Universidad de La Habana
Facultad de Matemática y Computación

Recuperación de imágenes usando incrustaciones multimodales

Autor:
Raúl Beltrán Gómez

Tutores:
Dr. Yudivian Almeida Cruz

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

Enero de 2024

https://github.com/rb58853/ML-RSI-Images

Dedicación

Agradecimientos

Agradecimientos

Opinión del tutor

Opiniones de los tutores

Resumen

El tema de la recuperación precisa de imágenes representa un desafío constante en el campo de la ciencia de la computación. En esta tesis, se propone un enfoque innovador para abordar dicho problema, aplicando los modelos de Segmentación *Segment Anything* (SAM) y el modelo *Contrastive Language–Image Pretraining* (CLIP) para la generación de embeddings. En este estudio, se explora la recuperación de imágenes mediante consultas precisas, empleando el modelo *Contrastive Language-Image Pretraining* y la generación de embeddings multimodales con el mismo. Se pone especial énfasis en la recuperación de imágenes basada en la posición, procesando tanto texto como imágenes para extraer características con este fin.

Abstract

The topic of precise image recovery presents a constant challenge in the field of computer science. In this thesis, an innovative approach is proposed to tackle this issue, utilizing the Segmentation models *Segment Anything* (SAM) and the *Contrastive Language–Image Pretraining* (CLIP) model for generating embeddings. In this study, the recovery of images through accurate queries is explored, using the *Contrastive Language-Image Pretraining* model and multimodal embedding generation. A special emphasis is placed on position-based image recovery, processing both text and images to extract features for this purpose.

Índice general

Introducción

En el pasado, el uso de la inteligencia artificial estaba restringido y se empleaba principalmente en casos de uso específicos. Las entidades que la utilizaban solían estar familiarizadas con este campo y tenían objetivos bien definidos.

En la actualidad se ha logrado un avanze considerable en este campo, obteniendo resultados que hace años parecían poco probables. Cada vez más personas están comenzando a aprovechar estos beneficios, y la tecnología está cambiando rápidamente, con la inteligencia artificial siendo el centro de todo. Si bien antes esta tecnología era menos utilizada, el lanzamiento de nuevos modelos de lenguaje accesibles para todos, como GPT[19], ha despertado el interés y la adopción de la inteligencia artificial por parte de un público más amplio.

Es innegable que la interacción entre los seres humanos y las máquinas está experimentando cambios significativos. Cada vez se les encomiendan más tareas que antes eran exclusivas de las personas, como la traducción, el diseño de imágenes e incluso la generación de código, que ahora son abordadas por la inteligencia artificial, al menos hasta cierto grado de correctitud.

Como parte de este avance, el campo de la visión artificial también ha evolucionado notablemente. La visión artificial permite a las computadoras y sistemas extraer información relevante de imágenes digitales, videos y otras entradas visuales. Gracias a esta capacidad, dichos sistemas pueden tomar medidas o realizar recomendaciones basadas en dicha información. Podríamos decir que si la inteligencia artificial permite a las computadoras pensar, la visión artificial les permite ver, observar y comprender.

El impresionante progreso del aprendizaje automático en los últimos años, especialmente el aprendizaje profundo (*Deep Learning*), ha revolucionado el campo de la visión artificial, posibilitando nuevas aplicaciones que antes parecían inimaginables. Desde diagnósticos de imágenes en el campo de la medicina, la automatización de vehículos, el reconocimiento de objetos y la segmentación de imágenes, entre otros.

La visión artificial requiere grandes cantidades de datos para aprender y descubrir patrones. Necesita una exposición extensa a un contenido para adquirir conocimientos sobre él. La era de la información en la que vivimos actualmente, donde abundan los datos, es el entorno perfecto para que estos algoritmos de aprendizaje se desarrollen. La combinación de este acceso a conjuntos de datos masivos con las nuevas arqui-

tecturas de aprendizaje profundo ha dado lugar al surgimiento de modelos de visión altamente capacitados. Muchos de los modelos de visión artificial actuales han sido entrenados con cientos de millones de imágenes.

Si bien los primeros modelos de visión se especializaban en clasificar objetos específicos para determinar su presencia en la imagen, con el lanzamiento de la nueva arquitectura de procesamiento del lenguaje, conocida como *transformers*[17], en el año 2017, se ha logrado una integración de las tareas de visión artificial y procesamiento del lenguaje natural, lo cual ha arrojado resultados impresionantes. Un ejemplo de ello es el modelo CLIP[21], entrenado con 400 millones de imágenes y texto proveniente de Internet, lo que le permite comprender la similitud existente entre textos e imágenes.

0.1. Motivación

El campo de la recuperación de imágenes a través de características como los embeddings ha experimentado un notable avance en los últimos años. Con la aparición de modelos multimodales que comprenden tanto texto como imágenes dentro del mismo espacio, la creación de sistemas de recuperación que utilizan estas características se presenta como una opción muy útil e innovadora. Estos sistemas de recuperación, que emplean técnicas de búsqueda vectorial en lugar de búsqueda basada en palabras clave, buscan grandes colecciones de vectores en un espacio dimensional alto para encontrar vectores similares a una consulta determinada. Este enfoque suele ser más eficaz que las técnicas tradicionales de recuperación de imágenes, ya que puede reducir el espacio de búsqueda y mejorar la precisión de los resultados.

Además, la recuperación de imágenes mediante incrustaciones multimodales permite convertir los datos de una imagen en un vector, lo que facilita la comparación y la recuperación de imágenes similares.

0.2. Antecedentes

El campo de la visión artificial ha experimentado un continuo progreso y expansión, dando lugar a diversas arquitecturas y modelos que integran la comprensión de lenguaje y visión. Entre los ejemplos destacados se encuentran CLIP, BLIP[9], LLaVA[12] y GPT-4V[22].

CLIP (*Contrastive Language-Image Pretraining*)[21] es un modelo desarrollado y publicado por *OpenAI* en el año 2021. Fue concebido con el propósito de comprender y abordar tareas de visión y lenguaje de manera unificada, permitiendo establecer conexiones entre texto e imágenes.

BLIP (*Bootstrapping Language-Image Pre-training*)[9], por su parte, es otro modelo de preentrenamiento de visión y lenguaje desarrollado por Salesforce Research. Hizo su debut en el año 2022 y, al igual que CLIP, tiene como objetivo comprender y generar tareas de visión y lenguaje de manera conjunta, siendo capaz de generar descripciones precisas de imágenes.

LLaVA (*Large Language-and-Vision Assistant*)[12] es un modelo multimodal de gran escala que combina un codificador de visión con un modelo de lenguaje avanzado para el entendimiento general de contenido visual y lingüístico. Fue presentado por un equipo de investigación de Microsoft en colaboración con la Universidad de Columbia y la Universidad de Wisconsin-Madison en abril del año 2023.

GPT-4V[22], o Modelo de Visión de GPT-4[16], es una extensión del popular modelo de lenguaje GPT-4[16] desarrollado por *OpenAI*. GPT-4V posee la capacidad de comprender imágenes y vincularlas con el modelo de lenguaje de GPT-4, lo que permite obtener resultados altamente precisos en tareas relacionadas con visión y lenguaje. Este modelo fue publicado en marzo del año 2023, aunque su componente de visión no estuvo disponible para el público hasta octubre del mismo año.

0.3. Problemática

Aunque estos modelos de visión y lenguaje poseen una notable capacidad para analizar imágenes en relación con el texto, actualmente se han enfocado en tareas específicas que difieren de la recuperación de información. Sin embargo, han mostrado resultados satisfactorios que pueden sentar las bases para abordar de manera efectiva el campo de la recuperación de imágenes.

Recientemente, el enfoque se ha desplazado hacia la recuperación de imágenes basada en características, con un énfasis particular en el uso de embeddings. Los embeddings son representaciones vectoriales que capturan la esencia de los datos de entrada. En el contexto de la recuperación de imágenes, los embeddings pueden codificar características visuales y contextuales de las imágenes, lo que las hace adecuadas para tareas de recuperación.

En lugar de centrarse en el etiquetado manual de imágenes, se están dedicando esfuerzos para desarrollar sistemas que utilicen embeddings y características para recuperar imágenes. Esta estrategia se centra en la recuperación de las propias imágenes, en lugar de obtener información relacionada con las imágenes.

El objetivo es desarrollar un sistema de recuperación de imágenes que pueda generar y utilizar eficazmente los embeddings para recuperar imágenes a partir de descripciones detalladas y precisas.

0.4. Objetivos

0.4.1. Objetivo general

El objetivo de este trabajo consiste en desarrollar un sistema automatizado extracción de características de imágenes para procesarlas, y un sistema de recuperación altamente preciso que utilice dichas caracteríticas como información para crear el proceso de recuperación, empleando modelos de aprendizaje automático. El propósito es lograr la recuperación de la imagen más adecuada mediante consultas que cuenten con descripciones sumamente precisas. Se prestará una atención especial atención a la descripcion de posiciones de los objetos.

0.4.2. Objetivos específicos

- Extraer incrustaciones (*embeddings*) de las imágenes utilizando modelos de aprendizaje de máquinas previamente entrenados.
- Emplear modelos de aprendizaje automático entrenados con extensas cantidades de datos especializados en la descripción de imágenes como una variante de de características de la imagen.
- Diseñar una arquitectura escalable que permita la incorporación de nuevos modelos de visión artificial a medida que este campo se expande con el tiempo.
- Utilizar modelos de segmentación de imágenes para obtener descripciones más detalladas, analizando la imagen no solo en su totalidad, sino también por segmentos.
- Procesar las descripciones finales proporcionadas para crear un sistema de tokens que se ajuste al formato de consultas más esperadas, con el objetivo de lograr una recuperación precisa de la información.
- Desarrollar un sistema óptimo de recuperación de información para recuperar las imágenes almacenadas en la base de datos correspondiente a las imágenes procesadas.

0.5. Organización

El resto del documento se encuentra organizado de la siguiente manera. En el capítulo 1 se realiza el análisis de una serie de modelos, arquitecturas y trabajos anteriores relacionados con la generación de texto a partir de imágenes. Además, se

exploran técnicas de recuperación de información con potencial para la extracción de imágenes. Este capítulo constituye el estado del arte en el campo.

El capítulo 2 se dedica a explicar la propuesta de solución, incluyendo la justificación de la elección de los modelos y la arquitectura final. También se detallan las propuestas relacionadas con el modelo de recuperación de información.

El capítulo 3 sección 1, se enfoca en la comparación de diversas soluciones, variando los modelos utilizados en cada una de ellas, así como los hiperparámetros modificados.

En el capítulo 3, seccion 2, se recopilan los detalles de la implementación y se abordan los desafíos surgidos debido a la limitación de hardware y acceso a información.

Finalmente, en el capítulo 6 se presentan las conclusiones derivadas de la investigación llevada a cabo.

Capítulo 1

Estado del Arte

1.1. Investigaciones basadas en uso de CLIP

La recuperación de imágenes ha experimentado avances significativos en los últimos años, propulsados por el desarrollo de tecnologías de aprendizaje profundo. Un aspecto clave en este campo es la integración de la comprensión del lenguaje y las imágenes. El modelo CLIP (*Contrastive Language–Image Pretraining*)[21] es un ejemplo destacado de esto, ya que permite a las máquinas aprender a entender tanto el lenguaje como las imágenes. Este modelo ha demostrado ser eficaz en diversas tareas, incluyendo la detección de objetos en imágenes y la recuperación de imágenes basada en descripciones textuales.

El modelo CLIP[14] se presenta en el artículo *CLIP: Connecting text and images*[14] de *OpenAI*, donde se introduce una red neuronal que aprende visualmente conceptos a partir de supervisión de lenguaje natural. CLIP[14] puede ser aplicado a cualquier imagen o texto, demostrando una capacidad similar a las capacidades *zero-shot* de GPT-2[19] y GPT-3[18].

Uno de los principales desafíos en la recuperación de imágenes es la falta de correspondencia exacta entre las descripciones textuales y las imágenes. Para abordar este problema, se han desarrollado técnicas de recuperación de imágenes basadas en características, como la utilizada en el artículo *Effective Conditioned and Composed Image Retrieval: Combining CLIP-Based Features*[2]. Este enfoque utiliza las características derivadas de CLIP[14] para mejorar la precisión de la recuperación de imágenes.

En el artículo *Text-to-Image and Image-to-Image Search Using CLIP*[10], Zoumana Keita presenta una técnica innovadora para la recuperación de imágenes que utiliza CLIP. En lugar de un enfoque tradicional de recuperación de imágenes basado en características, Keita propone un método que utiliza un dataframe local como índice de vectores. Cuando el usuario proporciona un texto o una imagen como criterio de búsqueda, el modelo realiza una búsqueda de texto a imagen por defecto.

Realiza una similitud de coseno entre cada vector de imagen y el vector de entrada del usuario, ordenando los resultados en función de la puntuación de similitud en orden descendente y devolviendo las imágenes más similares, excluyendo la primera que corresponde a la consulta misma.

1.2. Investigaciones de detección de objetos y segmentaciones

La detección de objetos en imágenes es otro aspecto crucial en la recuperación de imágenes. La detección de objetos permite identificar y localizar objetos específicos en una imagen, lo cual es fundamental para muchas aplicaciones de recuperación de imágenes. Una técnica comúnmente utilizada para la detección de objetos es el uso de cajas(*boxs*) delimitadoras, que definen un área rectangular alrededor de cada objeto detectado. Esta técnica se puede combinar con las características de CLIP[14] para mejorar la precisión de la detección de objetos.

La combinación de modelos de segmentación y extracción de características de la imagen ha sido un enfoque común en el campo de la visión artificial. Este enfoque implica dividir una imagen en partes más pequeñas y manejables, identificar características clave en estas partes y luego utilizar estas características para realizar tareas como la clasificación de imágenes, la localización de objetos y la detección de objetos.

En el artículo *Segment Anything*[11] de *Meta AI*, se presenta un modelo llamado *Segment Anything (SAM)* que se especializa en la segmentación de imágenes. El modelo SAM genera múltiples mapas binarios para cada imagen.

1.3. Investigaciones basadas en recuperación de imágenes

La mayoría de los modelos de recuperación de imágenes se fundamentan en el principio de recuperar de imagen a imagen. Sin embargo, existen modelos que, basándose en los anteriores y en modelos de lenguaje, utilizan la combinación de ambas entradas para recuperar información. En este contexto, el artículo *Sentence-level Prompts Benefit Composed Image Retrieval*[1] de *Yang Bai et al.* propone una estrategia innovadora para la recuperación de imágenes compuestas utilizando indicaciones a nivel de oración.

La recuperación de imágenes compuestas es un proceso que busca recuperar imágenes específicas utilizando una consulta que incorpora tanto una imagen de referencia como una descripción relativa. A pesar de que muchos modelos de CIR (*Composed*

Image Retrieval) existentes adoptan la estrategia de fusión tardía para combinar las características visuales y lingüísticas, este artículo sugiere una alternativa diferente.

Se han sugerido varios enfoques para generar un token de palabra a partir de la imagen de referencia, que luego se integra en la descripción relativa para la CIR. En contraste, los autores del artículo proponen aprovechar los modelos V-L (*Vision - Language*) preentrenados, como BLIP-2 (*Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models*)[13], para generar indicaciones a nivel de oración.

Este enfoque innovador implica la concatenación de la indicación a nivel de oración aprendida con la descripción relativa. De esta manera, se pueden utilizar de manera eficaz los modelos existentes de recuperación de imágenes basados en texto para mejorar el rendimiento de la CIR. Así, este método propone una forma elegante y efectiva de mejorar la precisión y la eficacia de la recuperación de imágenes compuestas.

Capítulo 2

Propuesta

La propuesta para el sistema de recuperación de información de imágenes basado en contenido, comprende dos etapas principales y una etapa final de sistema de recuperación. La tarea de cada una de las etapas, en conjunto, posibilita lograr el objetivo principal, obtener relevancia para un conjunto de imágenes, desde una consulta en forma de texto en lenguaje natural. Para vincular conceptos en un mismo espacio, ya sea imagen o texto, se emplean embeddings (vectores multi-dimensionales) como contextos. Sera posible establecer similitud entre cualquier cualquier concepto que se encuentre en forma de embedding, en este caso, contamos con conceptos como imágenes o conjunto de píxeles, y texto, potencialmente, el lenguaje natural.

2.1. Etapas de la propuesta

La primera etapa de procesamiento y extracción de características de la imagen tiene lugar durante el tiempo de indexación. Las características de las imágenes se procesan y almacenan en forma de vectores en el momento de generación de los datos. Posteriormente, utilizando estos vectores, se procede a recuperar la información en tiempo real mediante métodos mucho más rápidos y eficientes que los utilizados en tiempo de indexación. Este análisis de la imagen incluye diversas opciones y permite la ajustabilidad de parámetros para variar los resultados de acuerdo a lo que se determine más adecuado para lograr una mejor similitud entre contextos.(ver diagrama 2.1)

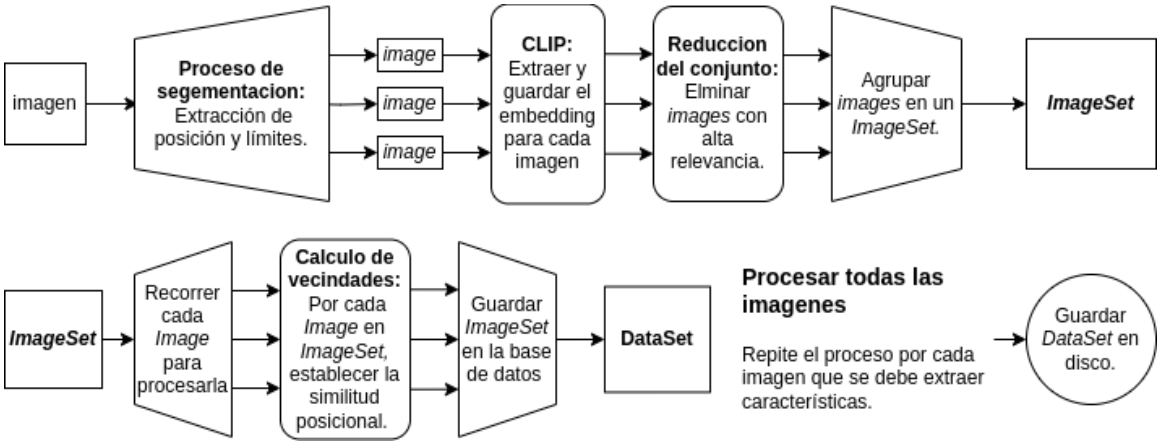

Figura 2.1: Etapa de procesamiento de imagen.

La segunda etapa se encarga de procesar el texto proporcionado por la consulta de los usuarios. Esta etapa está compuesta por una variante implementada, y dos variantes propuestas. Está diseñada para ser escalable, lo que permite mejorar significativamente el procesamiento del texto en futuras implementaciones.(ver diagrama 2.2)

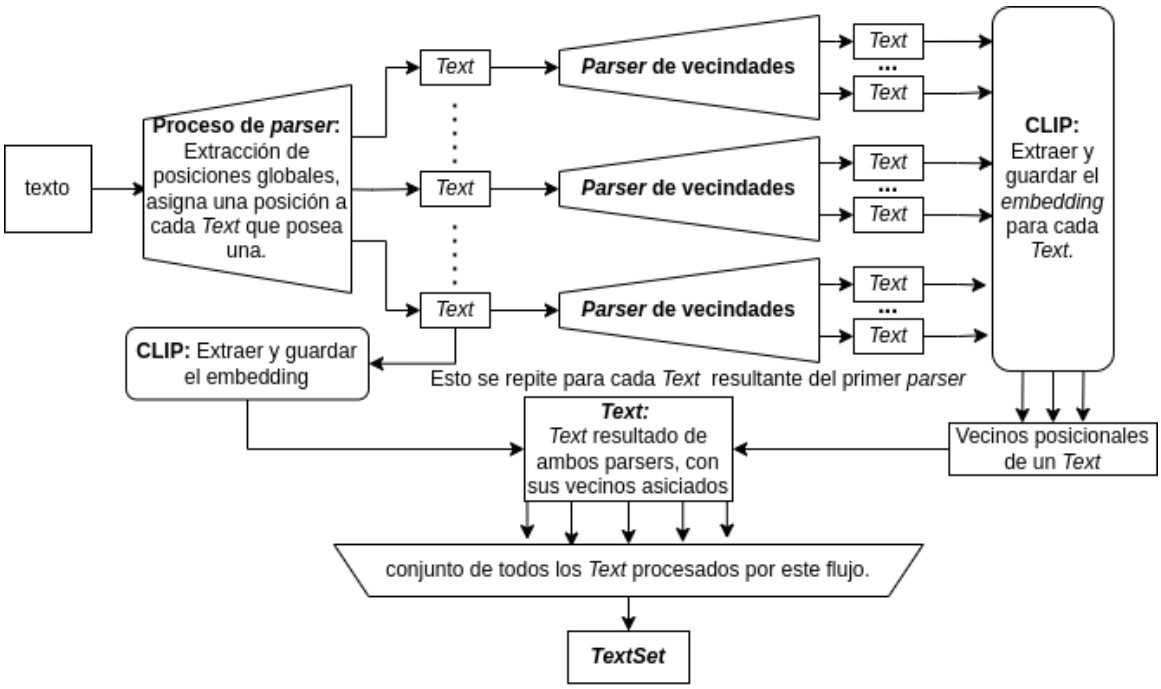

Figura 2.2: Etapa de procesamiento de texto.

La tercera y última etapa, es la encargada de recuperar datos relevantes para un concepto entrado por consulta. Potencialmente, los datos serán imágenes y las consultas texto, no obstante, dado el enfoque planteado, los datos y la entrada pueden pertenecer a cualquier concepto del cual sea posible obtener su embedding. Un ejemplo de ello es que, con el mismo sistema de evaluacion, se puede pasar por consulta tanto imágenes como textos, ya que ambos poseen embeddings como contexto.(ver diagrama 2.3)

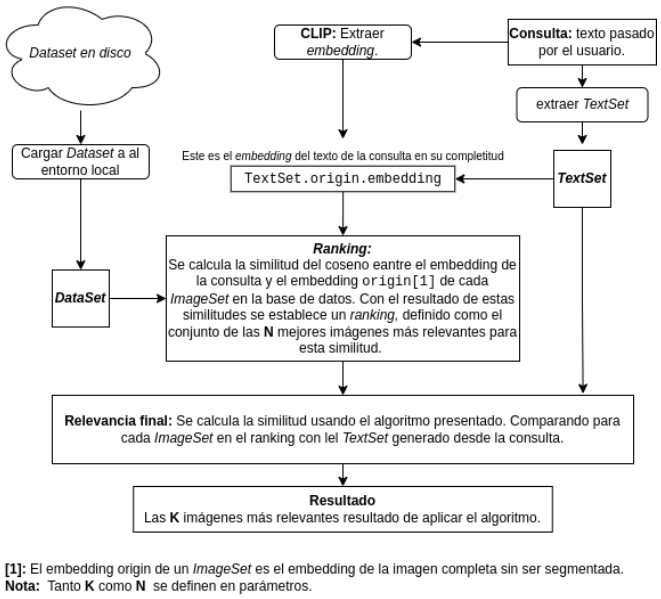

Figura 2.3: Etapa del sistema de recuperación.

2.2. Etapa de procesamiento de imagen

Esta etapa se divide en tres variantes que, a su vez, poseen una base común como metodología, cambiando solo los conceptos que se emplean en cada una de estas. La metodología principal está dada por una serie de procesos que serán explicados en esta sección. Para facilitar la comprencion de esta, se definen dos conceptos fundamenales.

El primero, es el concepto de ***Image***. Una *Image* se define como un conjunto de características que se extraen de una imagen: un embedding, una descripción, una posición, una serie de vecinos por posición relativa a esta imagen, y límites en los ejes coordenados. Los vecinos hacen referencia a las imágenes cercanas a la *Image* en cuestión. Los límites definen la región rectangular donde se encuentra ubicada la imagen, dado que una imagen no siempre ocupa todo el espacio del plano.

El segundo concepto es ***ImageSet***. Un *ImageSet* se define como un conjunto de

Images donde cada una de las imágenes de referencia de estos, está relacionada con una misma imagen, una imagen general. Cada una de estas *Images* constituye una segmentación de la imagen principal. En resumen, *ImageSet* es el conjunto de características de una imagen subdividida en segmentaciones relacionadas, posicionalmente, entre sí.

Con estos dos conceptos aclarados, se pasa a definir la metodología base para el procesamiento de imágenes. Inicialmente, para una imagen dada, se define un *ImageSet* que será la encargada de agrupar todas las características de la imagen en una relación, en un conjunto. Para la extracción de características y procesamiento de la imagen, esta pasa a ser proecsada y segmentada por un modelo de segmentación de imágenes. Este trabajo, en particular, utiliza el modelo de segmentación SAM (*Segment Anything Model*)[6]; dado la escalabilidad del código, este último es perfectamente sustituíble por otro modelo que se encargue de realizar la tarea de sementación de imágenes. En trabajos futuros, sera posible hacer uso de modelos más potentes o más rápidos que el integrado actualmente. El tamaño de las segmentaciones, así como los hiperparámetros del modelo SAM, son modificables. En caso de ser necesario, los parametros del modelo pueden ajustarse a la problemática que ha surgido.

Terminada la tarea de segmentacion de imagen, el paso que sigue, es descartar las imágenes que resultan ser semejantes a muchos conceptos muy distantes entre sí, las imágenes que son relevantes y afectan por ello los resultados esperados. Este proceso de descartar imágenes relevantes es muy importante para que los modelos usados no generen conceptos relevantes que resultan perdida de precisión general. Para realizar este proceso, se cuenta con un conjunto de conceptos muy distantes entre sí, un umbral a partir del cual se considera relevante una similitud, y un porcentaje que indica, cuándo una imagen posee una relevancia alta para el universo. Ergo, una imagen sera descartada por alta relevancia, si resulta ser similar a una parte considerable del conjunto de conceptos distan mucho entre sí. Una vez concluido este proceso, se encontrará en el *ImageSet* definido al inicio, todo conjunto de imágenes que la conforman, cada una de las cuales será una imagen que contendrá, en este momento, un identificador, una posición relativa al espacio general, y los límites superiores e inferiores de cada eje del plano. Estos dos últimos se obtienen como resultado de aplicar el modelo de segmentación. Cada una de estas imágenes, están ahora relacionadas como conjunto, teniendo esta relación, sus posiciones y sus límites. El siguiente paso es establecer las relaciones de vecindad posicionales para las imágenes entre sí.

Para establecer esta relación, se cuentan con etiquetas, ya que se espera que la descripción posicional sea por texto, las etiquetas entonces están acorde a ello, están acorde a lo que se espera como lenguaje natural. Las etiquetas son las siguientes:

- **n**: Indica la cercanía a la posición superior relativa a la imagen. Dice que tan cerca está un vecino de la posición superior relativa a la imagen en cuestión.

- **s**: Indica la cercanía a la posición inferior relativa a la imagen. Dice que tan cerca está un vecino de la posición inferior relativa a la imagen en cuestión.
- **e**: Indica la cercanía a la posición derecha relativa a la imagen. Dice que tan cerca está un vecino de la posición derecha relativa a la imagen en cuestión.
- **w**: Indica la cercanía a la posición derecha relativa a la imagen. Dice que tan cerca está un vecino de la posición derecha relativa a la imagen en cuestión.
- **beside**: Indica la cercanía a la posición derecha o izquierda relativa a la imagen. Dice que tan cerca está un vecino de algún lado de la imagen en cuestión.
- **ne**: Indica la cercanía a la posición derecha superior relativa a la imagen. Dice que tan cerca está un vecino de la posición a 45° de la imagen en cuestión.
- **nw**: Indica la cercanía a la posición izquierda superior relativa a la imagen. Dice que tan cerca está un vecino de la posición a 135° de la imagen en cuestión.
- **se**: Indica la cercanía a la posición derecha inferior relativa a la imagen. Dice que tan cerca está un vecino de posición a 315° de la imagen en cuestión.
- **sw**: Indica la cercanía a la posición izquierda inferior relativa a la imagen. Dice que tan cerca está un vecino de la posición a 225° de la imagen en cuestión.
- **in**: Indica que el vecino está dentro de la imagen en cuestión.
- **next**: Indica que el vecino está en una posición cercana a la imagen en cuestión.

Cada vecino será entonces, una relación etiqueta-(concepto, similitud), donde se hace referencia al concepto en la posición dada, y se indica que tan similar es la posición con respecto a una etiqueta. Por ejemplo, (ver imagen1).

La definición de la similitud posicional ha sido un tema muy cuestionable, dado que se busca similitud entre lenguaje natural e imágenes. Definir entonces, qué tan similares son las posiciones entre imágenes, está estrechamente relacionado con la interpretación de cómo se puede pretender explicar posiciones a nivel de lenguaje natural y cómo se espera la salida para ello.

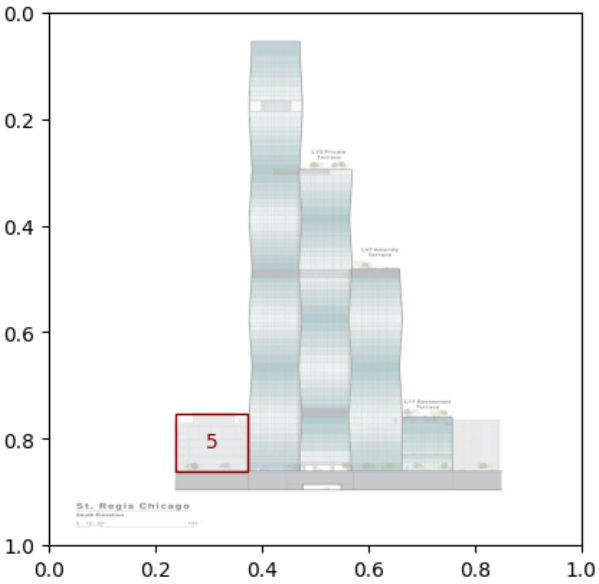

Figura 2.4: Interpretacion de relaciones espaciales

Un ejemplo claro de que la definición de posición está ajustada a la interpretación del observador sería una imagen en la que se ve un edificio. En *Figura 2.4* se podría decir que el edificio está a la derecha de la *Image 5*[figura 2.4], porque en realidad lo está, en su base coinciden. También se podría decir que el edificio está arriba, en la posición superior de la *Image 5*. Y, evidentemente, el edificio parece estar en la zona superior derecha de la *Image 5*. Se interpreta cada una de estas posiciones, definir que posición es mejor sigue siendo un problema de interpretcion del observante.

Como respuesta a esta ambiguedad se plantearon varias métricas o alternativas para poder decidir qué usar. Por ejemplo, la métrica que utiliza distancias angulares dudará que el edificio está completamente ubicado en su derecha, mientras que las otras interpretarán que hay un edificio en su derecha con una similitud perfecta. Las métricas usadas se definen a continuación.

Ángulo entre los centros y Distancia entre los extremos

Para el calculo de esta similitud se definen las siguientes varibles:

- $d =$ Distancia euclideana entre las imágenes. Esta se calcula como la distancia mínima entre los límites de las *Images*.
- $a =$ Ángulo entre el centro del *Image* y el centro del *Image* vecino.

- $b =$ El ángulo correspondiente a la posición dada, con respecto a la imagen, por ejemplo, para la posición derecha superior, se espera que el ángulo esté cerca de los 45 grados.
- $u =$ Umbral multiplicador de resultado, con ello se ajusta la salida como se desee.

Con estas variables definidas, donde para cada posición varía el valor del b esperado, llegamos a que la similitud posicional propuesta sera: será igual a:

$$sim_pos = (1 - d) * (1 - |a - b|/2 * \pi) * u$$

Reescalado las imágenes.

Esta alternativa usa la misma idea de la anterior, solo que ahora antes de calcular la distancia angular, se reescalan las imágenes a una misma dimensión. De esta manera se consigue que las posiciones sean más importantes que las escalas de los objetos que se analizan. Ahora un objeto con mucho más escala que otro se ajustará al pequeño para definir sus posiciones relativas. En caso del ejemplo anterior, *Figura 2.4*, el edificio estará a la derecha de la *Image 5*, pero no se considerará en zona superior porque la base en el eje horinzontal es la misma, es decir parten de la misma posición. Esta es una manera interpretar relaciones posicionales, por eso ha sido planteada como una variante.

Alternativa de pesos en los ejes coordenados.

Esta alternativa calcula la distancia entre los objetos tanto en el eje vertical como en el horizontal. Tanto al eje principal como al secundario se le asignara un pes de importancia que, ajustara la distancia de este eje. El eje principal será el de dezplazamiento de la posición indicada, por ejemplo, si se está analizando un objeto a la derecha o a la izquierda, entonces el eje principal será el eje horizontal, ya que de derecha a izquierda el desplazamiento es horizontal, y el secundario será el vertical. Teniendo esto en cuenta se ajustan los parámetros a conveniencia de tal forma que se acerque al mejor resultado posible, descrito posicionalmente. Cada uno de los parámetros ajustables correspondientes a esta función son los siguientes.

- **Multiplicador de posiciones relativas:** Este parámetro se usa para ajustar con un multiplicador los valores de las posiciones relativas entre objetos, es decir, el resultado de calcular la similitud poscional es modificado por este multiplicador, dando mas o menos importancia a este proceso.

- **Importancia distancia en el eje no principal:** Define qué tan importante se considera la cercanía o lejanía en el eje principal de una vecindad. Por ejemplo, si estamos trabajando en una vecindad izquierda o derecha, el eje no principal será el horizontal, el eje x. Una vez se tiene la distancia del eje no principal, se eleva este resultado a la potencia del valor indicado por este parámetro.
- **Importancia distancia en el eje principal:** Define qué tan importante se considera la cercanía o lejanía en el eje principal de una vecindad. Por ejemplo, si estamos trabajando en una vecindad izquierda o derecha, el eje principal será el horizontal, el eje y. Una vez se tiene la distancia del eje principal, se eleva este resultado a la potencia del valor indicado por este parámetro.

A continuación se muestran valores de similitud por vecindad para dos métricas distintas, mostrando la similitud de los vecinos para la *imagen 3*. Estos resultados son ajustables cambiando valores de parámetros especificos. La etiqueta define la posición relativa a la imagen en custion, en este caso la imagen 3, cada valor está compuesto por dos elementos, la referencia a la imagen en la posición 0, y en la posición 1, el valor de semejanza de sus posiciones con relación a la zona de la imagen que le correspond, esta zona está definida por la etiqueta.

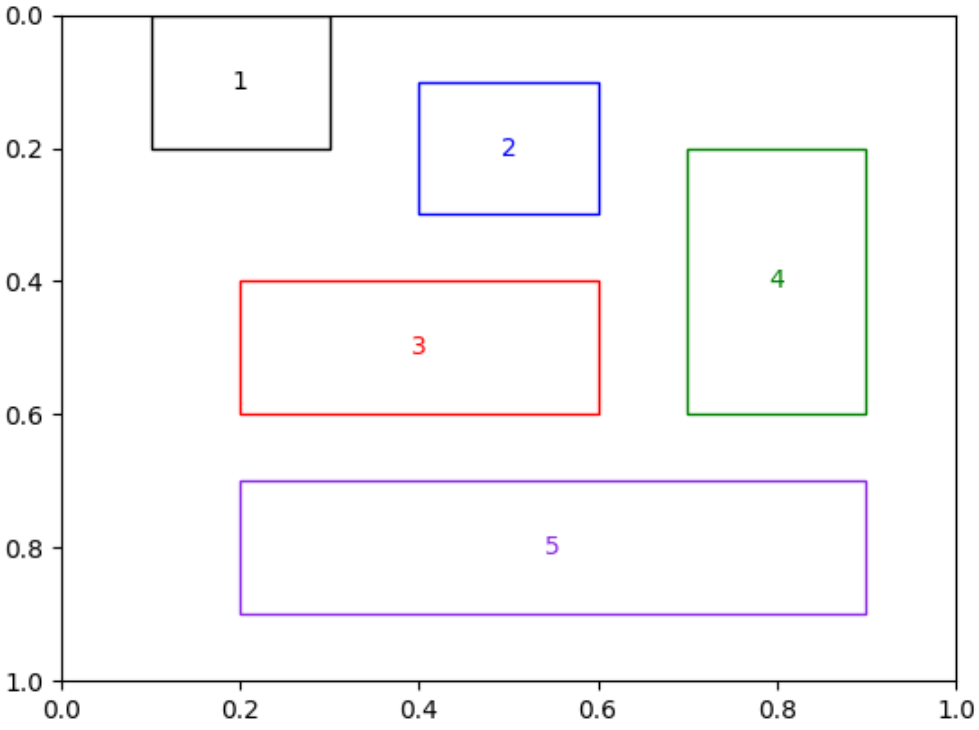

Figura 2.5: *ImageSet* de prueba para observar vecindades.

Ángulo entre los centros y Distancia entre los extremos

```
Image3 Neighbords using angle_similarity
 w: [(image 1, 0.2291)]
 e: [(image 4, 0.7596), (image 5, 0.2656)]
 n: [(image 1, 0.5638), (image 2, 0.7156), (image 4, 0.1559)]
 s: [(image 5, 0.6343)]
 nw:[(image 1, 0.4583)]
 ne:[(image 4, 0.2807)]
 sw:[]
 se:[(image 5, 0.5313)]
 in:[]
 beside: [(image 1, 0.2062), (image 4, 0.6836), (image 5, 0.2390)]
 next: [(image 5, 0.6343), (image 1, 0.5638), (image 2, 0.7156),
        (image 4, 0.7596)]
```

Alternativa de pesos en los ejes coordenados

```
Image3 Neighbords using weights_similarity
 w: [(image 1, 0.4734)]
 e: [(image 4, 0.9369), (image 5, 0.5694)]
 n: [(image 1, 0.6555), (image 2, 0.9369), (image 4, 0.5694)]
 s: [(image 5, 0.9369)]
 nw: [(image 1, 1.1289)]
 ne: [(image 4, 1.5063)]
 sw: []
 se: [(image 5, 1.5063)]
 in: []
 beside: [(image 1, 0.4734), (image 4, 0.9369), (image 5, 0.5694)]
 next: [(image 5, 1.5063), (image 1, 1.1289), (image 2, 0.9369),
        (image 4, 1.5063)]
```

La segunda alternativa no tiene en cuenta las dimensiones de las segmentaciones para calcular la posición relativa. En la primera métrica, si una imagen es más alta, está más extendida en el eje y, entonces su ángulo relativo a otra varía según la escala de la misma.

Teniendo definida entonces, la metodología principal de esta etapa. Sobre la base de esta, se proponen tres variantes para ser utilizadas como conceptos, entre los cuales, poder hallar relavancia mediante cálculos de similitud.

2.2.1. Uso de descripciones de las imágenes

Este método propone generar la descripción de la imagen en lenguaje natural, con el texto en lenguaje natural hallar la similitud directamente sería una opción válida, pero no es considerada en un primer momento porque un trabajo vago sire el lenguaje natural no nos garantiza una buena precisión al momento de hallar similitud. Como alternativa a esto se usan embeddings del texto, generados con modelos potentes de visión artificial que están diseñados para generar embeddings para texto. En el caso particular de este proyecto se usa el modelo CLIP[14] que permite generar embeddings tanto desde texto como desde imágenes, vectores de 512 dimensiones. El proceso completo de esta metodología sería entonces generar descripciones desde la imagen, desde esta descripción generar un embedding y usar este último como contexto que se compare para hallar similitud entre este y los embeddings generados desde el texto de la consulta.

Este proceso requiere de modelos diseñados para describir imágenes en lenguaje natural, modelos como son BLIP[9], BLIP2[13], LLaVA[12] y GPT4-V.[22] El proceso de generar descripciones suele ser lento según el modelo que se utilice, los modelos más

potentes requieren de un hardware, de GPU, muy potente. En el caso de este proyecto se utiliza BLIP[9] para generar descripciones, también cuenta con el modelo BLIP2 integrado pero no es utilizado debido a su exigencia de hardware y la insuficiencia del mismo.

Como todo modelo de Machine Learning, estos modelos están sujetos a errores y tienen sus puntos fuertes y débiles. Por lo general, imágenes poco claras generan texto con concepto muy alejado a lo esperado, por ello se suelen perder algunas segmentaciones que se consideran importantes. No obstante la gran mayoría de las imágenes como estas suelen ser desechadas por tener mucha relevancia.

2.2.2. Uso de embedding de las imágenes

La segunda variante, es la que, actualmente, se usa como default en el proyecto, dado que es más ligera y alcanza mejores resultados que la variante 1.

Esta variante implica tomar la imagen segmentada, procesarla en un modelo generador de embeddings, se usa CLIP, y usar este embedding como concepto para hallar similitud. Dado que CLIP[14] está diseñado para generar embeddings para imágenes y para textos en el mismo espacio, esto es muy útil para usar este mismo generador para ambas partes y poder comparar similitud entre ellos.

El proceso de generar embeddings con CLIP[14] es considerablemente rápido, tardando, grosso modo, 20ms para generar el embedding de una imagen con una GPU v100; se puede decir que es aún más rápido en cuanto a su velocidad relativa a modelos generadores de descripciones, que suelen tardar varios segundos, dependiendo del modelo específico que se use. El proceso de generación de embeddings con CLIP[14] , si bien esta expuesto a fallos o salidas no esperadas, suele ser más preciso que utilizar un modelo generador de descripciones al alcance del hardware disponible o la disponibilidad del modelo, dado que los más potentes no son código abierto. Se desconoce que tan preciso puede ser usar el modelo GPT4-V[22], el cual es muy potente en este campo, como también se desconoce que tan preciso sería usar CLIP-2[23]para generar embeddings, ambos modelos son de código cerrado de OpenIA.

2.2.3. Combinación de ambos modelos

La tercera variante es combinar ambas variantes anteriores. De esta forma, el cálculo de la similitud sería entonces la media de la similitud por texto y la similitud por imágenes. Esta variante suele ser tan precisa como los modelos de descripción que se usan para su descripción, pero exige mucho hardware con el cual no se cuenta, se hace difícil utilizar modelos más potentes que BLIP[9].

2.3. Etapa de procesamiento de texto

Esta etapa consiste en extraer características del texto ingresado por consulta. Específicamente, características posicionales de los objetos en la imagen. Para ello, se desea que, para un texto de entrada en lenguaje natural, se genere una serie de textos parseados por posiciones descritas, equivalente a lo que se hace en la etapa 1 con las imágenes. El proceso de obtener posiciones en el texto suele ser más engorroso que en el procesamiento de las imágenes, dado que en el caso de las imágenes, al ser segmentadas, cada una de estas segmentaciones posee la información posicional en el plano de sí misma. En el caso del lenguaje natural es necesario parsear lo que expresa un usuario en una consulta. La etapa del procesamiento del texto de la consulta, a diferencia de la etapa de procesamiento de la imagen, ocurre en tiempo real en el momento de la consulta, ergo debe ser considerablemente rápida para que cumpla con las expectativas.

Al igual que en la etapa de procesamiento de imágenes, aquí también se definirán dos conceptos fundamentales para comprender la metodología utilizada.

El primero es el concepto de ***Text***. Un *Text* se define como un conjunto de características que se extraen de un texto, las cuales son: un string, un embedding, una posición y una serie de vecinos por posición relativos a este. Los vecinos hacen referencia a la cercanía entre conceptos. Por ejemplo, para el texto en lenguaje natural *In top right side of the image there are a dog next to a cat sleeping on a couch*, se espera un *Text* de la siguiente forma:

```
text.string = "a dog next to a cat sleeping on a couch."
text.pos = (0.5, 0.5)
text.neighbors[next] = "cat sleeping on a couch"
```

Para mejor comprension del lector, se debe asumir que las posiciones van desde $[(-1,1),(1,1)]$.

El segundo concepto es ***TextSet***. Un *TextSet* se define como un conjunto de *Texts* donde cada uno de representan subdivisiones de un único texto general, además, también posee un *Text* que hace referencia al texto original. En pocas palabras, *TextSet* es el conjunto de textos que conforman el texto original, separados por puntos o por posiciones indicadas con lenguaje natural.

Para esta etapa de procesamiento de lenguaje se plantean 3 variantes posibles, de las cuales, una ha sido implementada.

Análisis sintáctico y parser

La variante principal e implementada en el proyecto es el uso de un parser para lenguaje natural con el fin de parsear contenido que haga referencia a relaciones

posicionales. El lenguaje natural es ambiguo, un parser diseñado sobre este no será perfecto, pero puede abarcar una gran parte de las estructuras gramaticales que forman, de alguna manera, la definición de posiciones a través de texto.

Esta alternativa tiene como principal punto débil la ambigüedad del lenguaje, ergo, en algunos casos muy particulares el proceso de parser no nos dará la salida esperada. Como punto fuerte se puede decir que esta alternativa tiene un alto nivel de escalabilidad, cada vez que se quiera definir una nueva regla gramatical es posible hacerlo, dado que el lenguaje ya es, por naturaleza, ambiguo, una nueva regla gramatical no afectará en este aspecto; siempre que se agregue una nueva regla, esta no debe generar conflictos graves con las anteriores.

Para la implementación de este sistema de parser se usa la biblioteca *SLY (Sly Lex Yacc)*[3], esta biblioteca está diseñada para realizar análisis léxicos y sintácticos para lenguajes de programación, no está exactamente diseñada para este tipo de gramática ambigua pero se logra adaptar a las necesidades y al uso que se le da.

El análisis de un texto está dividido en dos fases, la primera fase de tokenización, y la segunda fase de análisis sintáctico o parser. Los tokens están definidos según la necesidad de buscar patrones comunes para describir posiciones en imágenes. Cada una de las palabras del texto serán procesadas por un análisis léxico que definirá si esta es token o palabra común. Dado que en el lenguaje natural todas las palabras pueden ser parte de otro texto, las palabras coincidentes con los tokens no siempre serán uno de estos, es decir, que una palabra esté en el grupo de tokens no implica que sea un token necesariamente. Una palabra es un token, sí solo sí, está contenida en un patrón que constituye una regla gramatical en su completitud y, este patrón no genera conflictos con otros patrones anteriormente analizados y seleccionados como tokens; de esta forma solo se consideran tokens los patrones que se usan para describir relaciones posicionales. Seguido a esta fase de tokenización está la fase de análisis sintáctico. Esta fase encarga de parsear todo el texto por relaciones posicionales en caso de que se especifiquen las mismas. Este análisis sintáctico o parser, está dividido en dos subetapas, dos parsers distintos. El primer parser consite en un análisis para las relaciones globales de los objetos con respecto a la imagen, y e segundo análisis se encarga de las relaciones posicionales relativas entre conceptos.

Parser de posiciones globales

En un primer momento se parsean las posiciones globales, las posiciones de un objeto respecto a la imagen en su totalidad, de esta forma se obtiene la posición de cada *Text* que conformará el *TextSet*. Para ello se definen algunas reglas gramaticales que se usan en el lenguaje natural para expresar posición en el espacio, o relativa a la imagen. Por ejemplo, para la frase: *"there are a dog next to a ball in top of image"*, se parsea a través de la regla */ÏS text ON pos OF IMAGE"*, de esta forma se obtiene un

Text con string igual a *"dog next to a ball"* y posición, grosso modo, igual a (0,0,66). En un primer momento el tokenizador o analizador léxico referente a este análisis de posiciones globales posee como tokens los siguientes:

```
tokens = [ON, TO, OF, AND, IS, POS, POSITION, IMAGE, WORD, NEXT]
```

Cada uno de estos es representado por una palabra clave que puede corresponder a ellos, no poseen una palabra única para cada token, un token puede estar representado por más de una palabra, y cada una de estas cuando aparezca en el texto, puede representar un token de lo anteriores. Las palabras claves representan un token dado, el conjunto de palabras claves es perfectamente extensible, solo se debe agregar a este una nueva palabra clave y esta será entonces tratada como posible token.

```
keywords = {
        ON = [ in, on, at, near, find...],
        OF = [ of ],
        TO = [ to ],
        AND = [ and ],
        IS = [ is, are, there's, find ...],
        POS = [ left, right, buttom, bottom, top, down, up,
        lower, center, middle, corner...],
        POSITION = [ position, side, location...],
        IMAGE = [ image, picture, photo...],
        NEXT = [ next, near ]
        }
```

Durante la fase de tokenización, cada palabra en el texto se somete a un proceso para determinar su tipo de token. Este proceso implica verificar si la palabra coincide con un patrón posible en la gramática. Si la palabra cumple con este criterio, se clasifica como un token; si no, se designa como una palabra común o WORD.

Posteriormente, en la fase de análisis sintáctico, se define la gramática de las relaciones espaciales, la cual se considera el pilar fundamental del procesamiento de texto. Este análisis se basa en la identificación de patrones gramaticales que permiten expresar relaciones espaciales en el texto. Las reglas gramaticales han sido definidas siguiendo la siguiente gramática.

```
WORD ::=  WORD
        | Any

pos ::=  POS
       | POS POSITION
```

```
text ::=  WORD
        | [text]+

relation ::=  ON pos IS text
            | ON pos OF IMAGE IS text
            | ON pos text
            | IS text ON pos OF IMAGE
            | IS text ON pos ","
            | IS text ON pos !OF
            | text ON pos !OF
            | text ON pos OF IMAGE
            | ON pos "," IS text
            | ON pos OF IMAGE "," IS text
            | ON pos OF IMAGE "," text
            | ON pos "," text
```

!TOKEN indica que en ese patrón no puede haber un token tipo TOKEN en esa posición. Nótese que, en el caso de *ÏS text ON pos !OF"*, la ausencia de !OF al final podría generar conflictos de ambigüedad y hacer que palabras necesarias sean descartadas. [**TOKEN**]|+ indica que el token se repite una o más veces. **Any** se refiere a un token cualquiera que sea tratado como palabra.

En la segunda parte del parser, se analizan las posiciones relativas de los objetos entre sí. Al igual que en el caso anterior, existen una serie de reglas definidas para esto. Por ejemplo, para el texto *"dog next to a ball"*, el parser seguirá la regla *"text NEXT TO text"*. De esta forma, se espera obtener un text con string igual a *"dog"*, *neighbors["next"] = .ª ball"*.

El proceso de parser y tokenizacion, es igual al anterior, solo cambian las reglas gramaticales, y tokens no necesariamente idénticos. Por ejemplo el token *Image* no existe en esta análisis. La gramática del parser está definida por las reglas, tokens y palabras claves definidos a continuación.

```
Tokenization
tokens = [ON, OF, AND, IS, POS, POSITION, WORD, NUM, TO, NEXT]

keywords = {
        ON = [in, on, at, find]
        OF = [of]
        AND  = [and]
        IS = [is, are, theres, find]
        POSITION = [position, pos, side,location]
```

```
        NEXT = [next]
        TO = [to]
        POS = [ left, right, bottom, top, down, up, lower, center,
        middle, corner, near, bellow, front, beside ...]
    }

Gramatic
WORD ::=  WORD
            | Any

pos ::=  POS
           | POS POSITION

text ::=  WORD
            | [text]+
relation ::=   IS text ON pos OF text
            | IS text TO pos OF text
            | IS text ON pos TO text
            | IS text TO pos TO text
            | text ON pos TO text
            | text ON pos OF text
            | IS text , pos TO text
            | IS text , pos OF text
            | IS text pos TO text
            | IS text pos OF text
            | text pos TO text
            | text pos OF text
            | IS text NEXT TO text
            | IS text NEXT OF text
            | text NEXT OF text
            | text NEXT TO text
            | ON text pos , IS text
            | ON text pos , text
            | ON text pos IS text
            | ON pos OF text , IS text
            | ON pos OF text , text
            | TO pos OF text , IS text
            | TO pos OF text , text
            | ON pos TO text , IS text
            | ON pos TO text , text
```

```
| TO pos TO text , IS text
| TO pos TO text , text
| ON pos OF text IS text
| TO pos OF text IS text
| ON pos TO text IS text
| TO pos TO text IS text
| pos TO text , IS text
| pos TO text , text
| pos OF text , IS text
| pos OF text , text
| pos TO text IS text
| pos OF text IS text
| NEXT TO text , text
| NEXT OF text , text
| NEXT TO text IS text
| NEXT OF text IS text
```

Otras variantes de procesamiento de texto

Se ha considerado el uso de un Large Language Model (LLM) para procesar los textos de las consultas. A través de una API, se describiría el sistema del parser deseado y se esperaría que este, se encargue de parsear el texto. Modelos como GPT-3[18 podrían ser una opción viable. Sin embargo, se ha intentado reentrenar modelos de lenguaje para pasarles entradas y salidas, pero estos no han tenido éxito debido a que no están diseñados para devolver siempre la misma respuesta para una misma entrada. Por lo tanto, no es posible entrenarlos con este objetivo. Los modelos que han fallado en este aspecto incluyen GPT-2[19], BERT[4] y T5[20].

Otra alternativa válida es la creación de un modelo propio que se encargue de procesar texto y devolver relaciones posicionales de este. Se dispone de un parser, aunque no sea perfecto, sí permite conocer cuándo será correcto y en qué tipos de casos siempre devolverá la respuesta esperada. Con esta información, se puede conformar una gran parte de un enorme conjunto de datos de forma automática, utilizando modelos de lenguajes actuales como GPT-3[18]. Al describir al modelo, el tipo de estructura gramatical que se sabe, es correcta, se pueden generar muchos casos de entrenamiento que serán correctamente procesados por el parser. En combinación de estas dos condiciones, se generarán grandes cantidades de datos de manera automática. Este modelo tiene la ventaja de que la dataset puede ser ampliada con casos para los cuales el parser pueda fallar, permitiendo así aprender también de estos. Incluso se pueden crear diferentes parsers que no sean ambiguos entre sí para generar casos de prueba. Además, gracias al proceso de tokenizacion, también se tiene una relación

de importancia para cada palabra, lo cual puede constituir una capa de atención que se centra, principalmente, en los tokens pertenecientes a la gramática.

Una propuesta para este modelo ha sido creada pero aún no ha sido entrenada. El modelo está implementado en tensorflow.keras, y esta constituido por las siguientes capas:

- Dos capas de entrada, una para los tokens y otra para la atención que se le debe prestar a estos.
- Dos capas de *Embedding* que convierten los tokens a un espacio de *embedding*.
- Dos capas *Cov1D* que son utilizadas para aprender patrones.
- Una capa de *Attention*, para recibir las entradas y generar pesos segun la atención que se deba prestar a cada token.
- Dos capas *GlobalAveragePooling1D* para redimensionar el espacio a una dimensión.
- Una capa *Concatenate*, para concatenar ambas capas superiores.
- Una capa *Dense* de 256 dimensiones.
- Una capa *Dense* de 512 dimensiones.
- Una capa *Dense* de 1024 dimensiones.
- Una capa *Dense* con catidad de dimensiones igual a la longitud de la salida.
- Una capa *Reshape* para ajustar la salida al formato de matriz deseado.

Las primeras 6 capas del modelo están basadas en la documentación oficial de TensorFlow[8] para el uso de capas de atención, las capas de atención son las metodologías más fuertes y más utilizadas en el procesamiento del lenguaje natural. A partir del anno 2017 cuando fue publicado *Attention is all you need*[17], el procesamiento del lenguaje natural se ha inclinado a esta arquitectura.

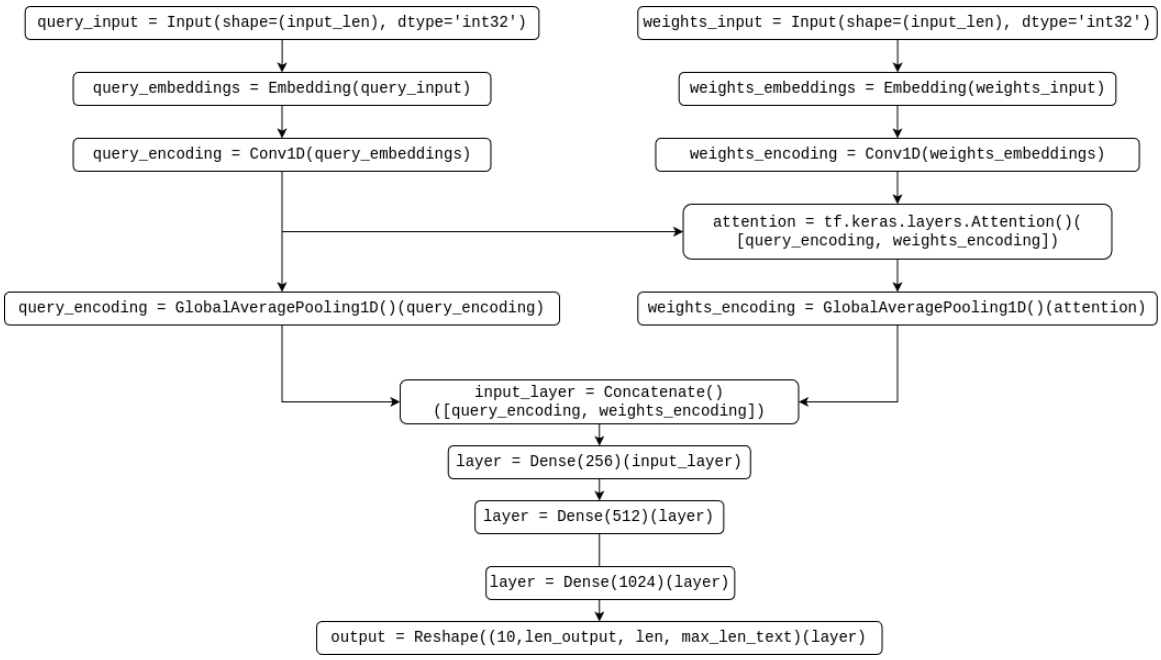

Figura 2.6: Modelo en Keras

2.3.1. Etapa de cálculo de similitud y recuperación de información

Después de las dos etapas anteriores, que implican el procesamiento de imágenes durante la indexación y el procesamiento de texto en tiempo real, se procede a calcular la similitud entre el texto proporcionado en la consulta y las imágenes de la base de datos.

Para calcular la similitud entre un *Text* y una *Image*, se siguen los siguientes pasos:

- Se calcula la similitud del coseno entre el embedding del *Text* y de la *Image*:

$$sim_cosine = 1 - cos(v1, v2)/||v1||v2||$$

- Se calcula la similitud por posición, para ello se multiplica el valor de la similitud del coseno por la similitud posicional más uno, para $v1$ y $v2$ vectores de posición y K un parametro ajustale, la similitud posicional esta dada por la forma:

$$sim_pos = K - \sqrt{((v_1.pos.x - v_2.pos.x)^2 + (v_1.pos.y - v_2.pos.y)^2)}.$$

  Luego la similitud final para un concepto en una posición dada distinta de *None*, estará dada por la fórmula

$$sim_inpos = sim_cosine * (sim_pos + 1)$$

- Si el *Image* y el *Text* poseen similitud suficiente, es decir, una similitud superior a un umbral que determina si es relevante, entonces se avanza al siguiente paso. Se pregunta por cada vecino en la posición p relativa al *Image* la similitud de este con cada vecino en la posición p relativa a *Text*. Por cada vecino en posición p en *Text*, se guardará el mejor, el más relevante de los vecinos en la posición p de *image*. Cada uno de los resultados obtenidos de este proceso se suma a la similitud final. Si el mejor de los resultados para una posición p está por debajo de un cierto umbral, entonces significa que no hay nada semejante a lo que se espera en la posición indicada, por lo que la similitud entonces, disminuirá.

```
for neighbord in Text.neigbords:
    sim_inpos = sim_inpos(neigbord, Image.neighbords)
    if max(sim_inpos) > umbral:
        sim_region = max(sim_inpos))
    else:
        sim_region = max(sim_inpos) - umbral
    end_sim_region+= sim_region
sim = sim_inpos(Text, Image)*(1+ end_sim_region)
```

- La similitud final esta dada por la similitud entre un *ImageSet* y un *text feaure*, es decir por un conjunto de *texts* y un conjunto de *Images*. Cada *Text* halla similitud con cada una de las *Images*, si esta similitud relevante, entonces se guarda. Luego de recorrer todas las imágenes, la similitud para el *Text* será la mejor de todas las similitudes con cada una de las *Images*. La similitud final, es entoces la suma de cada una de las similitudes por cada *Text* en *TextSet*. Esto garantiza que una descripción detallada y precisa de la imagen, encuentre, en caso de existir, imágenes muy relevantes a la descripción posicional.

$$\sum(sim(Text_i, ImageSet)|sim_i > umbral$$

Como se ha mencionado, los parámetros utilizados para los cálculos de similitud, distancias, importancia y características son modificables. Esto permite buscar un mejor equilibrio de similitud, si es necesario. Esta flexibilidad mantiene una arquitectura escalable que no requiere cambios en métodos o clases para variar los resultados. A continuación, se presentan estos parámetros.

- **umbral de importancia de la distancia euclideana** = float: Elevar a la potencia la distancia euclideana. Este valor define la imporatncia que se le da a las posiciones globales.

- umbral de division de la distacia euclideana = float: se divide el valor de la distancia euclineada por el valor de este parámetro.
- **similitud mínima relevante para regiones vecinas** = float: Si la similitud es menor que este valor, se considera poco relevante y, en caso de que todas las relaciones pocicionales sean poco relavantes, la similitud de la imagen va a disminuir para una descripción dada.
- **similitud mínima relevante** = float: Esta es la similitud a partir de a cual puede considerarse relevante un concepto, se utiliza en el ambito general, la relevancia posicional y general no tienen que depender una de la otra.
- **similitud mínima relevante original** = float: Esta es la similitud a partir de a cual puede considerarse relevante la smilitud entre la imagen original y texto original.
- **similitud mínima relevante usando descripciones** = float: Esta es la similitud a partir de a cual puede considerarse relevante una comparación a nivel de texto contra texto.
- **uso de regiones negativas** = Bool: Define si las regiones pueden aportar efecto negativo a la similitud, en el caso de cercanía entre un objeto y otro.
- **importancia de las regiones negativas** = float: Define que tan importante es una región negativa, mientras más alto, más baja será la similitud de una región poco relevante.

2.4. Otras Propuestas

Una propuesta inicialmente considerada, fue mejorar el reconocimiento de colores en la imagen, esta finalmente fue abandonada. Se decidió centrar más interés en la propuesta de reconocimiento posicional, ya que CLIP[14] maneja eficientemente las relaciones de colores en las imágenes, y por el contrario no es tan competente en el manejo de las relaciones posicionales.

El análisis de colores es una propuesta válida que implica un análisis exhaustivo de las segmentaciones para determinar de manera óptima a qué objeto pertenece un color. Sin embargo, para implementarlo, se requiere trabajar con máscaras como forma de segmentación y definir combinaciones de embeddings junto con el color. Aunque CLIP[14] no es particularmente eficaz en la generación de embeddings para las máscaras, no está claro si esto mejoraría los resultados. Si se opta por utilizar otro modelo generador de embeddings, se tendrían que implementar más formas de

similitudes en las que el modelo actual no es bueno, probablemente, para esos nuevos modelos, el diseño de esta propuesta resultaría en mejores resultados.

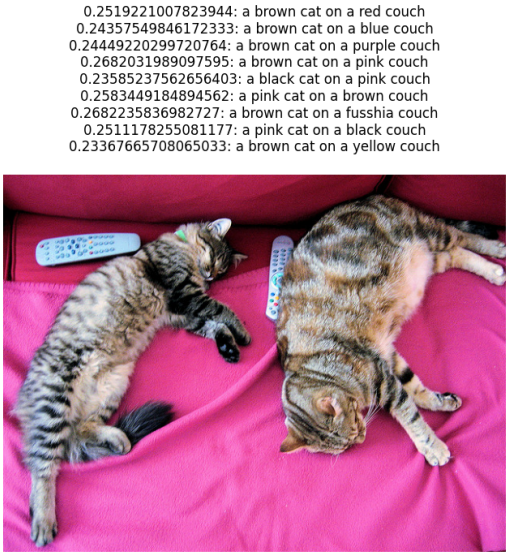

Figura 2.7: Colores

Figura 2.8: Posiciones

En las figuras 2.7 y 2.8, se puede observar que el control de colores de CLIP[14] es efectivo, ya que aumenta la similitud a medida que los colores resultan más similares a los de la imagen. Sin embargo, al contrario de las relaciones posicionales, no logra alcanzar la similitud esperada.

El uso de colores a nivel de cuadro de segmentación podría resultar en la selección de colores con una mayor área en la imagen, lo cual no es completamente preciso. Un enfoque más adecuado para esta propuesta de colores sería utilizar las máscaras de segmentación de la imagen con un fondo transparente. Esto podría resultar en una pérdida de precisión en la generación del embedding, pero podría conducir a resultados más interesantes que los que CLIP[14] puede manejar actualmente.

Capítulo 3

Detalles de Implementación y Experimentos

3.1. Experimentos

El objetivo principal de la propuesta presentada es mejorar la similitud entre textos más descriptivos y las imágenes correspondientes que contienen la mayor cantidad de detalles descritos en el texto. Para lograr esto, se han implementado tres variantes distintas que se centran en recuperar imágenes para descripciones posicionales en la consulta.

En una fase experimental, cada una de estas implementaciones ha sido probada y comparada tanto entre los resultados que se obtiene de cada fase, como entre cada resultado y la recuperación simple de consulta e imagen, similitud por distancia del coseno. Este proceso permite evaluar la efectividad de cada método y determinar cuál ofrece los mejores resultados.

Para facilitar la comprensión de los resultados y los experimentos realizados, se proporciona un conjunto de imágenes de muestra. Estas imágenes han pasado por el sistema de recuperación, donde se han aplicado medidas de similitud y se ha utilizado cada variante. De esta forma, se puede analizar cómo cada método afecta a la recuperación de las imágenes y su grado de similitud con los textos descriptivos.

3.1.1. Procesamiento de imágenes usando embbedings

El uso de esta variante demostró ser particularmente efectivo para la recuperación de consultas detalladas en términos de posición. Para ilustrar cómo funciona, consideremos un ejemplo que destaca una mejora significativa en la recuperación de información precisa.

Dos consultas fueron formuladas inicialmente, las cuales parecen bastante simila-

Figura 3.1

Figura 3.2

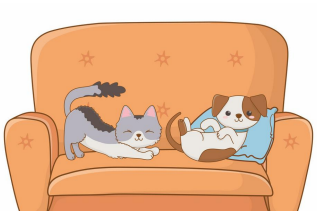

Figura 3.3

Figura 3.4

Figura 3.5

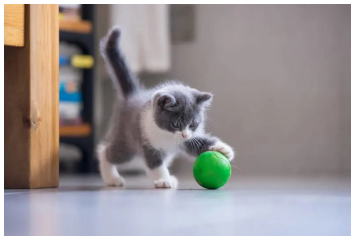

Figura 3.6

Figura 3.7

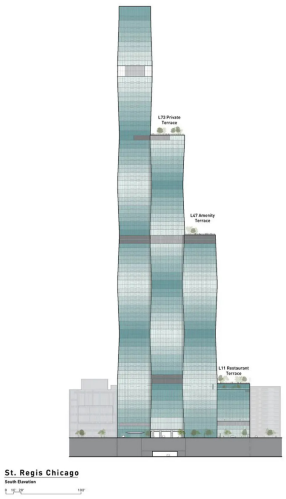

Figura 3.8

Figura 3.9

Figura 3.10

Figura 3.11

Figura 3.12

Figura 3.13

Figura 3.14

Figura 3.15

res. Sin embargo, la situación posicional es el factor principal que las diferencia. En el primer escenario, la consulta no coincide con las posiciones de los objetos, lo que resulta en un valor de relevancia mucho menor que la consulta que hace referencia precisa a las relaciones posicionales entre los conceptos de la imagen. En este caso, la imagen es bastante clara, lo que facilita que el modelo CLIP[14] no genere errores y lance resultados no esperados. Es bastante claro lo que es cada objeto en la imagen.

Este ejemplo demuestra cómo la variante implementada puede mejorar la precisión de la recuperación de información, especialmente cuando se trata de consultas detalladas en términos de posición

```
"a cat on left side of dog":
value: 0.3123254179954529

"there is a dog on left of a cat":
value:0.6470594258080886
```

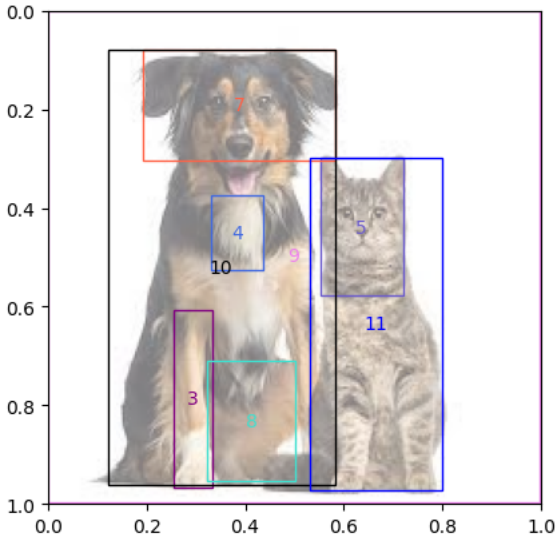

Figura 3.16

Paralelamente, se determinó la relevancia para las mismas consultas e imágenes, utilizando únicamente el mismo procedimiento que emplea CLIP[14], similitud del coseno[15], para encontrar similitud. En este caso, se calculó la similitud mediante la distancia del coseno.

```
"a cat on left side of dog":
value: 0.3123254179954529

"there is a dog on left of a cat":
value: 0.31687411665916443
```

Es posible observar que la relevancia de la imagen para ambas consultas es muy similar. Sin embargo, el resultado de esta medida de similitud se inclina hacia la consulta que expresaba incorrectamente las relaciones posicionales.

Para una mejor observación de este proceso, se ejecutó la recuperación de información en la muestra de imágenes. De esta manera, se pueden observar los resultados alcanzados por esta metodología. A continuación, se muestra una serie de consultas y resultados de aplicar la misma.

```
"a cat on left of an another cat":
image_1: 0.5910724142355481
image_2: 0.552273309297955
image_3: 0.57958666555157
image_4: 0.20404787361621857
```

```
image_5: 0.17712320387363434
image_6: 0.27128875255584717
image_7: 0.16639965772628784
image_8: 0.15555395185947418
image_9: 0.5865751181561818
image_10: 0.6205110420687305
image_11: 0.2572214603424072
image_12: 0.27217862010002136
image_13: 0.27809980511665344
image_14: 0.569566040832626
image_15: 0.6235474817208575

"a cat on left side of dog":
image_1: 0.24559526145458221
image_2: 0.5900955027099477
image_3: 0.5933583085861937
image_4: 0.2507782280445099
image_5: 0.16662538051605225
image_6: 0.24885490536689758
image_7: 0.28234352743521146
image_8: 0.14847618341445923
image_9: 0.6239545388814016
image_10: 0.3215339481830597
image_11: 0.2331039309501648
image_12: 0.24892988801002502
image_13: 0.6274699419732972
image_14: 0.3123254179954529
image_15: 0.27217918634414673

"there is a dog on left of a cat":
image_1: 0.2507342994213104
image_2: 0.5737969104184191
image_3: 0.6017459795652185
image_4: 0.25493425130844116
image_5: 0.17491364479064941
image_6: 0.26476845145225525
image_7: 0.17373675107955933
image_8: 0.16383123397827148
image_9: 0.30724239349365234
image_10: 0.6605514306535915
```

```
image_11: 0.2553481161594391
image_12: 0.2543933391571045
image_13: 0.30912819504737854
image_14: 0.6470594258080886
image_15: 0.2859894335269928

"a woman beside to cat, a woman in right of a dog":
image_1: 0.25769439339637756
image_2: 0.251406192779541
image_3: 0.27823591232299805
image_4: 0.2350061535835266
image_5: 0.1671127825975418
image_6: 0.23403123021125793
image_7: 0.17912636697292328
image_8: 0.16113649308681488
image_9: 0.3029889166355133
image_10: 0.3197519779205322
image_11: 0.2800091803073883
image_12: 0.23648248612880707
image_13: 0.29852136969566345
image_14: 0.3131987154483795
image_15: 0.25541138648986816
----------------------------------------
```

En estos ejemplos, se pueden apreciar buenos resultados relacionados con las posiciones espaciales. Un claro ejemplo de ello es la consulta "there are a dog in left of a cat". Los resultados se ajustan a las relaciones espaciales y no encuentran relevancia alta para imágenes donde cada uno de sus conceptos coincide en gran medida con lo pasado por la consulta. Por ejemplo, las imágenes de las figuras 3.13 y 3.9 poseen conceptos muy parecidos, pero posiciones diferentes.

Por otro lado, encontramos resultados con alta precisión que, para una observación humana, no deberían tenerla. El caso de la imagen de la figura 3.3, resulta relevante dado que CLIP[14] entiende que el objeto a la izquierda tiene gran parecido con "dog", y por lo tanto el sistema lo encuentra relevante. En el caso de la imagen 3.2, las posiciones son ambiguas, los objetos están superpuestos, y el modelo de segmentación, SAM[11], genera particiones varias para el perro y para el gato, lo que suele causar una ambigüedad posicional a nivel de imágenes.

Paralelamente a este ejemplo, se aprecia que la consulta .ª cat on left of a dog", que utiliza los mismos conceptos, también devuelve resultados satisfactorios en cuanto a la posición de los mismos.

El resultado de la última consulta indica que no se encontró, para ninguna imagen, relaciones posicionales planteadas en la consulta. Esto es debido a que la imagen de la mujer resulta poco relevante para el texto "mujer", por lo tanto no se entra a buscar vecindades de este concepto, y nunca se encuentra relevancia con lo pasado por la consulta

Figura 3.17

```
"woman"
value: 0.20488907396793365
```

Como se mencionó en el *capítulo 2*, el algoritmo de similitud está diseñado para que, a mayor descripción de la imagen, sean más precisos los resultados. Como resultado de esto, se muestran algunas consultas muy detalladas en cuanto a posición, con el objetivo de recuperar imágenes de forma más precisa. En este caso, a una consulta detallada solo mejora los resultados si encuentra las descripciones específicas pasadas.

Por ejemplo, para la consulta *in the center of image there are tow dogs sitting in a black and white couch. A dog beside to a dark gray dog. below the dog there are a black couch. right to a couch there are a black statue of buda. On top of couch are a white window.*, se obtienen los siguientes resultados.

```
image_1: 0.19233258068561554
image_2: 0.27561962241802707
image_3: 0.8454822663938976
image_4: 1.6529658667220515
```

```
image_5: 0.18111929297447205
image_6: 0.17510709166526794
image_7: 0.19716928899288177
image_8: 0.2104945033788681
image_9: 0.5446422211089514
image_10: 0.560958578905552
image_11: 0.2174830138683319
image_12: 0.21318000555038452
image_13: 0.24322502315044403
image_14: 0.5096454837731963
image_15: 0.23101291060447693
```

Se observa que la imagen recuperada„ figura 3.18, posee una similitud muy alta, la misma, para ojos humanos, es bastante cercana a lo descrito en la consulta de entrada. La siguiente imagen cercana en similitud también se acerca a lo descrito en la consulta.

Estos resultados refuerzan la eficacia del algoritmo de similitud diseñado para mejorar la precisión de los resultados de recuperación de imágenes. Al proporcionar una descripción detallada de la imagen, el algoritmo puede identificar mejor las características clave y las relaciones espaciales, lo que resulta en una recuperación de imágenes más precisa.

Figura 3.18

3.1.2. Procesamiento de imágenes usando descripciones

En esta sección, se presentan los resultados de la variante que utiliza descripciones, y se analiza la ineficiencia de la misma. Como resultado de esto, el uso de descripciones sin imágenes suele ser igualmente menos eficiente.

Esta variante presenta un problema fundamental: los textos de concepto cernanos suelen ser muy similares. Por ejemplo, en las muestras e imágenes, para las imágenes y consultas relacionadas a perros y gatos, las relevancias alcanzadas son muy similares, dado que este caso los conceptos **dog** y **cat** tienen una relación de similitud de 0,91283133. Esto implica que las imágenes que tengan **dog** como descripción serán altamente relevantes con los textos que tengan **cat** en su contenido, y en estos casos son muy similares. Veamos tres de las consultas con sus respectivas similitudes hacia las imágenes.

```
"a cat on left side of dog":
image_1: 1.1524357352905104
image_2: 1.3433827930426756
image_3: 1.2726673896730163
image_4: 1.2426372272442288
image_5: 0.6868996450696936
image_6: 0.6899906356873706
image_7: 0.6601000538778263
image 8: 0.5863789420884641
image_9: 1.3568643812126333
image_10: 1.3029270330526308
image_11: 1.228132925709438
image_12: 1.1415868365589665
image_13: 1.2949859036677203
image_14: 1.309315788897958
image_15: 1.269946229080809

"there is a dog on left of a cat:
image_1: 1.2083463593593047
image_2: 1.3538045434242947
image_3: 1.244656483332271
image_4: 1.25616415288856
image_5: 0.6519800801549985
image_6: 0.7511959871556898
image_7: 0.6586201567797191
image 8: 0.5911412861692614
```

```
image_9: 1.32958947197954
image_10: 1.3701337275645442
image_11: 1.3088403469722087
image_12: 1.2220715210928654
image_13: 1.2619162228862222
image_14: 1.4098256637597801
image_15: 1.3314236030203248

"vase of roses in top of a bedside table":
image_1: 0.4997751770591399
image_2: 0.5835944445734702
image_3: 0.5381005093464214
image_4: 0.5806767925529962
image_5: 0.5807832418280381
image_6: 0.530520117014973
image_7: 1.4137167510662856
image 8: 0.45439535468733877
image_9: 0.5458514239447663
image_10: 0.5084185778157145
image_11: 0.57788808475214
image_12: 0.5664107286655431
image_13: 0.5993313035010336
image_14: 0.5535414798660193
image_15: 0.5891194603734591
```

En la consulta .ª cat on left side of dog", se esperaría que la imagen [13] sea más relevante que la imagen 14. Sin embargo, en este caso no se obtuvieron buenos resultados. Para lograr mejores resultados, se puede probar más variaciones de la importancia del texto y el uso de otros modelos descriptivos.

Por otro lado, cuando estamos trabajando con conceptos distintos como rosas y mesas, la diferencia de similitud es notable en cuanto a los resultados recuperados.

3.1.3. Cálculos de posición

En esta sección, se pretende mostrar algunos ejemplos de cómo el uso de las posiciones en los conceptos impacta en el resultado de la similitud. Para ello, se observará la imagen *Image 10* del *ImageSet* en la figura 3.19, a la que se le pasan textos en cada caso. Se verá cómo se comportan los cambios de vecinos y de posiciones usados.

Para comprender los ejemplos próximamente presentados, se explica la leyenda que usan los mismos.

- `TEXT: "texto"` indica que "texto.[es] el texto pincipal del *Text*.
- `POS: (x,y)` indica la posición en la cual se encuentra la *Image* analizada.
- `label: ["neigbords"]` este label esta dado por cada una de las etiquetas de los vecinos de un *Text*, solo mostrando las que al menos, poseen un vecino; cada elemento que corresponde a esta correponde a un vecino.
- `SIMILARTY` indica que lo próximo a mostrar serán similitudes.
- `cosine: value` indica el valor de la similitud del coseno.
- `cosine_pos: value` indica el valor de la similitud del coseno teniendo en cuenta la posición de la imagen.
- `using region: value` indica el valor de similiud por posición y coseno, además, teniendo en cuenta los vecinos de cada concepto.

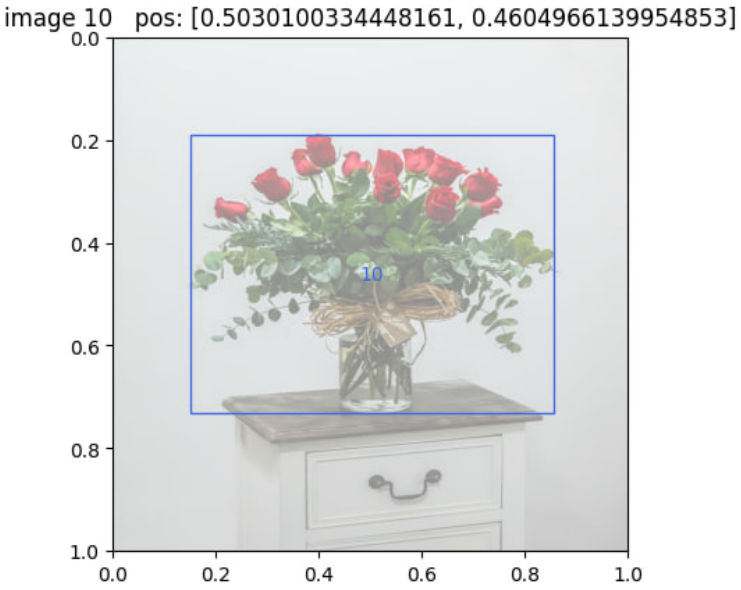

Figura 3.19

```
TEXT: A bouquet of roses
POS: (0.5, 0.5)
```

```
s:[a table]
SIMILARITY
    cosine: 0.2806638479232788
    cosine_pos: 0.31585035808965767
    using region relation: 0.410122760
----------------------------------------

TEXT: a roses
POS: (0.5, 0.5)
s:[a bedside table]
SIMILARITY
    cosine: 0.26832765340805054
    cosine_pos: 0.301967588
    using region relation: 0.3985802
----------------------------------------

TEXT: A bouquet of roses
POS: (0.5, 0.5)
s:[a gray and black bedside table]
SIMILARITY
    cosine: 0.2806638479232788
    cosine_pos: 0.3158503580896
    using region relation: 0.4276975
----------------------------------------

TEXT: A bouquet of roses
POS: (0.5, 0.5)
s:[a gray and black bedside table, a white drawer]
SIMILARITY
    cosine: 0.2806638479232788
    cosine_pos: 0.31585035808965767
    using region relation: 0.532067
```

El uso de posiciones globales puede mejorar el resultado de la similitud, en estos casos se observa cómo mejora la relevancia de la imagen si la posición coincide con la dada. Por otro lado, el uso de regiones, o de vecindades posicionales, puede tanto mejorar los resultados de la similitud como empeorarlos. En este ejemplo se aprecia entonces la influencia de las posiciones para el resultado del cálculo de similitud.

Veamos un ejemplo, usando la misma leyenda, de un caso donde al no detectar ningún vecino que cumpla con lo esperado en una posición, la similitud por regiones

disminuye entonces su valor. La imagen evaluada es la segmentación 17 de la figura 3.20.

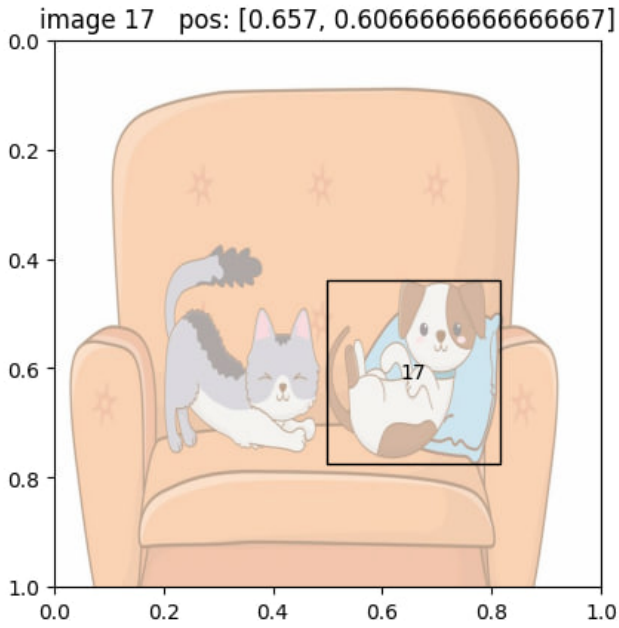

Figura 3.20

```
TEXT: a dog laying on a blue pillow
POS:(0.5, 0.5)
w:[a apple]
SIMILARITY
    cosine: 0.25085294246673584
    cosine_pos: 0.28454447699
    using region relation: 0.2739276
```

3.1.4. Observaciones

Como desarrollo principal, se buscaba lograr recuperar imágenes a partir de una consulta en lenguaje natural. Sin embargo, se observó que la metodología es aplicable para recuperar imágenes utilizando otras imágenes como consulta. En este caso, se toma en cuenta las relaciones posicionales de cada objeto en cuestión junto con sus conceptos agrupados en embeddings. El resultado de la recuperación de imágenes debe ser ajustado para lograr una recuperación precisa. Una vez se ajusten los parámetros para la recuperación imagen-imagen, será posible realizar dicha tarea. Nótese que este sistema de recuperación imagen-imagen basado en posciones y conceptos, consitituye

un enfoque distinto al utilizado, por lo general, en la búsqueda de imágenes con imágenes como consulta.

3.2. Detalles de implementación

La sección de detalles de implementación proporcionará una explicación detallada sobre la arquitectura del proyecto. Esta explicación cubrirá aspectos como la escalabilidad del sistema y los detalles específicos de implementación que son relevantes para el proyecto.

3.2.1. Estructura

El proyecto se estructura en distintas secciones. Cada sección del proyecto tiene un propósito definido y se explica a continuación, destacando su uso, funciones y objetivos.

Modelos descriptivos

En la sección de modelos de descripción, se realiza la implementación de cada uno de los modelos integrados. Aquí también se pueden agregar los modelos futuros que se deseen incorporar. Actualmente, los modelos BLIP[7] y BLIP2[5] están integrados, cada uno de ellos carga su contenido desde la biblioteca Transformers de Hugging Face. La estructura común para cargar modelos desde esta biblioteca se muestra a continuación, tomando como ejemplo el modelo BLIP[7].

```
1 from transformers import BlipProcessor, BlipForConditionalGeneration
2 BLIP.PROCESSOR = BlipProcessor.from_pretrained("Salesforce/blip-
    image-captioning-large")
3 BLIP.MODEL = BlipForConditionalGeneration.from_pretrained("
    Salesforce/blip-image-captioning-large").to("cuda")
```

La función `from_pretrained()` de la biblioteca *Transformers* permite cargar un modelo preentrenado desde Hugging Face 1. Esta función puede cargar el modelo a partir de una identificación de modelo, un directorio local donde se encuentran los pesos del modelo guardados, o un archivo de estado PyTorch.

Modelos de segmentación

De manera similar a la sección de modelos descriptivos, existe una sección dedicada a los modelos de segmentación. En esta sección, se integra el modelo SAM[6], que se utiliza en el código de tal manera que permita manipular las salidas y parámetros del modelo sin necesidad de alterar el código. Para lograr esto, se dispone de una clase

SAM-Environment en la sección de entorno, la cual contiene todos los parámetros modificables de este modelo. De esta forma, si se desea cambiar su comportamiento, se tiene la opción de hacerlo.

La codificación de SAM[6] permite solicitar a las segmentaciones imágenes de tipo "box.º de tipo "mask". El primer tipo devuelve todos los píxeles encontrados en el cuadro de segmentación, mientras que el segundo devuelve los píxeles de la máscara de segmentación con el fondo blanco. La elección de qué tipo de segmentación utilizar queda a discreción del procesador de imágenes.

Generadores de Embeddings

En esta sección se integran los modelos de generación de embeddings. En el caso particular de este trabajo, se ha integrado el modelo CLIP[14]. Al igual que en los casos anteriores, si se necesita integrar otros modelos de generación de embeddings, debe hacerse dentro de esta carpeta.

Entorno(*Environment*)

En esta sección se concentra todo lo relacionado con el entorno. Desde aquí, se pueden realizar modificaciones en todos los parámetros del sistema que se desee. Por ejemplo, si se requiere que el código utilice descripciones en el cálculo de similitudes, se debe ajustar la variable correspondiente en la clase mencionada. En el presente ejemplo, se debe cambiar la variable *USE-CAPTION-MODEL = False* a *True* en la clase *ImageEmbeddingEnv*. Además, se proporciona un ejemplo de una clase en el entorno, ya que este es el código potencialmente modificable.

```
1    class ImageEmbeddingEnv:
2    '''
3    ## ImageEmbeddingEnv
4    ### Variables
5    - 'USE_CAPION_MODEL' Indica que se usara un modelo de
     descripciones de im\'agenes para usar tambien emedding de la
     descrpcion en la imagen al calcular similitud
6    - 'CAPTION_MODEL' Este es el modelo que se usara para describir
     la imagen
7    - 'CAPTION_IMPORTANCE' Importancia del caption, define que tan
     importante sera el caption de una imagen para la similitud. '=1'
     indica que tiene igual imporatancia que la imagen
8    - 'POS_UMBRAL': Importancia dada a la distancia entre las
     posiciones de las im\'agenes, mientras menor valor mayor
     importancia se le da a las posiciones
9    - 'PRIMARY_POW': Elevacion a la ptencia del eje principal a la
     hora de calcular distancias
10   - 'SECUNDARY_POW': Elevacion a la ptencia del eje no principal a
     la hora de calcular distancias
```

```
11      - ‘MAX_DISTANCE‘: Distancia maxima que puede haber entre dos im
    \’agenes para ser consideradas vecinas
12      - ‘KEY_IMAGES‘: Esto es el tipo de imagen que usara como
    principal. (box or mask).
13      ### Funciones
14      - ‘max_similarity()‘: Devuelve la maxima similitud que se puede
    alcanzar en la distancia entre dos im\’agenes.
15      ’’’
16
17      USE_CAPION_MODEL = False
18      CAPTION_MODEL =BLIP
19      CAPTION_IMPORTANCE = 1
20      PRIMARY_POW = 1.2
21      SECUNDARY_POW = 0.7
22      MAX_DISTANCE = 1
23      POS_UMBRAL = 0.85
24      KEY_IMAGES = ’box’
25
26      def max_similarity():
27          return ImageEmbeddingEnv.MAX_DISTANCE/ImageEmbeddingEnv.
    POS_UMBRAL
```

Característcas(*features*)

Esta es la sección más crucial, donde se implementan las clases que abordan los conceptos fundamentales del trabajo: Image, ImageSet, Text, TextSet. Idealmente, esta clase no debería ser modificada, a menos que se desee asignar nuevas funcionalidades a estas clases. Cada clase tiene sus propios procesos internos, pero su principal objetivo es almacenar características en una clase determinada. La clase Image es una subclase que se encarga exclusivamente de crear funciones para el cálculo de similitudes de vecindad. El resto de los componentes de ambas clases dependen únicamente de procesos externos, no implementados dentro de la clase en sí. Por ejemplo, el proceso de segmentación de imágenes y el análisis sintáctico del texto se llevan a cabo en módulos específicos encargados de esta función en particular.

Gramática

En esta sección se encuentran los tokenizadores y analizadores sintácticos que procesan las consultas en forma de texto. Cada uno de estos está diseñado para ser escalable, lo que permite un futuro mejor procesamiento y comprensión del lenguaje.

Recuperación de información

En esta sección se ubica el motor de recuperación de información y la base de datos. Ambas son clases que implementan el comportamiento necesario para recuperar y guardar la información. El sistema de recuperación tiene dos funciones principales: una función de ranking y una función de búsqueda. La primera se enfoca en calcular la similitud entre una consulta y todas las imágenes de la base de datos utilizando la distancia del coseno. Se establece un umbral de tamaño máximo del ranking y se guardan las n mejores imágenes. Luego, las imágenes en ese ranking son comparadas con la consulta utilizando el algoritmo principal propuesto. Este proceso se divide en dos partes, ya que calcular la similitud del coseno es considerablemente más rápida que el cálculo utilizando el algoritmo principal. Para calcular la similitud de 10,000 imágenes, el algoritmo principal tarda, aproximadamente, 42 segundos usando una GPU v100, mientras que la similitud por coseno tarda ese tiempo para 1,000,000 de imágenes.

La clase data tiene como principal objetivo acumular la información, guardarla en disco y cargarla desde el disco. Para ello cuenta con funciones para guardar y cargar desde una dirección, así como funciones para agregarle nuevos datos desde el código y, teniendo estos, poder guardarlos. La información guardada y cargada por la base de datos será, de cada imagen, el embedding, la posición, el nombre, referencia a los vecinos y similitud de cercanía a estos, y límites de una imagen

Similitud

En esta sección se implementan todas las funciones de similitud que se utilizarán para el cálculo. Si se desea agregar una nueva función de similitud, debe hacerse en esta clase.

Main

Finalmente, se llega a la función global, desde la cual se utilizan los demás módulos. Esta sección está contenida por un archivo *.ipynb* desde el cual se llama y ejecuta cada función deseada. El uso de un notebook es debido a la necesidad de ejecutar el proyecto en una máquina virtual de Google Colab.

Conclusiones

En este estudio, se presentó una metodología para recuperar imágenes a partir de consultas en forma de lenguaje natural. Esta metodología se estructuró en tres etapas clave: procesamiento de imágenes, procesamiento de consultas y recuperación de información. Se puso especial énfasis en el uso de embeddings y posiciones como características más relevantes.

Los resutados de esta metodología fueron satisfactorios, logrando una notable mejora en la relevancia al utilizar consultas más precisas que indican relaciones posicionales de la imagen. Se llegó a la conclusión de que el uso de características como los embeddings generados por un lenguaje multimodal, garantiza una adecuada recuperación de las imágenes en términos de comprensión de conceptos. La combinación de estas características con la detección de objetos tanto en la imagen como en el lenguaje natural, logarn un buen resultado para el objetivo de recuperar imágenes a partir de consultas precisas.

Recomendaciones

Para futuros trabajos, se recomienda implementar una serie de funciones para mejorar el comportamiento actual. El procesamiento del lenguaje en el trabajo se centra en los patrones de descripción de posiciones, como mejora a esto, una propuesta sería preprocesar el texto para organizarlo de manera más adecuada para el cumplimiento de la tarea deseada. En esta misma fase de procesamiento de texto, otra posible implementación es el uso de modelos LLM (Large Language Models) para procesar la consulta y analizarla con un modelo tan potente como los que estarán disponibles en el futuro.

En el campo del procesamiento de la imagen, se puede implementar otros reconocimientos de conceptos en la imagen, no solo reconocimiento posicional, sino que podría implementarse reconocimiento de colores, por ejemplo. Otro punto a destacar sería establecer parámetros y objetivos para ampliar este sistema, de esta forma, que no se centre solo en recuperación de texto a imagen, sino de imagen a imagen.

Bibliografía

[1] Yang Bai y col. «Sentence-level Prompts Benefit Composed Image Retrieval». En: *arXiv preprint arXiv:2310.05473* (2023) (vid. pág. 7).

[2] Francesco Baldrati y col. «Effective Conditioned and Composed Image Retrieval: Combining CLIP-Based Features». En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. de 2022. URL: https://openaccess.thecvf.com/content/CVPR2022/papers/Baldrati_Effective_Conditioned_and_Composed_Image_Retrieval_Combining_CLIP-Based_Features_CVPR_2022_paper.pdf (vid. pág. 6).

[3] David Beazley. *Sly Lex Yacc*. 2016. URL: https://sly.readthedocs.io/en/latest/sly.html (vid. pág. 22).

[4] Jacob Devlin y col. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. URL: https://arxiv.org/abs/1810.04805 (vid. pág. 26).

[5] Hugging Face. *BLIP-2*. 2022. URL: https://huggingface.co/docs/transformers/main/model_doc/blip-2 (vid. pág. 45).

[6] Hugging Face. *SAM Documentation*. https://huggingface.co/docs/transformers/model_doc/sam. 2022 (vid. págs. 13, 45, 46).

[7] Huggingface. *Huggingface BLIP*. https://huggingface.co/models?filter=blip. 2022 (vid. pág. 45).

[8] Google Inc. *Attention - TensorFlow API Documentation*. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Attention (vid. pág. 27).

[9] Caiming Xiong y Steven Hoi. Junnan Li Dongxu Li. «BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation». En: *Salesforce Research, deepai.org* (2022) (vid. págs. 2, 3, 19, 20).

[10] Zoumana Keita. *Text-to-Image and Image-to-Image Search Using CLIP*. 2023. URL: https://www.pinecone.io/learn/clip-image-search/ (vid. pág. 6).

[11] Alexander Kirillov y col. «Segment Anything». En: *arXiv:2304.02643* (2023) (vid. págs. 7, 37).

[12] Haotian Liu. Chunyuan Li. Qingyang Wu. Yong Jae Lee. «Visual Instruction Tuning». En: *Microsoft Research* (2023) (vid. págs. 2, 3, 19).

[13] Junnan Li y col. «BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models». En: *arXiv preprint arXiv:2301.12597* (2023) (vid. págs. 8, 19).

[14] OpenAI. «CLIP: Connecting text and images». En: (2021). URL: `https://openai.com/research/clip` (vid. págs. 6, 7, 19, 20, 30, 31, 34, 35, 37, 46).

[15] OpenAI. *CLIP: OpenAI's Official GitHub Repository*. `https://github.com/openai/CLIP`. 2021 (vid. pág. 35).

[16] OpenAI. *GPT-4: An AI Multimodal That Claims to Be of Latest Generation*. 2023. URL: `https://techcrunch.com/2023/03/14/openai-launches-gpt-4-a-multimodal-ai-that-claims-to-be-of-latest-generation/` (vid. pág. 3).

[17] Ashish Vaswani. Noam Shazeer. Niki Parmar. Jakob Uszkoreit. Llion Jones. Aidan N. Gomez. Lukasz Kaiser. Illia Polosukhin. «Attention Is All You Need». En: *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA,* (2017) (vid. págs. 2, 27).

[18] Alec Radford y col. *GPT-3: Language Models are Unsupervised Multitask Learners*. `https://arxiv.org/abs/2005.14165`. 2020 (vid. págs. 6, 26).

[19] Alec Radford y col. *Language Models are Unsupervised Multitask Learners*. 2017. URL: `https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf` (vid. págs. 1, 6, 26).

[20] Colin Raffel y col. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2019. URL: `https://arxiv.org/abs/1910.10683` (vid. pág. 26).

[21] Alec Radford. Jong Wook Kim. Chris Hallacy. Aditya Ramesh. Gabriel Goh. Sandhini Agarwal. Girish Sastry. Amanda Askell. Pamela Mishkin. Jack Clark. Gretchen Krueger. Ilya Sutskever. «Learning Transferable Visual Models From Natural Language Supervision». En: *OpenIA* (2021) (vid. págs. 2, 6).

[22] Zhengyuan Yang. Linjie Li. Kevin Lin. Jianfeng Wang. Chung-Ching Lin. Zicheng Liu. Lijuan Wang. «The Dawn of LMMs:Preliminary Explorations with GPT-4V(ision)». En: *Microsoft Corporation* (2023) (vid. págs. 2, 3, 19, 20).

[23] Yihan Zeng y col. «CLIP^2: Contrastive Language-Image-Point Pretraining from Real-World Point Cloud Data». En: *arXiv preprint arXiv:2303.12417* (2023) (vid. pág. 20).