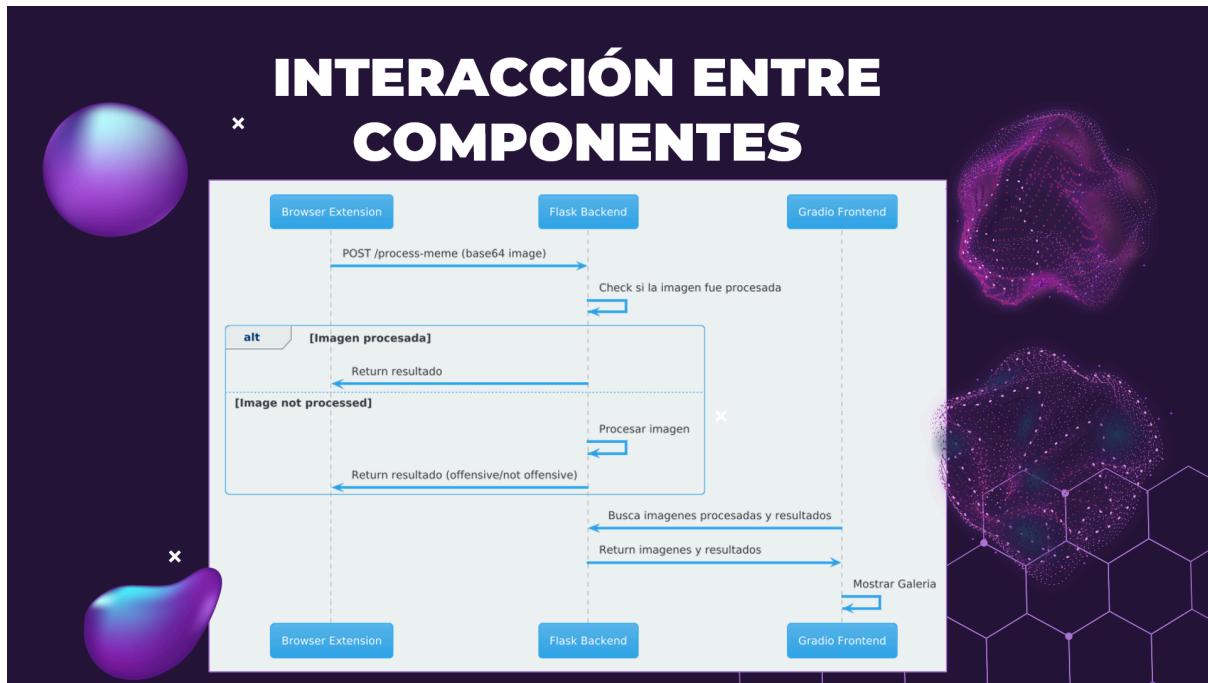


Figura 59: Interacción entre componentes.

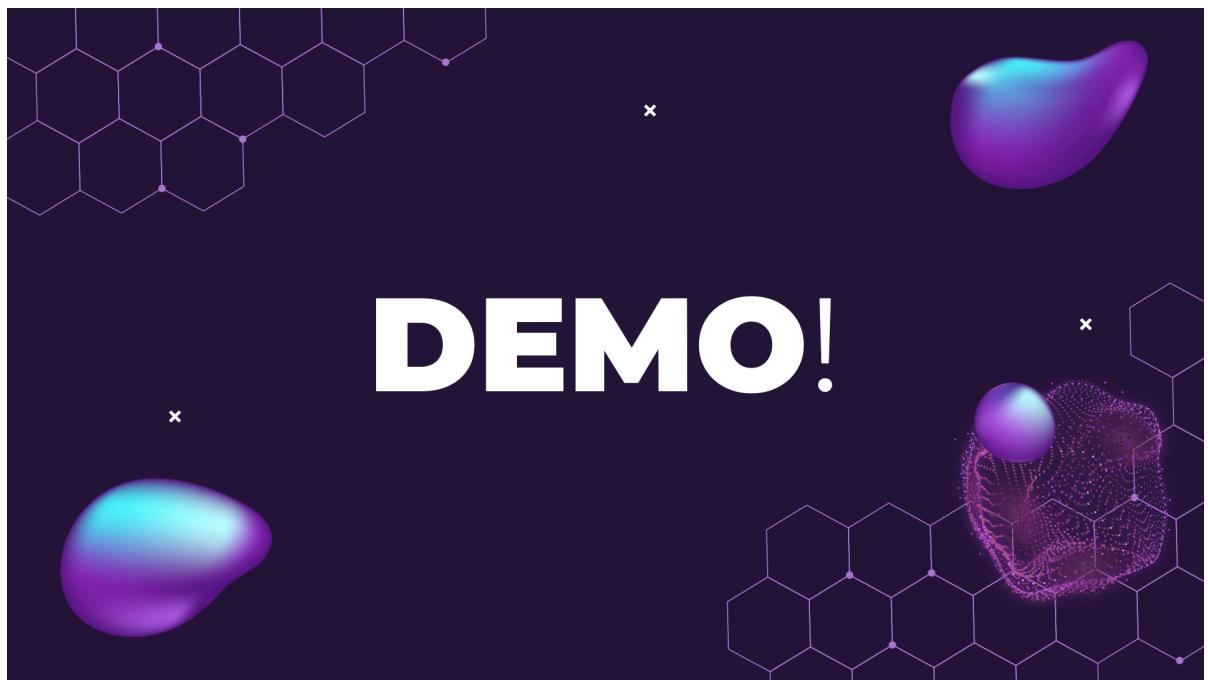


Figura 60: Demo.



Figura 61: Agradecimiento.

Link a Video de Presentación

El video de presentación se encuentra en este [enlace](#).

Referencias

- [1] DrivenData. Benchmarking Hateful Memes: A Data Science Perspective. Disponible en <https://drivendata.co/blog/hateful-memes-benchmark/>. Accedido en octubre 2024.
- [2] Flask. A minimal application. Disponible en: <https://flask.palletsprojects.com/en/3.0.x/quickstart/#a-minimal-application>. Accedido en octubre 2024.
- [3] Gradio. Documentation. Disponible en: <https://www.gradio.app/docs>. Accedido en octubre 2024.
- [4] Papers with Code. Multimodal Meme Dataset (MultiOFF) for Identifying Offensive Content in Image and Text. Disponible en <https://paperswithcode.com/paper/multimodal-meme-dataset-multioff-for>. Accedido en octubre 2024.
- [5] Papers with Code. Vilio: State-of-the-Art Visio-Linguistic Models Applied to Hateful Memes. Disponible en <https://paperswithcode.com/paper/vilio-state-of-the-art-visio-linguistic>. Accedido en octubre 2024.
- [6] Papers with Code. Detecting Hate Speech in Memes Using Multimodal Deep Learning Approaches. Disponible en <https://paperswithcode.com/paper/detecting-hate-speech-in-memes-using>. Accedido en octubre 2024.
- [7] Neptune AI. How to Code BERT Using PyTorch: A Tutorial. Disponible en <https://neptune.ai/blog/how-to-code-bert-using-pytorch-tutorial>. Accedido en octubre 2024.
- [8] Wikipedia. Red Neuronal Residual. Disponible en https://es.wikipedia.org/wiki/Red_neuronal_residual. Accedido en octubre 2024.
- [9] Meta AI. Hateful Memes Challenge and Dataset. Disponible en <https://ai.meta.com/blog/hateful-memes-challenge-and-data-set/>. Accedido en octubre 2024.
- [10] Kaggle. Hateful Memes Dataset by William Berrios. Disponible en <https://www.kaggle.com/datasets/williamberrios/hateful-memes>. Accedido en octubre 2024.