

Universidad de La Habana
Facultad de Matemática y Computación



Título de la tesis

Autor:

Nombre del autor

Tutores:

Nombre del primer tutor

Nombre del segundo tutor

Trabajo de Diploma
presentado en opción al título de
Licenciado en (Matemática o Ciencia de la Computación)

Fecha

github.com/username/repo

Dedicación

Agradecimientos

Agradecimientos

Opinión del tutor

Opiniones de los tutores

Resumen

Resumen en español

Abstract

Resumen en inglés

Índice general

| | |
|--|----|
| Introducción | 1 |
| 1. Estado del Arte | 6 |
| 2. Propuesta | 7 |
| 3. Detalles de Implementación y Experimentos | 8 |
| Conclusiones | 9 |
| Recomendaciones | 10 |

Índice de figuras

Ejemplos de código

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 avance 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, 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 archi-

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[**transformers**], 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[**clip**], entrenado con 400 millones de imágenes y texto proveniente de Internet, lo que le permite comprender la similitud existente entre textos e imágenes.

Motivación

En el contexto de los avances recientes en el campo de la visión artificial, se ha abierto la posibilidad de automatizar el etiquetado de imágenes. Los sistemas de recuperación de información más prominentes, como Google, actualmente recuperan imágenes utilizando etiquetas asignadas manualmente o palabras clave en la web asociada a la imagen en cuestión. La perspectiva de transferir esta labor manual a máquinas resulta atractiva, alineándose con las tendencias actuales de la inteligencia artificial y el aprendizaje automático, con el potencial de transformar la manera en que gestionamos y organizamos las imágenes.

El desarrollo de un sistema que se dedique específicamente a etiquetar imágenes automáticamente y un Sistema de Recuperación de Imágenes para recuperarlas representa un campo poco explorado. El enfoque de hacer esto manual tiene algunas limitaciones. Además, el proceso de asignación de etiquetas puede ser laborioso.

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[**blip**], LLaVA[**llava**] y GPT-4V[**gpt-4v**].

CLIP (Contrastive Language-Image Pretraining) 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), 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) 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, o Modelo de Visión de GPT-4, es una extensión del popular modelo de lenguaje GPT-4 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.

Problemática

Aunque estos modelos de visión y lenguaje poseen una notable capacidad para analizar imágenes en relación con el texto, se enfocan en tareas específicas que difieren de la recuperación de información. A pesar de ello, ofrecen resultados satisfactorios que pueden sentar las bases para abordar de manera efectiva el campo de la recuperación de imágenes.

En la actualidad, los sistemas de recuperación de imágenes se basan en el etiquetado manual, y los sistemas de búsqueda no se centran lo suficiente en la recuperación de imágenes en sí. En consecuencia, no se dedican a crear un sistema completo de etiquetado y consultas capaz de recuperar imágenes desde descripciones detalladas y precisas. En su lugar, se utilizan sistemas de recuperación de información menos precisos para este ámbito, ya que su objetivo principal suele ser obtener información relacionada con las imágenes, no las propias imágenes.

Ergo, el proceso de etiquetado de imágenes exclusivamente con el propósito de recuperarlas con información detallada deja margen de mejora. En este trabajo, se busca abordar esta problemática, buscando alcanzar un etiquetado que satisfaga los objetivos planteados y, al mismo tiempo, un procesamiento de las consultas que se ajuste al tipo de etiquetado empleado.

Objetivos

Objetivo general

El objetivo de este trabajo consiste en desarrollar un sistema automatizado de etiquetado de imágenes y un sistema de recuperación altamente preciso que utilice las etiquetas asignadas, 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 al formato de las consultas más frecuentemente utilizadas en las búsquedas de imágenes.

Objetivos específicos

- Emplear modelos de aprendizaje automático entrenados con extensas cantidades de datos especializados en la descripción de imágenes.
- Diseñar una arquitectura escalable que permita la incorporación de nuevos modelos de visión artificial a medida que este campo se expande con el tiempo.
- Realizar reentrenamiento de los modelos base utilizados con el fin de mejorar la eficiencia en la descripción de las imágenes.
- Utilizar modelos de segmentación de imágenes para obtener descripciones más detalladas, analizando la imagen no solo en su totalidad, sino también por segmentos.
- Integrar modelos de visión con modelos de lenguaje, como CLIP, para verificar y seleccionar la descripción más precisa de la imagen en cuestión.
- Procesar las descripciones finales proporcionadas para crear un sistema de tokens que se ajuste al formato de consultas más utilizado, 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.
- Realizar un análisis exhaustivo de cada modelo utilizado y plantear soluciones que, por casos de limitaciones de recursos o falta de información, su implementación práctica no es viable.

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.

En el capítulo 2 se lleva a cabo un estudio detallado de cada uno de los modelos empleados para abordar el problema, comparando sus características y eficiencia.

El capítulo 3 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 y el proceso de reentrenamiento.

En el capítulo 4 se recopilan los detalles de la implementación y se abordan los desafíos surgidos debido a la limitación de hardware y acceso a información.

El capítulo 5 se enfoca en la comparación de diversas soluciones, variando los modelos utilizados en cada una de ellas, así como los hiperparámetros modificados. También se evalúa el rendimiento del modelo en su etapa inicial y después de haber sido reentrenado.

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

Capítulo 1

Estado del Arte

Capítulo 2

Propuesta

Capítulo 3

Propuesta

La propuesta para el sistema de recuperación de información de imágenes basado en contenido comprende dos fases principales y una fase final de sistema de recuperación. La primera fase se enfoca en el procesamiento y obtención de características de una imagen. Se emplean embeddings (vectores multi-dimensionales) como contextos, con el fin de obtener similitud entre los mismos con uso de modelos entrenados para ello.

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.

La segunda etapa se encarga de procesar el texto proporcionado por la consulta de los usuarios. Esta etapa también está compuesta por más de una variante. Está diseñada para ser escalable, lo que permite mejorar significativamente el procesamiento del texto como lenguaje natural.

Fase de procesamiento de imagen

Esta fase se divide en variantes que, a su vez, poseen una base común en su metodología, cambiando solo los conceptos que se emplean en cada una de ellas. La metodología principal está dada por una serie de procesos que serán explicados en esta sección. Para comprender la metodología se definen dos conceptos fundamentales.

El primero es el concepto de **Image**. Una *Image* se define como un conjunto de características que se extraen de una imagen, las cuales son: un embedding, una descripción, una posición, una serie de vecinos por posición relativos 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 de ambos ejes coordenados.

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 sola imagen, una imagen general. Cada una de estas *Images* constituye una segmentación de la imagen general. En resumen, *ImageSet* es el conjunto de características de una imagen subdividida en segmentaciones relacionadas.

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 es segmentada; para realizar esta segmentación se utilizan modelos de segmentación de imágenes previamente entrenados. Este trabajo, en particular, utiliza el modelo de segmentación SAM (*Segment Anything Model*)[REF]; dado la escalabilidad del código, este modelo es perfectamente sustituible por otro modelo que se encargue de realizar la tarea de segmentar imágenes, en trabajos futuro se pueden usar modelos más potentes o más rápidos. El tamaño de las segmentaciones, así como los hiperparámetros del modelo SAM, son modificables. En caso de ser necesario, los parámetros del modelo pueden ajustarse a la problemática que ha surgido.

Terminada la tarea de segmentación 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 pérdida 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 cuando una imagen posee una relevancia alta para el universo. Luego, una imagen será 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 feature de la imagen el 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 x e y. 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.

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 estar en la posición superior de la imagen en cuestión.
- **s**: Indica la cercanía a la posición inferior relativa a la imagen. Dice que tan cerca está un vecino de estar en la posición inferior de la imagen en cuestión.
- **e**: Indica la cercanía a la posición relativa derecha a la imagen. Dice que tan cerca está un vecino de estar en la posición derecha de la imagen en cuestión.
- **w**: Indica la cercanía a la posición relativa izquierda a la imagen. Dice que tan cerca está un vecino de estar en la posición superior de la imagen en cuestión.
- **beside**: Indica la cercanía a la posición relativa derecha o izquierda a la imagen. Dice que tan cerca está un vecino de estar en un lado de la imagen en cuestión.
- **ne**: Indica la cercanía a la posición relativa derecha superior a la imagen. Dice que tan cerca está un vecino de estar en la posición a 45° de la imagen en cuestión.
- **nw**: Indica la cercanía a la posición relativa izquierda superior a la imagen. Dice que tan cerca está un vecino de estar en la posición a 135° de la imagen en cuestión.
- **se**: Indica la cercanía a la posición relativa derecha inferior a la imagen. Dice que tan cerca está un vecino de estar en la posición a 315° de la imagen en cuestión.
- **sw**: Indica la cercanía a la posición relativa izquierda inferior a la imagen. Dice que tan cerca está un vecino de estar en 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 posición-tipo de posición, donde se indica que tan similar es la posición de este con respecto a la imagen y al tipo de posición en cuestión. Por ejemplo(ver imagen).

La definición de la similitud posicional ha sido un tema muy cuestionable, dado que se busca similitud entre lenguaje natural e imágenes. Definir entonces que tan similares son las posiciones 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.

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. Se podría decir que el edificio está a su derecha, porque en realidad lo está, en su base coinciden. También se podría decir que el edificio está arriba, en la posición superior del objeto central. Y por supuesto, el edificio parece estar en la parte superior derecha del sujeto del centro. Se interpreta cada una de estas posiciones, pero queda definir cuál es mejor, definir que la posición derecha será mejor que la posición derecha arriba.

En respuesta a ello 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 variables:

- e = Distancia euclideana entre las imagenes. Esta se calcula como la distancia mínima entre los limites de las *images*.
- a = Angulo entre el centro del **image** y el centro del **image** vecino.
- b = El ángulo correspondiente a la posicion dada, con respecto a la imagen, por ejemplo para la posicion derecha superior se espera que el angulo este cerca de los 45° .
- 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:

$$sim_{pos} = (1 - e) * (1 - |a - b|) * u$$

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

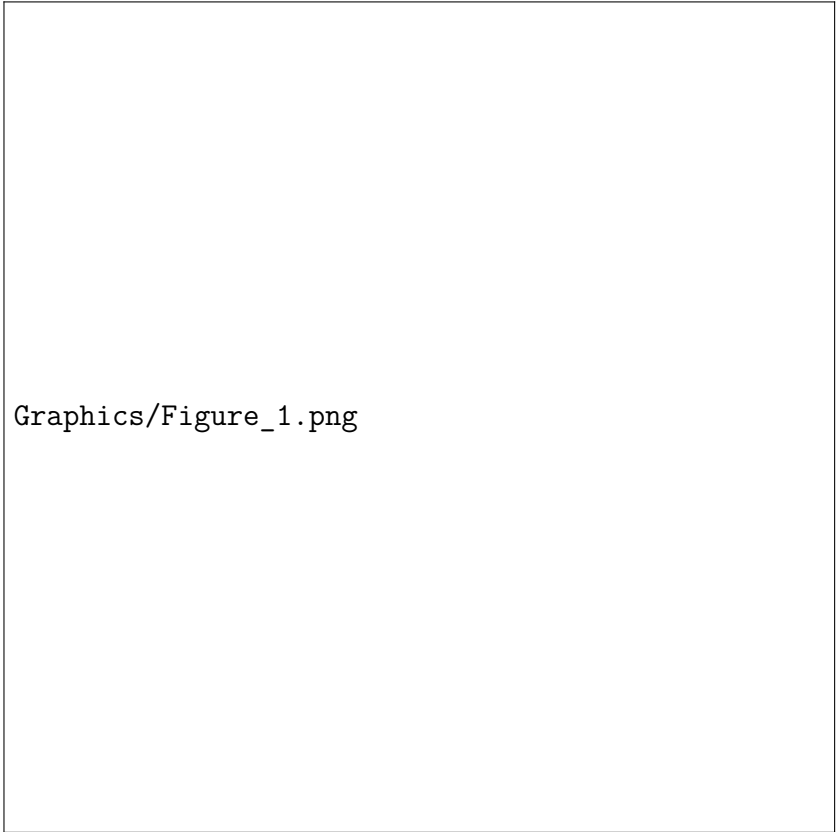
se consigue que sean más importantes las posiciones antes que las escalas de los objetos que se analizan. Ahora un objeto con mucho más escala que otro se ajustará al pequeño para definir sus posiciones relativas.

Alternativa de pesos en los ejes coordenados.

Esta alternativa calcula la distancia entre los objetos tanto en el eje vertical como en el horizontal. Se le da valor de importancia al eje principal que se analiza y al eje no principal. Por ejemplo, si se está analizando que se tiene a la derecha o a la izquierda, entonces el eje principal será el eje horizontal y el secundario será el vertical, teniendo esto en cuenta se ajustan los parámetros a conveniencia de tal forma que que acerque a lo mejor posicionalmente descrito posible. Cada uno de los parámetros correspondientes a esta función:

- Multiplicador de posiciones relativas: Este parámetro se usa para ajustar con un multiplicador los valores de las posiciones relativas entre objetos.
- 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.
- 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.

A continuación se muestran valores de similitud por vecindad para dos métricas distintas. Estos valores son ajustables cambiando valores de parámetros indicados. La etiqueta define la posición a la cual es similar un valor, cada valor está compuesto por dos elementos, la referencia a la imagen y el valor de semejanza de sus posiciones con relación a la zona de la imagen que le corresponde.



Graphics/Figure_1.png

valign=c, center

Ángulo entre los centros y Distancia entre los extremos

```

1  w: [(image 1, 0.22916)]
2  e: [(image 4, 0.75963), (image 5, 0.26565)]
3  n: [(image 1, 0.56386), (image 2, 0.71565), (image 4, 0.15595)]
4  s: [(image 5, 0.63434)]
5  nw: [(image 1, 0.45833)]
6  ne: [(image 4, 0.28072)]
7  sw: []
8  se: [(image 5, 0.53130)]
9  in: []
10 beside: [(image 1, 0.2062), (image 4, 0.6836), (image 5, 0.2390)]
11 next: [(image 5, 0.6343), (image 1, 0.5638), (image 2, 0.7156), (
    image 4, 0.7596)]

```

Alternativa de pesos en los ejes coordenados

```

1  w: [(image 1, 0.47344711826630503)]
2  e: [(image 4, 0.9369042655519807), (image 5, 0.5694883797500658)]
3  n: [(image 1, 0.6555178357675728), (image 2, 0.9369042655519806), (
    image 4, 0.5694883797500657)]
4  s: [(image 5, 0.9369042655519807)]
5  nw: [(image 1, 1.128964954033878)]
6  ne: [(image 4, 1.5063926453020464)]

```

```

7  sw: []
8  se: [(image 5, 1.5063926453020464)]
9  in: []
10 beside: [(image 1, 0.47344711826630503), (image 4,
    0.9369042655519807), (image 5, 0.5694883797500658)]
11 next: [(image 5, 1.5063926453020464), (image 1, 1.128964954033878),
    (image 2, 0.9369042655519806), (image 4, 1.5063926453020464)]

```

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, esta más extendida en el eje y, entonces su ángulo relativo a otra varía según la su escala.

Teniendo el modelo o metodología principal de esta fase. Sobre la base de este, se proponen tres variantes para ser utilizadas como conceptos entre los cuales, poder hallar relevancia mediante cálculos de similitud.

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 sobre 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 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, BLIP2, LLaVA y GPT4-V. 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 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.

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 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 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, 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, el cual es muy potente en este campo, como también se desconoce que tan preciso sería usar CLIP-2 para generar embeddings, ambos modelos son de código cerrado de OpenAI.

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.

Fase de procesamiento de texto

Esta fase 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 fase 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 fase del procesamiento del texto de la consulta, a

diferencia de la fase 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 fase 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 texto de la siguiente forma:

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

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

El segundo concepto es text feature. Un text feature se define como un conjunto de texts donde cada uno de estos son subdivisiones de un solo texto general, además también posee un text que hace referencia al texto original. En pocas palabras, text feature es el conjunto de textos que conforman el texto original, separados por puntos o por posiciones indicadas con lenguaje natural. Esta fase 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 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[REF], 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 está 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 será 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 si solo si está contenida en un patrón que constituye una regla gramatical en su completitud, de esta forma solo se consideran tokens los patrones que se usan para describir relaciones posicionales. Luego de la fase de tokenización sigue la fase de análisis sintáctico, el cual se hace cargo de parsear todo el texto por relaciones posicionales en caso de que se especifiquen las mismas. Este análisis semántico o parser, está dividido en dos subetapas, dos parsers distintos: un análisis para las relaciones globales de los objetos con respecto a la imagen, y otro análisis para las relaciones posicionales.

*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 text feature. 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 “IS text ON pos OF IMAGE”, de esta forma se obtiene un text con string igual a “dog next to a ball” y pos, 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, 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 y será procesado como token.

```

1 keywords = {
2     ON = [ in, on, at, near, find ],
3     OF = [ of ],
4     AND = [ and ],
5     IS = [ is, are, there's, find ],
6     POS = [ left, right, buttom, bottom, top, down, up, lower,

```

```

    center, middle, corner ],
7     POSITION = [ position, side, location ],
8     IMAGE = [ image, picture, photo ],
9     NEXT = [ next, near ]
10    }

```

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, la gramática de las relaciones espaciales se considera de mayor importancia. Este análisis se basa en la identificación de patrones gramaticales que permiten expresar relaciones espaciales en el texto. Veamos como han sido definidas sus reglas patrones.

```

1  WORD ::= WORD
2        | Any
3
4  pos  ::= POS
5        | POS POSITION
6
7  text ::= WORD
8        | [text]+
9
10 relation ::= ON pos IS text
11            | ON pos OF IMAGE IS text
12            | ON pos text
13            | IS text ON pos OF IMAGE
14            | IS text ON pos ","
15            | IS text ON pos !OF
16            | text ON pos !OF
17            | text ON pos OF IMAGE
18            | ON pos "," IS text
19            | ON pos OF IMAGE "," IS text
20            | ON pos OF IMAGE "," text
21            | ON pos "," text

```

!TOKEN indica que en ese patron no puede haber un token tipo TOKEN en esa posicion. Notese que, en el caso de "IS 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.

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 = dog, neighbors[next] = a ball.

Este parser es muy similar al anterior, con la diferencia de que en este solo cambian algunos tokens utilizados y las reglas gramaticales. Los tokens de la etapa léxica son los siguientes:

```

1 tokens = [ON, OF, AND, IS, POS, POSITION, WORD, NUM, TO, NEXT]
2
3 keywords = {
4     ON = [in, on, at, find]
5     OF = [of]
6     AND = [and]
7     IS = [is, are, theres, find]
8     POSITION = [position, pos, side, location]
9     NEXT = [next]
10    TO = [to]
11    POS = [ left, right, bottom, top, down, up, lower, center,
12           middle, corner, near, bellow, front,      beside ]
13    }

```

Reglas gramaticales para las relaciones en esta etapa.

```

1      relation ::=  IS text ON pos OF text
2                  | IS text TO pos OF text
3                  | IS text ON pos TO text
4                  | IS text TO pos TO text
5                  | text ON pos TO text
6                  | text ON pos OF text
7                  | IS text , pos TO text
8                  | IS text , pos OF text
9                  | IS text pos TO text
10                 | IS text pos OF text
11                 | text pos TO text
12                 | text pos OF text
13                 | IS text NEXT TO text
14                 | IS text NEXT OF text
15                 | text NEXT OF text
16                 | text NEXT TO text
17                 | ON text pos , IS text
18                 | ON text pos , text
19                 | ON text pos IS text
20                 | ON pos OF text , IS text
21                 | ON pos OF text , text
22                 | TO pos OF text , IS text
23                 | TO pos OF text , text
24                 | ON pos TO text , IS text
25                 | ON pos TO text , text
26                 | TO pos TO text , IS text
27                 | TO pos TO text , text
28                 | ON pos OF text IS text
29                 | TO pos OF text IS text
30                 | ON pos TO text IS text

```

```

31 | TO pos TO text IS text
32 | pos TO text , IS text
33 | pos TO text , text
34 | pos OF text , IS text
35 | pos OF text , text
36 | pos TO text IS text
37 | pos OF text IS text
38 | NEXT TO text , text
39 | NEXT OF text , text
40 | NEXT TO text IS text
41 | NEXT OF text IS text

```

Variantes propuestas 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 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, BERT y T5 1, 2.

Otra alternativa es la creación de un modelo propio. 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. Al describir al modelo el tipo de estructura gramatical que se sabe que es correcta, estos son procesados correctamente 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 falla, permitiendo así aprender de ellos también. Incluso se pueden crear diferentes parsers que no sean ambiguos entre sí para generar casos de prueba 4, 5.

Una propuesta para este modelo ya ha sido creada pero aún no ha sido entrenada. Esta está implementada en tensorflow.keras, y es un modelo con las siguientes capas:

1. Dos capas de entrada, una para los tokens y otra para la atención que se le debe prestar a estos.
2. Dos capas de Embedding que convierten los tokens a un espacio de embedding.
3. Dos capas cnnlayer que son utilizadas para aprender patrones.
4. Una capa de Attention, para recibir las entradas y generar pesos.
5. Dos capas GlobalAveragePooling1D para redimensionar el espacio a una dimensión.

6. Una capa Concatenate, para concatenar ambas capas superiores.
7. Una capa Dense de 256 dimensiones.
8. Una capa Dense de 512 dimensiones.
9. Una capa Dense de 1024 dimensiones.
10. Una capa Dense con cantidad de dimensiones igual a la longitud de la salida.
11. 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 para el uso de capas de atención

INSERTAR GRAFICA DEL MODELO

Fase de Cálculo de Similitud y Recuperación de Información

Después de las dos fases 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:

1. Se calcula la similitud del coseno entre el embedding del *text* y de la *image*:

$$1 - \cos(v_1, v_2) / ||v_1||v_2||$$
2. Se calcula la similitud por posicion, para ello se multiplica el valor de la similitud del coseno por la similitud posicional, la similitud posicional esta dada por:

$$1 - \sqrt{((v_1.pos.x - v_2.pos.x)^2 + (v_1.pos.y - v_2.pos.y)^2)}.$$
3. 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 posicion p relativa al *image* la similitud de este con cada vecino en la posición p relativa a text. Por cada vecino en posicion p en texto, se guardara el mejor, el mas relevante, de los vecinos en la posicion 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 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 disminuirá.

4. La similitud final esta dada por la similitud entre un *image feature* 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 es mayor que cierto umbral, entonces se guarda, luego de recorrer todas las imagenes, la similitud para el *text* sera la mejor de todas las similitudes con las *images*. La similitud final, es entoces la suma de cada una de las similitudes por cada *text* en *text feature*.

Parametros ajustables en las fases

-
- umbral de importancia de la distancia euclidean = float: Elevar a la potencia la distancia euclidean.
- umbral de division de la distacia euclidean = float: se divide el valor de la distancia euclidean por este valor.
- similitud minima relevante para regiones vecinas = float: Si la similitud es menor que esto se considera insignificante y se deja de usar
- similitud minima relevante = float: Esta es la similitud a partir de a cual puede considerarse relevante un concepto
- similitud minima relevante original = float: Esta es la similitud a partir de a cual puede considerarse relevante algo en la imagen original y texto original
- similitud minima relevante usando descripciones = float: Esta es la similitud a partir de a cual puede considerarse relevante algo 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 cercania entre un objeto y otro
- importancia de las regiones negativas = float: Defne que tan importante es una region negativa, mientras mas alto, mas baja la similitud de una region poco relevante

Otras Propuestas

Una propuesta inicialmente considerada para mejorar el reconocimiento de colores en la imagen fue finalmente abandonada. Se decidió centrar más interés en la propuesta de reconocimiento posicional, ya que CLIP maneja eficientemente las relaciones de colores en las imágenes, a pesar de que 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 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 nuevos modelos, el diseño de esta propuesta resultaría en mejores resultados para ese modelo específico.

En las imágenes, se puede observar que el control de colores de CLIP 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 puede manejar actualmente.

Observaciones

Como desarrollo principal se pretendía lograr recuperar imágenes desde una consulta en lenguaje natural, pero, además se observa que la metodología es viable para recuperar imágenes usando imágenes como consulta, teniendo en cuenta las relaciones posicionales de cada objeto en cuestión junto con sus conceptos agrupados en embeddings. El resultado de recuperar imagen imagen debe ser ajustado para lograr una recuperación precisa, una vez se ajusten parámetros para la recuperación imagen-imagen, este será posible de realizar.

Capítulo 4

Detalles de Implementación y Experimentos

Conclusiones

Conclusiones

Recomendaciones

Recomendaciones