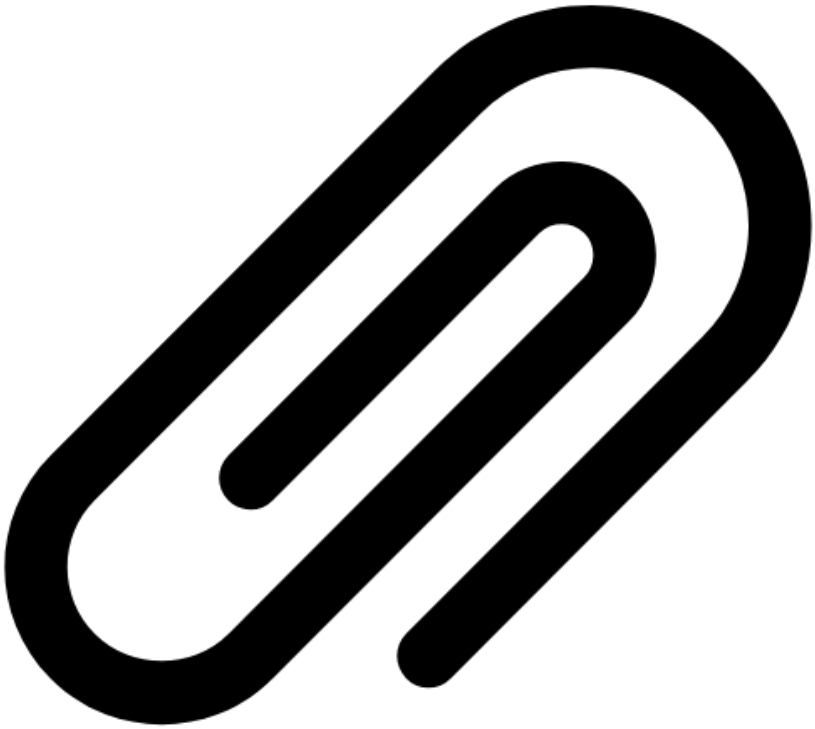
28 DE OCTUBRE DE 2025



CLIP

SANTIAGO SOUTO ORTEGA
UIE

Contenido

Introducción

Esta práctica tiene como objetivo comparar las diferencias entre crear embeddings con un modelo del lenguaje y una ResNet y el modelo de OpenAI Clip. Para ello, se divide la práctica en dos partes. Primero una parte denomida pre-clip y una segunda llamanda clip. Con ello, abordaremos el problema de relacionar la información visual con la escrita.

Para el enfoque Pre-Clip, se utilizó una ResNet-50 debido a que es famosa en el ámbito del reconocimiento de imágenes debido a su arquitectura de aprendizaje residual que evita el problema del *vanishing gradient*. Además, se utilizó Sentence Transformers para procesar el tecto.

Por otro lado, se utilizó el modelo ya preentrenado de Clip para hacer los embeddings tanto de las imágenes como del texto.

Dataset

Este dataset contiene 20 fotografías repartidas de forma equitativa en 4 categorías diferentes: nubes, bebidas, comida, aviones y coches de competición. Estas categorías han sido elegidas para que el dataset fuera diverso. Esta decisión permite evaluar tanto la capacidad de los modelos para distinguir entre diferentes tipos de categorías como su habilidad para identificar similitudes y diferencias dentro de la misma.

Además, se incorpora un json donde cada categoría agrupa las diferetes fotos con los siguientes campos: filename y description. El fomato de archivo json fue elegido debido a la facilidad de trabajar y recuperar información desde python con un simple bucle dentro del script.

Las imágenes, fueron elegidas para que no fueran muy pesadas pero que tampoco su baja resolución impidiese que se viera de forma correcta.

Por último, la organización de la carpeta donde se almacenan está pensada de la siguiente manera. Una carpeta con cada una de las clases con sus respectivas imágenes y un json con las descripciones.

Pre-Clip

Arquitectura y componentes

La implementación de Pre-Clip viene de la mano de una clase ya que es la manera más cómoda de desarrollar módulo a módulo y al finalizar llamar a un único método que corra el programa.

Para el procesamiento de imágenes se ha usado una ResNet-50, una red convolucional profunda que ha demostrado ser muy exitosa en tareas de visión por computador. En este caso, ha sido escogido un modelo ya preentrenado del dataset de ImageNet, por lo que en este caso el programa solo está ejecutando la inferencia, no se necesita de más entrenamiento. Sin embargo, para utilizar este modelo para extraer embeddings, es necesario modificar la arquitectura eliminando la capa de clasificación final. Permitiéndonos acceder a la representación interna dentro de la red y no conocer que objeto está presente dentro de la imagen. En este caso, la última capa tiene 2048 unidades en la ultima capa (He) por lo que esta será la dimensión de nuestros embeddings.

Para que la red pueda trabajar con estas imágenes, es necesario pasarlas a imágenes de 256*256px, realizar un corte central de 224*224px, convertir los valores de píxeles a tensores, y finalmente normalizar estos valores utilizando la media y desviación estándar específicas de ImageNet. Esto es clave ya que el modelo fue entrenado para recibir estos inputs.

```
self.preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    ),
])
```

Este es el código correspondiente a este preprocesado que es necesario aplicar a cada una de las imágenes.

Para el modelo de texto, se utilizó *DistilUse-base-multilingual-cased* de la librería transformers debido a que viene preentrenado a la par de una baja utilización de recursos. Además, según el propio Hugging face tiene más de dos millones de descargas solo en el último mes.

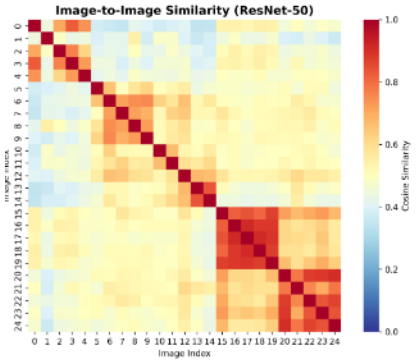
Este modelo fue entrenado especificamente para generar representaciones semánticas de oraciones. A diferencia de otros, este entiende el contexto y la relaciones entre palabras produciendo embeddings de 512 dimensiones.

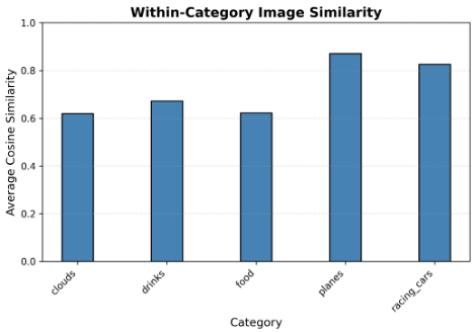
Incompatibilidad de dimensiones

El problema de usar un enfoque con dos modelos diferentes es que las dimensionesd los embeddings son muy diferentes (2048 y 512) por lo que es imposible compararlas.

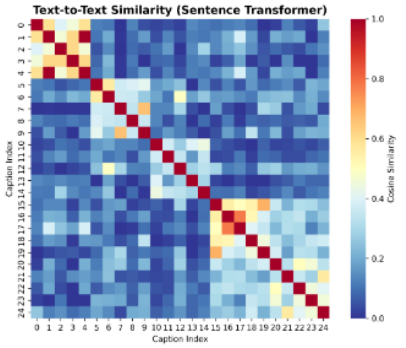
Por lo tanto, a la hora de analizar los resultados, fue necesario centrarse en estudiar la estructura interna de cada modalidad por separado en vez de emparejar directamente las imágenes con sus descripciones.

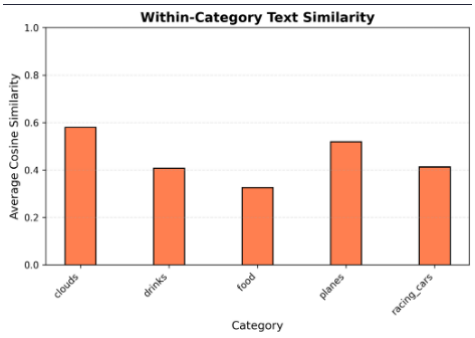
Para las imágenes, calculé la matriz de similitud entre imagenes utilizando la similitiud del coseno como medida entre todos los pares posibles de embeddings. Viendo el resultado, nos contramos que las imágnees de una misma categoría tienden a mostrar valroes de similitud más altos entre sí, lo cual es esperado dado que comparte caraterísticas visuales comunes. Sin embago, tambien se pueden aprenciar carateticas similares entre imagenes de diferentes categorías. El motivo, es que todas la imagenes pueden contener colores o ciertas estrucutras en el fondo que se asemejen.





Por otro lado, para la comparación de las descripciones, sigue la misma dinámica que la anterior, siendo capaz de captar mas similiritudes entre textos de la misma categria pero de forma mas leve que en el caso de las imagnes. Segurmaenteg una descripción mas precisa ayudaría a que el modelo fuera capaz de discernir mejor las diferencias.





Clip

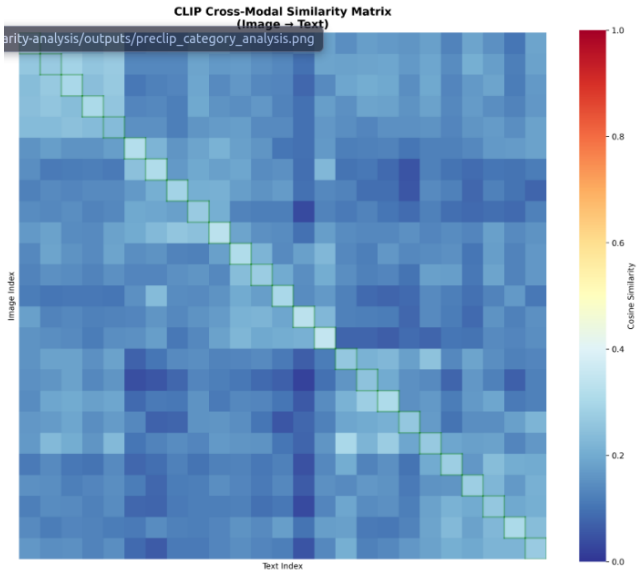
Implementación

El código de CLIP implementa una clase que utiliza el modelo preentrenado openai/clip-vit-base-patch32 para analizar imágenes y texto. Tras cargar las imágenes y sus descripciones desde un archivo JSON, extrae embeddings de 512 dimensiones tanto para las imágenes (usando Vision Transformer) como para los textos (usando Transformer). La función compute_cross_modal_similarity() calcula una matriz de similitud coseno entre todas las imágenes y textos. El análisis evalúa la precisión del emparejamiento imagen-texto encontrando para cada imagen qué descripción tiene mayor similitud y verificando si coincide con la correcta.

Resultados

Los resultados con CLIP fueron mejores que con el enfoque Pre-CLIP. El modelo identificó correctamente la mayoría de las correspondencias entre imágenes y descripciones. Esto demuestra que un modelo entrenado específicamente para relacionar imágenes y texto funciona mejor para esta tarea.

El análisis de la matriz de similitud reveló patrones claros. Los valores situados en la diagonal de la matriz, que representan los pares correctos de imagen-descripción, mostraron valores superiores a aquellos fuera de la diagonal. Los valores de similitud correspondientes a pares correctos se ubicaron típicamente en un rango alto, mientras que los pares incorrectos presentaron similitudes notablemente más bajas.



Referencias

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep residual learning for image recognition* [Preprint]. arXiv. https://arxiv.org/abs/1512.03385