UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

RECUPERACIÓN SEMÁNTICA DE IMÁGENES BASADA EN CONTENIDO EN EL COMERCIO ELECTRÓNICO A TRAVÉS DE MODELOS AUTO-SUPERVISADOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

GUILLERMO ANDRÉS MARTÍNEZ SALAZAR

PROFESOR GUÍA:
JOSÉ SAAVEDRA RONDO

PROFESOR CO-GUÍA:
NILS MURRUGARRA LLERENA

MIEMBROS DE LA COMISIÓN:
ANDRÉS ABELIUK KIMELMAN
DANIEL PEROVICH GEROSA

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN
POR: GUILLERMO ANDRÉS MARTÍNEZ SALAZAR
FECHA: 2022
PROF. GUÍA: JOSÉ SAAVEDRA RONDO

RECUPERACIÓN SEMÁNTICA DE IMÁGENES BASADA EN CONTENIDO EN EL COMERCIO ELECTRÓNICO A TRAVÉS DE MODELOS AUTO-SUPERVISADOS

El comercio electrónico forma parte sustancial en una gran cantidad de negocios enfocados en la venta de productos o servicios en línea, más aún cuando se toma en cuenta la situación actual ocasionada por la pandemia. Existen diversas herramientas que facilitan y mejoran la experiencia al usar este tipo de comercios, entre ellas, los motores de búsqueda que usan la recuperación de imágenes basada en contenido. Esta es un problema del área de visión por computadora en el cual se usan imágenes para realizar consultas y recuperar otras imágenes que sean similares a la consulta. Sin embargo, en la realidad, los motores de búsqueda que funcionan con imágenes poseen una baja efectividad, sobre todo si la consulta ingresada se presenta en un medio no controlado.

Los catálogos de comercio electrónico contienen gran variedad de información, como por ejemplo, los textos de las descripciones de sus productos, que pueden ser utilizados para agregar semántica, y por ende, mejorar la recuperación de imágenes. En este trabajo se crean seis *datasets* de distintos comercios electrónicos, y se realizan estudios preliminares sobre estos para evaluar el posible aporte de las descripciones de productos. Además, se diseñan, implementan y evalúan, sobre los mismos *datasets*, diferentes métodos para agregar semántica en la recuperación de imágenes: ajuste de *embeddings* dentro del catálogo, *query* adaptativa en tiempo real, redes neuronales *feed-forward fully-connected* para aprender a representar espacios visuales como espacios de texto, uso de UMAP para reducir dimensiones manteniendo topología local, y el uso y entrenamiento del recientemente presentado modelo CLIP.

Los resultados que se obtienen al utilizar tanto UMAP como las redes neuronales *feed-forward fully-connected* no logran superar el modelo ResNet-50 definido como *baseline*. Sin embargo, el resto de los métodos mejora la recuperación significativamente en la mayoría de los casos, tanto para imágenes limpias como para fotos e imágenes con ruido, lográndose apreciar en los resultados un aumento en la semántica. Para las imágenes del primer tipo, tomando en cuenta todos los catálogos, se obtienen mejoras de mAP@20 de hasta un 14.83 % al evaluar a grano grueso. Evaluando a grano fino, el mAP@20 logra aumentar en un 11.45 %. Por otro lado, en cuanto a las imágenes del segundo tipo, evaluadas únicamente en dos catálogos y a grano grueso, se obtienen mejoras de mAP@20 de un 13.2 % para un catálogo, y un 24.11 % para el otro. Finalmente, se proponen enfoques para mejorar los resultados de este trabajo, tanto diseñando otros algoritmos de ajuste de *embeddings* y *queries* adaptativas, como realizando entrenamiento de modelos con una mayor cantidad de datos, y utilizando imágenes con ruido. Además, como idea para expandir las aplicaciones de estos métodos, se propone estudiar la capacidad del modelo CLIP dentro del contexto de recuperación de imágenes basada en dibujos.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

1. Introducción

1.1. Contexto y formulación del problema

El comercio electrónico consiste en la compra y venta de bienes o servicios mediante internet. Es una parte sustancial de una gran cantidad de negocios que se enfocan en la venta de productos físicos o servicios en línea y está presente en nuestras vidas constantemente, ya sea por comodidad o necesidad. A esto se le suma la pandemia que afecta a la mayor parte de la población mundial en los tiempos actuales, lo que provoca que los negocios tengan la necesidad de adherirse o mejorar la logística detrás de su comercio electrónico, por lo que se aumenta la indispensabilidad de este.

La búsqueda de un producto particular en un catálogo de comercio electrónico puede resultar frustrante cuando se quieren características específicas de éste, las cuales no pueden representarse en una simple búsqueda por texto o filtro de catálogo. Dada esta problemática, un considerable número de comercios electrónicos ofrece herramientas que le facilitan la búsqueda al usuario. Dentro de estas herramientas se encuentra la *recuperación de imágenes basada en contenido* (CBIR, por sus siglas en inglés), también conocida como *consulta por contenido de imagen*, que es una aplicación dentro del área de *visión por computadora* que permite al usuario realizar una consulta únicamente mediante la utilización de una imagen para así encontrar de manera eficaz algo similar a lo que está buscando. Por ejemplo, si un usuario quiere buscar un balón dentro de un catálogo, puede ingresar la imagen de alguno que le parezca relevante y recibir una serie de alternativas, como se puede apreciar en la figura 1.1.

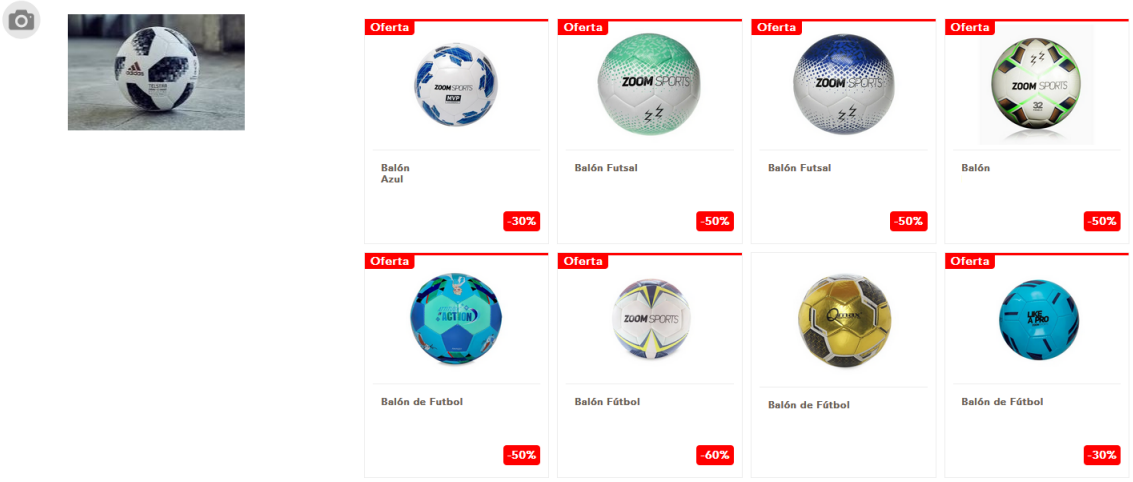


Figura 1.1: Resultados obtenidos en el buscador por contenido para un balón de fútbol en el sitio web de Pepeganga S.A.

Es así como el comercio electrónico constituye una de las aplicaciones con más demanda de motores de búsqueda que utilizan *recuperación de imágenes basada en contenido*, los cuales logran expresar la intención de los usuarios. Actualmente, estos motores de búsqueda por imágenes son generados mediante el entrenamiento de un modelo para un problema más general, como por ejemplo *ImageNet*, un *dataset* diseñado para ser usado en la investigación de *software* de reconocimiento de objetos. En adición, algunos modelos se ajustan con datos etiquetados para un problema específico. Sin embargo, estos modelos tienen baja efectividad, sobre todo cuando la consulta realizada a través de una imagen se encuentra en un medio no controlado o con elementos extras en esta. A este problema, se suman los resultados inconsistentes, retornando productos correctos pero dispersos entre una variedad de otros productos que están lejos de ser lo que el usuario busca. Por último, existe una cantidad prácticamente ilimitada de información que no está siendo explotada en estos catálogos de comercio electrónico, como por ejemplo, las descripciones de los productos. Esta información podría ser utilizada para agregar semántica y mejorar los resultados de las consultas.

En base a lo anterior, surge la motivación de reajustar un modelo aprovechando la información visual y textual de los productos de un catálogo de comercio electrónico, para luego evaluar el impacto de esta información, ya sea en conjunto o separada, para aprender bases léxicas. De esta manera se busca aumentar la semántica del modelo para generar respuestas más efectivas a consultas realizadas en base a contenido.

En esta memoria se proponen, implementan y evalúan distintos métodos auto-supervisados sobre 6 catálogos de comercio electrónico, los cuales buscan aumentar la semántica de una consulta con imagen a través de texto mediante el ajuste de las representaciones visuales originales de un catálogo directamente sin la necesidad de etiquetar de manera excesiva los datos, permitiendo así reducir el costo de crear *datasets* de evaluación de manera significativa. Los métodos que se implementan son diversos y con variados enfoques, ya sea mediante la reducción de dimensiones manteniendo topología local,

algoritmos que ajustan *embeddings* visuales de los productos de un catálogo, un algoritmo que modifica el *embedding* obtenido por una imagen consultada en tiempo real, redes neuronales *feed-forward fully connected* que buscan aprender un espacio visual y transformarlo a un espacio de texto, y la utilización del reciente modelo multi-modal CLIP, que promete resultados sobresalientes y que no ha sido probado en el contexto de recuperación de imágenes. Estos métodos son comparados con una ResNet-50, una red neuronal ampliamente utilizada dentro del contexto de estudio de este trabajo, la cual fue entrenada con *ImageNet*.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo general de este trabajo es reajustar un modelo de recuperación de imágenes para aumentar la semántica de la recuperación utilizando información visual y textual de productos en catálogos de comercio electrónico a través de modelos auto-supervisados.

1.2.2. Objetivos específicos

1. Crear *datasets* para utilizar de manera auto-supervisada en diversos contextos.
2. Utilizar técnicas de reducción de dimensión con métodos que preserven localidad en conjunto a datos de productos de un comercio electrónico para entrenar un modelo reducido basado en redes neuronales.
3. Evaluar técnicas de representación de texto respecto a la semántica de las representaciones.
4. Evaluar el impacto de las imágenes de los productos de un catálogo en la semántica de las representaciones visuales.
5. Ajustar representaciones visuales con respecto a las representaciones de texto creadas.
6. Evaluar el impacto de mezclar textos e imágenes para promover la semántica del resultado.

2. Marco teórico

2.1. Similitud de *embeddings*

La *recuperación de imágenes basada en contenido* es una aplicación dentro del área de *visión por computadora* que permite a un usuario realizar una consulta únicamente mediante la utilización de una imagen para así encontrar de manera eficaz algo similar a lo que está buscando. Que la recuperación esté basada en contenido significa que en la búsqueda se analizan aspectos propios de la imagen usada como consulta, como por ejemplo formas, texturas, colores, entre otros. En el proceso de recuperación se comparan las características de la imagen ingresada con otras imágenes de un *dataset* calculando la "distancia" o "similitud" entre estas características, para finalmente retornar las imágenes con menor "distancia" o mayor "similitud". Esto puede realizarse tanto para *embeddings* pertenecientes a un espacio visual como los de un espacio textual.

La distancia entre los vectores de características extraídos de dos imágenes o dos oraciones puede calcularse de distintas formas. Una opción válida es la distancia euclidiana, la cual considerando dos vectores X e Y de largo n, está definida como:

$$d(X,Y) = \sqrt{\sum_{k=1}^{n} |X_k - Y_k|^2} \tag{2.1}$$

Sin embargo, esta es susceptible a variaciones de puntos particulares, en donde un solo punto de un vector de características puede aumentar la distancia significativamente. Luego, es común definir la distancia como una similitud entre dos vectores, para lo que se utiliza la *similitud coseno*, que se define como:

$$S_C(X,Y) = \frac{\sum_{k=1}^{n} X_k Y_k}{\sqrt{\sum_{k=1}^{n} X_k^2}\sqrt{\sum_{k=1}^{n} Y_k^2}} \tag{2.2}$$

No obstante, independiente de la función de distancia utilizada, puede ocurrir el llamado *bursting effect*, que consiste en la presencia de altas diferencias en pocas dimensiones entre vectores de características. Para solucionar este problema suele usarse la *normalización de raíz cuadrada*, que calcula la raíz cuadrada del vector manteniendo los signos y luego lo transforma en un vector unitario.

2.2. Mean Average Precision (mAP)

Una métrica utilizada ampliamente en los problemas relacionados a la recuperación de información y problemas de detección es el *Mean Average Precision* o mAP, que evalúa la capacidad de un modelo de obtener resultados relevantes. Para llevar a cabo el cálculo de esta métrica, se necesita tener un conjunto de imágenes I sobre las que se realiza la búsqueda usando un conjunto de consultas o *queries* Q, para luego obtener el conjunto de imágenes ordenadas por similitud R. Así, para obtener un resultado de mAP, se realizan los siguientes pasos:

1. Definiendo el conjunto de relevantes para cada *query* q como

$$\Gamma_q = \{x \in I, q \in Q \mid \text{x es relevante para Q } \} \tag{2.3}$$

   Se define la función característica f

$$f_{\Gamma_q}(x) = \begin{cases} 1 & x \in \Gamma_q \\ 0 & \text{de lo contrario.} \end{cases} \tag{2.4}$$

   la cual decide si x es relevante para q o no.

2. Luego, se define la precisión para un elemento recuperado en la posición i del conjunto R como

$$pr_q(i; R) = \frac{\sum_{k=1}^{i} f_{\Gamma_q}(r_k)}{i} \cdot f_{\Gamma_q(r_i)} \tag{2.5}$$

   en donde este valor indica la pureza del resultado y varía entre 0 y 1 indicando mínima y máxima pureza respectivamente.

3. Después, para cada *query* q, se calcula la precisión promedio (*average precision*) de la siguiente manera:

$$AP_q(R) = \frac{\sum_{i=1}^{|R|} pr_q(i, R)}{\sum_{i=1}^{|R|} f_{\Gamma_{r_i}}} \tag{2.6}$$

4. Finalmente, el valor resultante del *Mean Average Precision* o mAP queda definido por

$$mAP(Q, R) = \frac{\sum_{q \in Q} AP_q(R)}{|Q|} \tag{2.7}$$

El cálculo del mAP es computacionalmente caro, por lo que existen variaciones para realizarlo sobre los K primeros resultados de un ranking, o bien, sobre los K primeros resultados relevantes de este, lo cual se define como mAP@K.

2.3. Redes neuronales

Una red neuronal es un modelo matemático cuya inspiración proviene del funcionamiento de las neuronas, las cuales reciben variados impulsos a través de sus dendritas, mientras que el axón transmite una señal. Para las redes neuronales artificiales, se define de igual manera una unidad básica llamada neurona o perceptrón, y el modelo matemático denominado como red neuronal se compone por múltiples perceptrones interconectados entre sí.

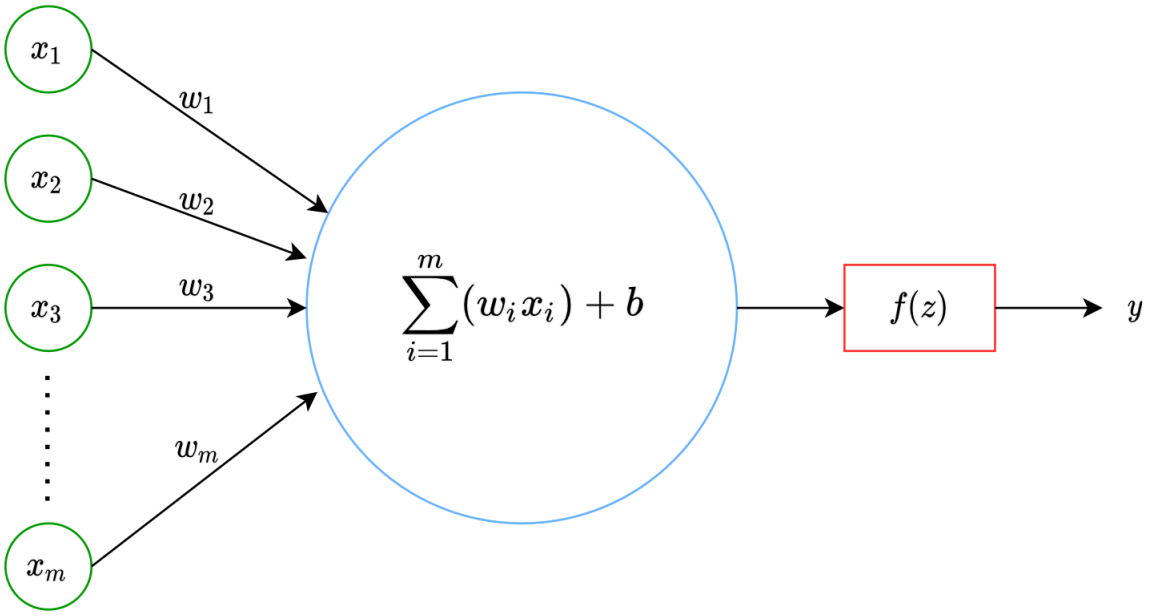


Figura 2.1: Estructura general de un perceptrón.

La estructura general de un perceptrón se puede observar en la figura 2.1, en donde los valores $x_1, x_2, ..., x_m$ son las componentes de un vector de entrada, y los valores $w_1, w_2, ..., w_m$ son las componentes del vector de pesos asociados al perceptrón. Con estos dos vectores, se realiza un producto punto, a cuyo resultado se le suma un valor arbitrario b, llamado *bias*. De esta manera, se obtiene un valor z, al cual se le aplica una función de activación f, que retorna un valor entre -1 y 1 dependiendo de qué función específica se utilice, para obtener el valor final y.

Al estar formadas por múltiples perceptrones, las redes neuronales poseen una gran cantidad de pesos distintos, los cuales, durante el entrenamiento, se ajustan utilizando los gradientes de estos mediante el proceso llamado *back-propagation*.

2.3.1. Capas *fully-connected*

Una red neuronal posee capas *fully-connected*, o densamente conectadas, cuando todos los valores obtenidos por los perceptrones de una capa son utilizados como entradas en todos los perceptrones de la capa siguiente. La estructura general de una red neuronal con capas *fully-connected* se puede observar en la figura 2.2.

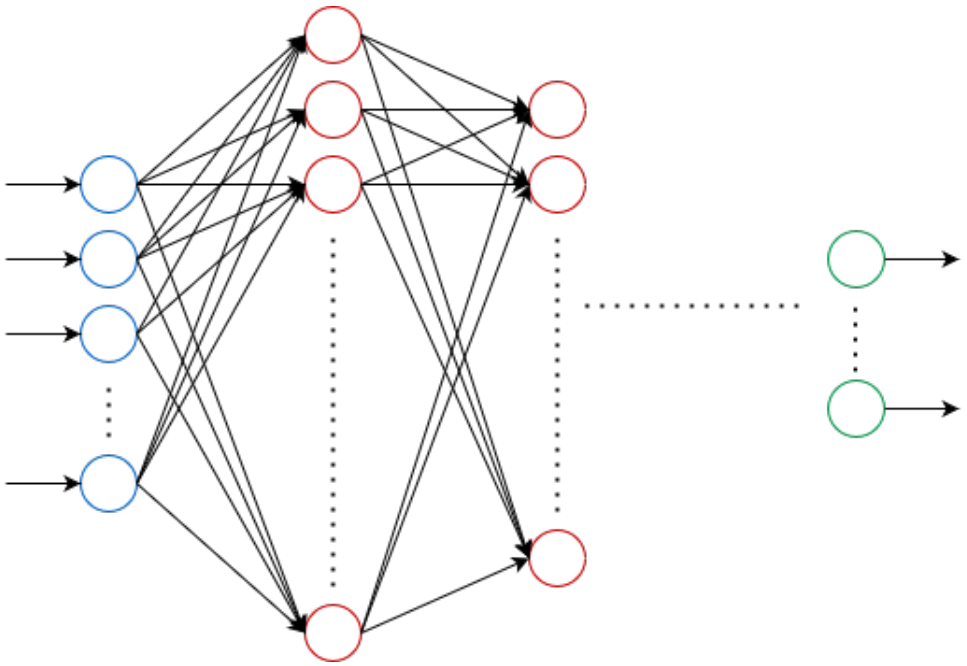


Figura 2.2: Estructura general de red neuronal *feed-forward fully-connected*.

2.3.2. Capas convolucionales

Las capas convolucionales se presentan en redes cuya arquitectura se adapta principalmente para ser utilizadas en el análisis y procesamiento de imágenes. La entrada de una capa convolucional ya no es un simple vector unidimensional, sino más bien tridimensional, definido por una altura, un ancho, y un número de canales.

La convolución consiste en una especie de producto punto deslizante, donde una matriz llamada filtro, generalmente de tamaño más pequeño que la entrada, se mueve dentro de esta, y se calcula el producto punto como si las dos matrices fueran vectores. En la figura 2.3

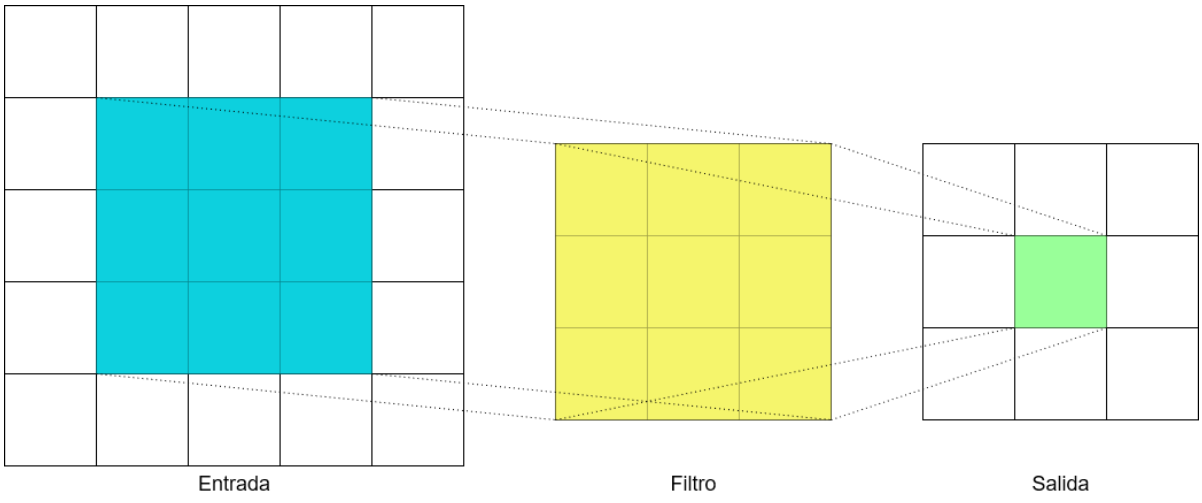


Figura 2.3: Ilustración de una convolución entre una entrada y un filtro.

2.3.3. Funciones de activación

Después de cada capa existe una función de activación, las cuales permiten definir el umbral de activación de la capa. Existen múltiples funciones de activación, en particular, la utilizada en este trabajo es la función ReLU.

La función de activación ReLU retorna 0 cuando el valor ingresado es menor o igual a 0, y retorna el mismo valor ingresado en caso contrario, lo cual se puede observar en la figura 2.4. Es una de las funciones de activación más utilizadas hoy en día, y su expresión se define en la ecuación 2.8.

$$ReLU(x) = max(0, x) = \begin{cases} 0 & x \leq 0 \\ x & \text{de lo contrario.} \end{cases} \tag{2.8}$$

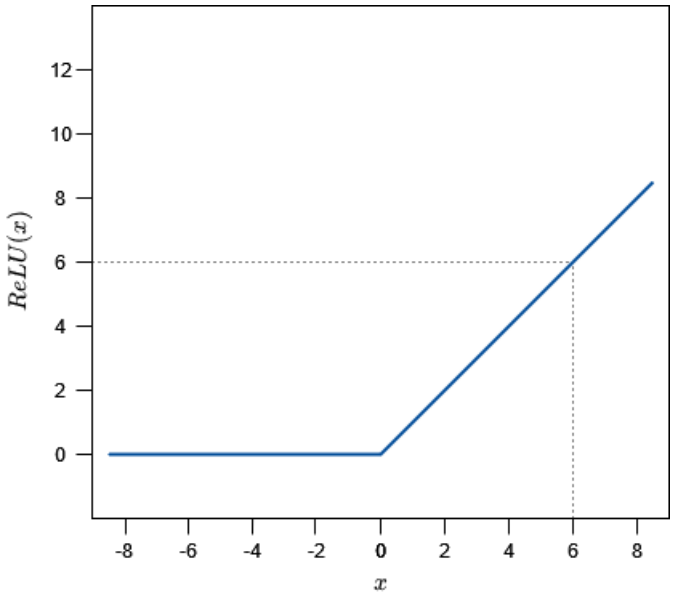


Figura 2.4: Gráfico de la función ReLU.

2.3.4. Funciones de pérdida

Las funciones de pérdida definen una función que se busca optimizar, mediante el ajuste de los parámetros de una red, para que su valor final resulte igual a 0, lo que es igual a decir que no existe pérdida. Esta optimización se realiza de manera reiterativa durante el entrenamiento. En particular, las funciones de pérdida utilizadas en este trabajo corresponden a las siguientes:

- ***Mean squared error loss***: Es una función utilizada ampliamente en las regresiones que mide el promedio de los errores al cuadrado. En general, es una opción viable cuando los datos de entrada poseen una distribución normal, y cuando se quieren penalizar fuertemente los errores grandes. Considerando a $\bar{X}$ como el valor objetivo de la función, y a $\bar{Y}$ como la salida real de la red, la función se define como:

$$\textit{MSE}_{Loss} = ||\bar{Y} - \bar{X}||_2^2 \tag{2.9}$$

- ***Cross-Entropy loss***: Esta función se utiliza normalmente para problemas de clasificación con multi-clases, y lo que hace realmente es medir la diferencia entre dos distribuciones de probabilidades. Al usar esta función de pérdida, se recomienda utilizar como función de activación únicamente a la función *softmax*. Definiendo como x el valor objetivo de la red, a $\hat{y}$ como la salida real de la misma, y a N como la cantidad de clases posible, es decir, el largo de salida del vector, la función *Cross-Entropy loss* queda definida por:

$$\textit{CrossEntropy}_{Loss}(x, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N} x_i \log \hat{y}_i \tag{2.10}$$

- ***Symmetric loss* (CLIP)**: La función de pérdida *symmetric loss* utilizada en CLIP [1] se utiliza para que la red aprenda un espacio vectorial conjunto tanto para imágenes como texto, es decir, que la salida para una imagen sea parecida, por ejemplo, a la salida para la descripción textual de la misma. Esta función se define como:

$$\textit{Symmetric}_{Loss} = \frac{\textit{CrossEntropy}_{Loss}(X, z) + \textit{CrossEntropy}_{Loss}(Y, z)}{2} \tag{2.11}$$

  en donde X es el valor real para el espacio visual, Y el valor real para el espacio de texto, y z un vector $z = [0, 1, ..., N-1]$, con N igual al tamaño del *batch* utilizado durante el entrenamiento.

2.3.5. Algoritmos de optimización

Los algoritmos de optimización se utilizan para definir como van cambiando los parámetros de una red neuronal cuando se está realizando el entrenamiento con el objetivo de minimizar la pérdida. Existen múltiples algoritmos de optimización con distintos ajustes. Los utilizados en este trabajo corresponden a los siguientes:

- ***Stochastic Gradient Descent* (SGD)**: Está basado en una función convexa y ajusta los parámetros de manera iterativa para minimizar dicha función en un mínimo local. Si bien es un optimizador bastante utilizado y en algunos problemas ofrece buenos resultados, es computancionalmente caro. La actualización de los parámetros de la red mediante SGD se realiza de la siguiente manera:

$$\theta = \theta - \alpha \nabla_\theta J(\theta; x^{(i)}, y^{(i)}) \tag{2.12}$$

  donde α corresponde al *learning rate* que se elige arbitrariamente, $J(\theta)$ es la función de pérdida, y $(x^{(i)}, y^{(i)})$ es un par del conjunto de entrenamiento.

  Existen algunos métodos que modifican el algoritmo de SGD buscando obtener mejores resultados. Uno de ellos es la incorporación de *momentum*, el cual es utilizado para reducir la alta varianza que posee SGD, suavizando la convergencia. Esto añade un parámetro más a la ecuación para actualizar un parámetro, obteniéndose:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \alpha \nabla_\theta J(\theta; x^{(i)}, y^{(i)}) \\ \theta &= \theta - v_t \end{aligned} \tag{2.13}$$

con γ el nuevo parámetro, el cual usualmente es igual a 0.9.

Otro método que modifica la optimización por SGD es la aplicación del *Nesterov momentum*, que modifica la fórmula anterior resultando en lo siguiente:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \alpha \nabla_\theta J(\theta - \gamma v_{t-1}; x^{(i)}, y^{(i)}) \\ \theta &= \theta - v_t \end{aligned} \tag{2.14}$$

- ***Adaptive Moment Estimation* (Adam)**: El algoritmo de optimización Adam calcula *learning rates* adaptativos para cada parámetro de la red. Trabaja con los *momentum* de primer y segundo orden. Definiendo a m_t y v_t como el primer y segundo orden de *momentum* respectivamente, se tiene que:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \end{aligned} \tag{2.15}$$

Luego, los parámetros se actualizan mediante la siguiente fórmula:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{2.16}$$

donde los valores por defecto usualmente utilizados son $\beta_1 = 0.9$, $\beta_2 = 0.999$ y $\epsilon = 10^{-8}$.

2.3.6. *Batch Normalization*

El algoritmo de *batch normalization* [2] fue presentado el año 2015 y busca hacer el entrenamiento de una red neuronal más rápido y de manera más estable. Esto lo logra mediante la normalización de los vectores en las capas escondidas de una red para el *batch* que se está utilizando. Usualmente, esto se denomina como una capa de *batch normalization*, y se coloca justo antes de aplicar la función de activación.

2.3.7. *Dropout*

El *dropout* [3] es una técnica de regularización introducida el año 2014, el cual hace que durante el entrenamiento de una red neuronal se omitan aleatoriamente neuronas con una probabilidad definida arbitrariamente en cada *batch*, lo cual se puede ver representado en la figura 2.5. De este modo, se busca reducir el sobre-ajuste de las redes neuronales.

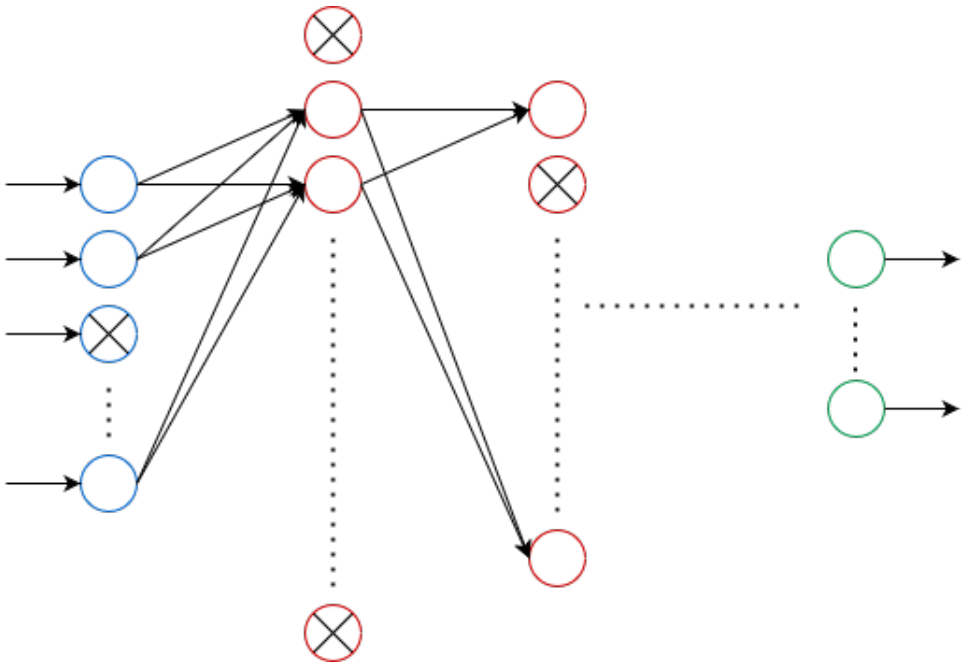


Figura 2.5: Estructura general de una red neuronal luego de aplicar la técnica de regularización *dropout*.

2.4. ResNet

Las redes neuronales profundas han permitido mejorar notablemente los resultados obtenidos en tareas de reconocimiento visual. No obstante, este tipo de redes se ven afectadas por el problema del desvanecimiento o explosión del gradiente. Este problema ha sido abordado mediante la inicialización normalizada y las capas de normalización.

La normalización permite a las redes neuronales profundas converger, sin embargo, durante este proceso, aparece el problema de degradación: el aumento de profundidad en una red genera una saturación en la exactitud (*accuracy*).

Para solucionar el problema de la degradación, *ResNet* [4] introduce el aprendizaje profundo residual. Esto se logra mediante la utilización de la salida de la capa de una red como entrada de la siguiente capa, y además, se hace una conexión directa de esta salida en una capa posterior. Todo este proceso se denomina bloque residual, y se puede observar en la figura 2.6.

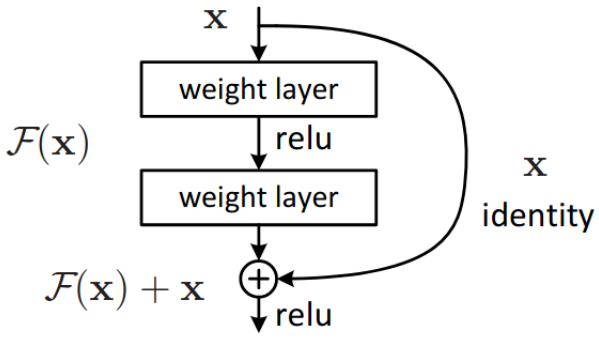


Figura 2.6: Estructura de bloque residual presentado en *Deep Residual Learning for Image Recognition* [4].

Existen diversas arquitecturas tipo ResNet que ocupan distintos números de capas. En este trabajo en particular se utiliza la arquitectura de ResNet-50, la cual posee 50 capas.

2.5. Semántica en la búsqueda por contenido

El contenido de una imagen se percibe como algo subjetivo por las personas, creando una semántica que los vectores de características extraídos por un modelo actual fallan al capturar, ocasionando resultados erróneos e inconsistentes. Un ejemplo de esto puede observarse en la figura 2.7, donde el sistema recibe como entrada la imagen de una aspiradora y luego se retornan productos en orden descendiente de similitud. Se puede apreciar como el sistema retorna aspiradoras en un inicio. Sin embargo, en un punto ofrece un producto completamente distinto y luego, nuevamente, muestra una aspiradora. Este patrón se observa constantemente en una enorme variedad de productos ingresados a sistemas actuales que se encargan de recuperar imágenes por contenido.

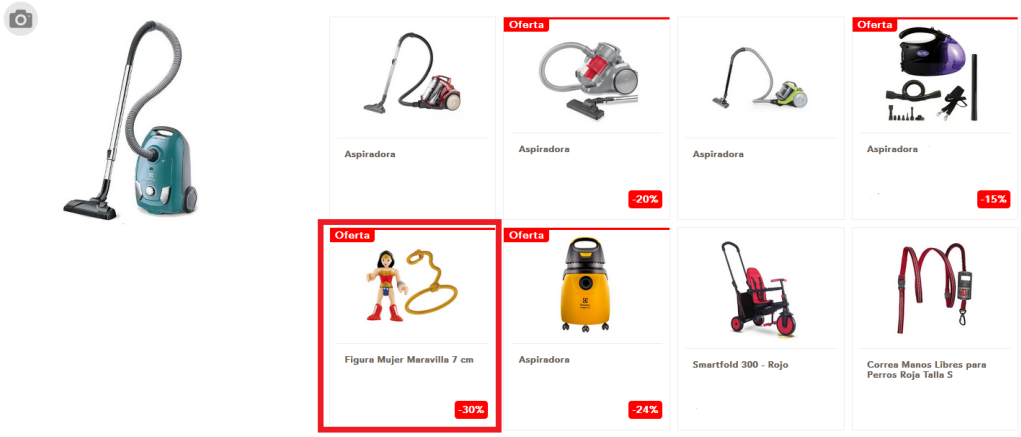


Figura 2.7: Resultados obtenidos en el buscador por contenido de Pepeganga S.A.

Así surge la necesidad de disminuir la diferencia de semántica existente entre lo que un usuario busca y lo que un modelo le retorna mediante la utilización de datos disponibles que no han sido aprovechados, lo que permitiría mejorar las respuestas a las consultas.

Por ejemplo, en el caso mostrado previamente, se podría evitar retornar un juguete entre aspiradoras si el modelo utilizara información textual de la descripción de los productos cuya similitud era mayor.

2.6. *Uniform Manifold Approximation and Projection*

Una manera de aumentar la semántica de los modelos es por medio de la utilización de UMAP (*Uniform Manifold Approximation and Projection*) [5], un método que permite reducir la dimensión de *embeddings* preservando localidad y que ha demostrado buenos resultados, superando el estado del arte en tareas de *SBIR* (Sketch-Based Image Retrieval) [6].

Esta técnica de reducción de dimensiones puede ser usada para visualizar de la misma manera que t-SNE [7], pero también para la reducción de dimensión no lineal en general. El algoritmo, entendiendo a la *variedad* como un espacio matemático abstracto (*manifold*), se basa en tres suposiciones sobre los datos:

- Los datos esta distribuidos uniformemente en la *variedad de Riemann*.
- La métrica de Riemann es localmente constante (o puede aproximarse como tal).
- La *variedad* está conectada localmente.

A partir de estos supuestos, es posible modelar la *variedad* con una estructura topológica difusa.

2.7. Vectores de palabras

Una alternativa a UMAP que ha demostrado ser efectiva en tareas de *visión por computadora* es la construcción de bases léxicas, en donde esto último corresponde a la tarea de poder determinar cuáles imágenes de un *corpus* representan a una palabra o un conjunto de palabras en particular [8]. Para poder llevar a cabo esto, se requieren vectores de palabras.

2.7.1. *Word2Vec*

El método para poder representar texto mediante vectores del *software Word2Vec* [9, 10] surge debido a las limitaciones de los modelos más simples que existían hasta ese momento, en donde simplemente incorporar más datos para el entrenamiento no era viable debido a la mala escalabilidad de estos modelos. Así, se presentan nuevas técnicas que permiten aprender vectores de palabras de alta calidad utilizando enormes *datasets* con billones de palabras.

Word2Vec presenta dos nuevos modelos:

- *Continuous Bag-of-Words* (CBOW)
- *Continuous Skip-gram* (SG)

La arquitectura del primer modelo, observada en la Figura 2.8, es similar a la del *feed-forward Neural Net Language Model* (NNLM) [11], una red neuronal que consiste de capas de entrada, proyección, oculta y de salida. Sin embargo, en CBOW, la capa no-lineal escondida es removida y la capa de proyección es compartida para todas las palabras. A esta arquitectura se le denomina modelo de bolsa de palabras (*bag-of-words*) debido a que el orden de las palabras anteriores no influye en la proyección. Además, en adición a utilizar palabras anteriores, se utilizan palabras del futuro, y el modelo es entrenado mediante la tarea de clasificar (predecir) una palabra que se encuentra entre las palabras anteriores y del futuro (contexto). Finalmente, el modelo es llamado CBOW dado que, a diferencia de un modelo de bolsa de palabras estándar, usa una representación del contexto distribuida de manera continua.

Por otro lado, la arquitectura del modelo *Continuous Skip-gram*, también observada en la figura 2.8, es parecida a la de CBOW. La diferencia recae en que en vez de predecir una palabra en base al contexto, intenta maximizar la clasificación de una palabra en base a otra de la misma oración. De manera más precisa, se buscan predecir las palabras anteriores y futuras a una palabra en específico para cierto rango.

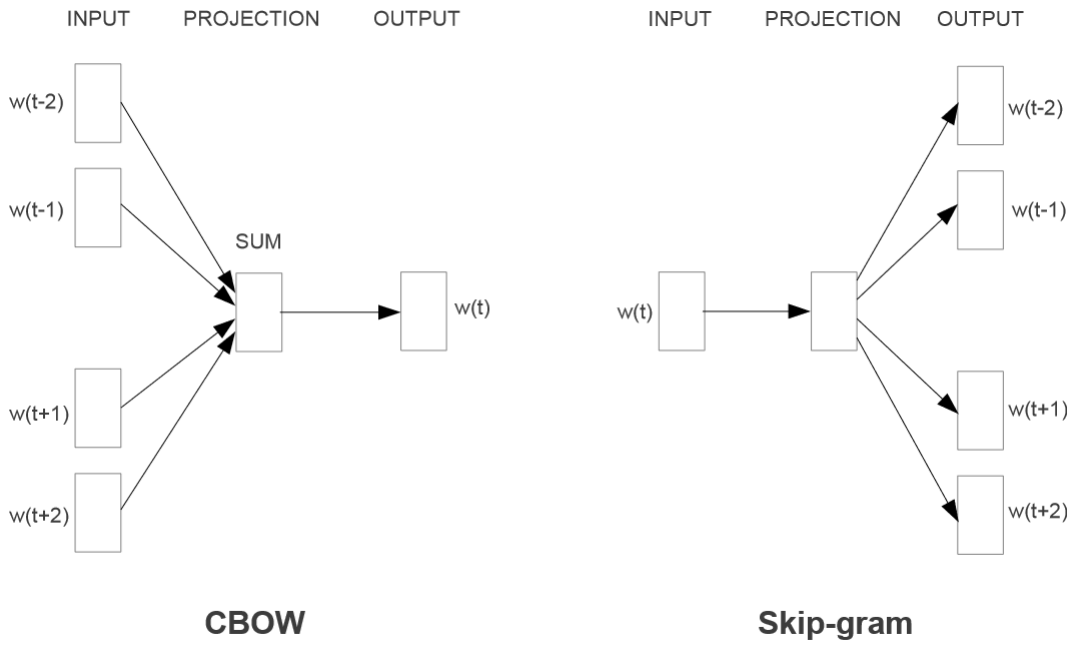


Figura 2.8: Arquitecturas CBOW y SG presentadas en *Efficient Estimation of Word Representations in Vector Space* [9].

2.7.2. *GloVe*

GloVe [12] aparece como una solución a los inconvenientes que sufren las dos "familias" de modelos para aprender vectores de palabras existentes hasta entonces: métodos de factorización de matriz global, como el análisis semántico latente (LSA) [13], y los métodos que utilizan una ventana local, como el modelo *Skip-gram*. Si bien los métodos como LSA aprovechan de manera eficiente la información estadística, lo hacen relativa-

mente mal en la tarea de analogía de palabras, lo cual indica una estructura de espacio vectorial subóptima. Por otro lado, los métodos como *Skip-gram* pueden funcionar mejor en la tarea de analogía, pero utilizan pobremente las estadísticas del corpus ya que se entrenan en ventanas de contexto local separadas en lugar de recuentos globales de co-ocurrencia.

El modelo *GloVe* obtiene su nombre de *Global Vectors*, porque las estadísticas del corpus global son capturadas directamente por este. El método presentado por *GloVe* define que el punto de partida apropiado para el aprendizaje de vectores de palabras debería ser con razones de probabilidades de co-ocurrencia en lugar de las probabilidades en sí mismas. Por lo tanto, se define la matriz de recuentos de co-ocurrencia palabra-palabra como X, cuyas entradas X_{ij} registran la cantidad de veces que la palabra j ocurre en el contexto de la palabra i. Además, se define como $X_i = \sum_k X_{ik}$ a la cantidad de veces que cualquier palabra aparece en el contexto de la palabra i. Por último, se tiene que $P_{ij} = P(j|i) = X_{ij}/X_i$. Utilizando las definiciones anteriores, surge la siguiente ecuación para definir un modelo general inicial,

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{2.17}$$

en donde $w \in \mathbb{R}^d$ son vectores de palabras y $\tilde{w}_k \in \mathbb{R}^d$ son vectores de palabras de contexto separados. Este modelo se refina en base a diversos supuestos y condiciones, obteniendo un modelo de regresión de mínimos cuadrados ponderados,

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2 \tag{2.18}$$

en donde V es el tamaño del vocabulario y b son los sesgos introducidos en el modelo. Por último, la función f se define como

$$f(x) = \begin{cases} (x/x_{max})^\alpha & \text{si } x < x_{max} \\ 1 & \text{de lo contrario.} \end{cases} \tag{2.19}$$

2.7.3. *FastText*

Los modelos populares de representación de palabras como vectores hasta el año 2017 presentan el problema de que ignoran la morfología de las palabras al asignar un vector distinto para cada una, lo cual se convierte en una limitación, especialmente para lenguajes con vocabularios extensos y con muchas palabras poco comunes. La librería *FastText* [14] surge para solucionar este problema basándose en el modelo *Skip-gram*.

El modelo propuesto aprende vectores de palabras tomando en cuenta la morfología de estas. Esto lo logra mediante el aprendizaje de vectores para cada carácter de largo n correspondientes a una palabra, y en donde la palabra es representada como una bolsa de caracteres de largo n. En adición, la palabra también es agregada al conjunto de caracteres para aprender la representación de la misma. Por ejemplo, tomando la palabra *where* y un $n = 3$, esta sería representada por los caracteres:

$$< wh, whe, her, ere, re > \tag{2.20}$$

y la secuencia especial

$$< where > . \tag{2.21}$$

Hay que notar que la secuencia $< her >$ correspondiente a la palabra *her* es diferente al carácter de largo $3 < ..., her, ... >$ de la palabra *where*. Finalmente, teniendo un diccionario de tamaño G compuesto de caracteres de largo n y dada una palabra w, se define como $\mathcal{G}_w \subset \{1, ..., G\}$ al conjunto de *n-gramas* (conjunto de caracteres de largo n) que aparecen en w. Luego, se asocia una representación vectorial z_g a cada *n-grama* g. Finalmente, una palabra es representada por la suma de las representaciones vectoriales de sus *n-gramas*, obteniendo así la función de puntuación:

$$s(w, c) = \sum_{g \in \mathcal{G}_w} z_g^T v_c. \tag{2.22}$$

2.7.4. BERT (Bidirectional Encoder Representations from Transformers)

BERT [15] es un modelo de representación de lenguaje introducido el año 2018, que a diferencia de los modelos anteriores, fue diseñado para pre-entrenar representaciones bidireccionales profundas a partir de texto sin etiquetar.

Existen dos estrategias distintas para aplicar representaciones de lenguaje pre-entrenadas a tareas específicas: *feature-based* y *fine-tuning*. La primera consiste en utilizar arquitecturas de tareas específicas que incluyen estas representaciones como características adicionales. La segunda, por otro lado, introduce una pequeña cantidad de parametros para la tarea específica, y se entrena el modelo sobre los parametros ya pre-entrenados.

BERT surge del argumento de que estas estrategias restringen el poder de las representaciones pre-entrenadas, especialmente en la estrategia de *fine-tuning*, ya que durante el pre-entrenamiento, los modelos estándares se encuentran limitados en cuanto a arquitectura al ser unidireccionales.

Existen dos pasos esenciales en el *framework* de BERT: *pre-training* y *fine-tuning*. Durante el primero, el modelo es entrenado con datos no etiquetados sobre dos tareas de pre-entrenamiento, modelamiento de lenguaje enmascarado y predicción de la siguiente oración. Para el segundo paso, el modelo es inicializado con los parametros obtenidos en el proceso de pre-entrenamiento , y se realiza un *fine-tuning*, esta vez utilizando los datos etiquetados de distintas tareas, como por ejemplo, la tarea de *question-answering*.

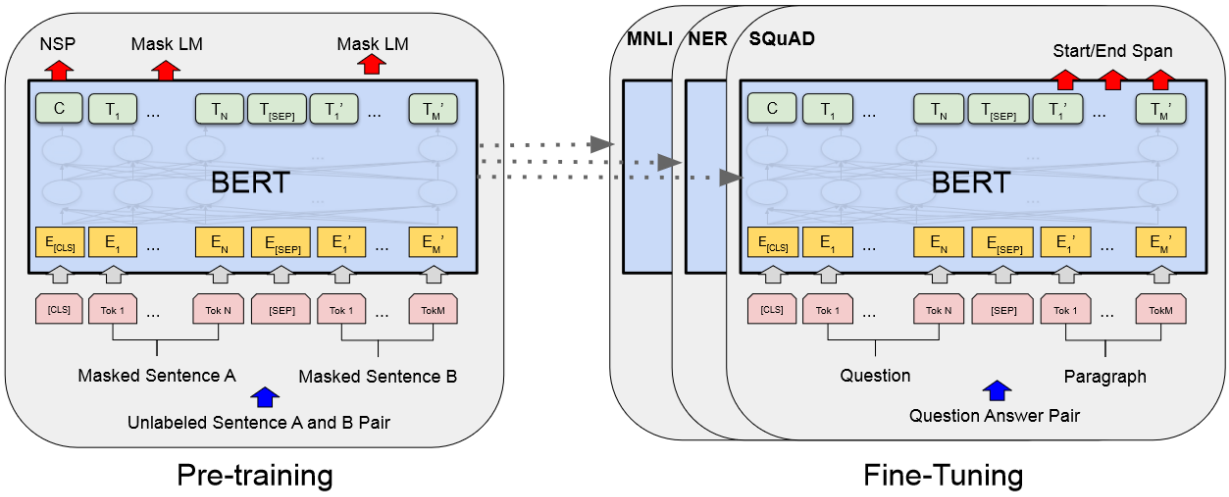


Figura 2.9: Procedimientos de *pre-training* y *fine-tuning* presentados en *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* [15].

2.7.5. RoBERTa (Robustly Optimized BERT Pretraining Approach)

RoBERTa [16] aparece en el año 2019 luego de una replicación del estudio realizado en *BERT* [15] el año 2018. Se argumenta que existe una falta de entrenamiento considerable en *BERT* y se propone una nueva receta para entrenar estos modelos, la cual iguala e incluso mejora los resultados obtenidos por *BERT*.

Las modificaciones propuestas incluyen, entre otras:

- Entrenar el modelo durante más tiempo, con *batches* más grandes, lo cual mejora la perplejidad del modelo y los resultados obtenidos en las *end-tasks*. Además, se decide usar más datos para el entrenamiento. Los *corpus* ocupados durante el entrenamiento están compuestos por *BookCorpus* [17] más Wikipedia en inglés (estos fueron los datos utilizados para entrenar *BERT*); *CC-News*, correspondiente a los artículos de noticias en inglés obtenidos del *dataset CommonCrawl News* [18]; *OpenWebText* [19], construido a partir de *URLs* publicadas en *Reddit*; y por último, *Stories*, un dataset introducido en la publicación *A simple method for commonsense reasoning* [20].

- Usar una máscara dinámica en los datos de entrenamiento. *BERT* genera las máscaras una única vez durante el pre-procesamiento de los datos, sin embargo, duplicaban estos datos 10 veces, para evitar usar la misma máscara durante el entrenamiento. *RoBERTa*, por otro lado, propone generar la máscara dinámicamente, es decir, generar la máscara cada vez que se ingresa una secuencia al modelo durante el entrenamiento.

- Remover el objetivo de predecir la próxima oración durante el entrenamiento, lo cual iguala o mejora levemente los resultados de las *downstream tasks*.

2.7.6. MPNet (Masked and Permuted Pre-training for Language Understanding)

En noviembre del año 2020 surge *MPNet* [21], modelo que hereda las ventajas de *BERT* y *XLNet* [22] y evita sus limitaciones. En particular, como ya se ha mencionado, *BERT* utiliza un modelamiento de lenguaje enmascarado para su pre-entrenamiento, sin embargo, este enfoque ignora la dependencia entre los *tokens* enmascarados, los cuales serán predecidos.

Para solucionar la desventaja anteriormente mencionada, *XLNet* propone el modelamiento de lenguaje permutado para el pre-entrenamiento. Sin embargo, este método presenta una limitación: cada *token* puede ver únicamente los *tokens* anteriores a él en una secuencia permutada, pero no conoce la posición de los *tokens* futuros, lo cual sí es conocido en *BERT* al momento de predecir el *token* enmascarado.

MPNet propone unificar el modelamiento de lenguaje enmascarado y el modelamiento de lenguaje permutado, lo cual divide los *tokens* en una secuencia en partes predichas y no predichas. Con la propuesta presentada en este estudio, se logra:

- Tomar en consideración la dependencia de los *tokens* predichos mediante el uso del modelamiento de lenguaje permutado, evitando el problema de *BERT*.
- Usar la información posicional como entrada, para hacer que el modelo observe esta información para todos los *tokens*, y así, solucionar el problema presente en *XLNet*.

2.8. CLIP: *Learning transferable visual models from natural language supervision*

CLIP [1] es un modelo multi-modal reciente, presentado el año 2021, el cual se basa en la restricción inherente del estado del arte para los sistemas de visión por computadora, que son entrenados para predecir un conjunto fijo de categorías para objetos, limitando su generalidad y usabilidad debido a que necesitan datos etiquetados adicionales para especificar cualquier otro concepto.

El enfoque de *CLIP* se considera que aprender directo desde texto relacionado a imágenes es una alternativa prometedora, y demuestra que la tarea de pre-entrenamiento de predecir cual texto corresponde a cual imagen es una forma eficiente y escalable de aprender representaciones de imagen de estado del arte, desde cero, con un *dataset* de 400 millones de pares (imagen, texto) que estan disponibles públicamente en internet. La idea en que se basa CLIP para lograr esto, es realizar un pre-entrenamiento usando un *encoder* de imágenes y otro de texto, e intentando optimizar el modelo para lograr que los dos *encoders* aprendan un espacio vectorial parecido, en donde la salida del *encoder* de texto para la descripción de una imagen sea un *embedding* parecido al obtenido por el *encoder* visual cuya entrada era la imagen descrita. La función de pérdida utilizada por CLIP para resolver este problema se llama *symmetric loss*, y se define como:

$$\mathcal{L} = \frac{\textit{CrossEntropyLoss}(X, z) + \textit{CrossEntropyLoss}(Y, z)}{2} \tag{2.23}$$

en donde X es el valor real para el espacio visual, Y el valor real para el espacio de texto, y z un vector $z = [0, 1, ..., N-1]$, con N igual al tamaño del *batch* utilizado durante el entrenamiento.

Este nuevo modelo es competitivo al ser comparado con modelos supervisados, pero se menciona que funciona bien únicamente para predecir las 1000 categorías de *ImageNet* sin realizar un *fine-tuning* correspondiente. Sin embargo, CLIP no se ha probado dentro del contexto de recuperación de imágenes, lo cual deja abierta la posibilidad de estudiar su viabilidad.

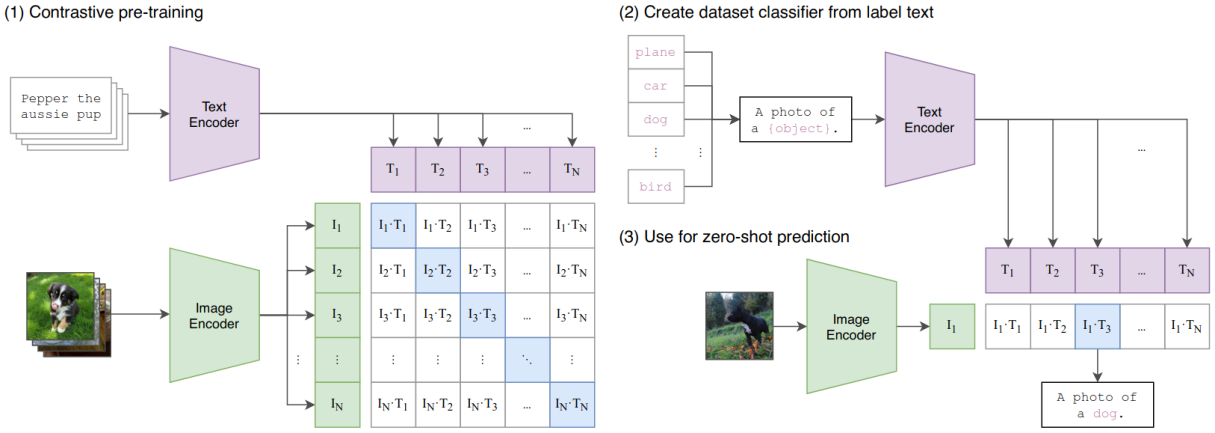


Figura 2.10: Resumen del enfoque de CLIP presentado en *Learning Transferable Visual Models From Natural Language Supervision* [1].

3. Estudios preliminares

3.1. Creación de *datasets*

Los datasets que se utilizan corresponden a catálogos de comercios electrónicos de distinta índole y naturaleza, incluyendo catálogos tanto de joyería, ropa, muebles, entre otros. Los datos se obtienen utilizando herramientas de *web scraping*, en particular, las librerías de Python *BeautifulSoup* y *Selenium*, la primera para extraer datos de páginas web estáticas, y la segunda para páginas web dinámicas.

Todos los productos pertenecientes a cada *dataset* son categorizados de dos maneras: por un lado, mediante una categoría global (*GlobalCategory*), la cual representa una clasificación de grano grueso de cada producto dentro de su *dataset* correspondiente; por otro lado, existe un árbol de categoría *CategoryTree*, el cual representa una clasificación más detallada o de grano fino, que se conforma de diversas subcategorías. Para todos los *datasets* las categorías se crean mediante los *breadcrumbs* presentes en la página específica de cada producto, a excepción del *dataset* de Pepeganga, cuyas categorías se tienen de antemano.

Para cada *dataset* se aíslan 100 productos aleatorios desde un inicio, los cuales son utilizados como *queries* a evaluar mediante la recuperación de imágenes, y no se contabilizan para estadísticas generales del *dataset*. En adición a esto, dos *datasets* contienen *queries* de evaluación con imágenes de productos reales, categorizados a mano, para poder realizar una evaluación de recuperación más cercana a la realidad.

3.1.1. Dataset de Pepeganga

Este corresponde al primer *dataset* con el que se trabaja y se realizan los experimentos iniciales. Desde el inicio del trabajo se posee un archivo tipo *Excel* proporcionado por *Impresee* con direcciones URL de los productos y sus respectivas imágenes, además de información de cada producto, como su nombre, categorías, entre otros. Pepeganga está formado por un catálogo variado, teniendo principalmente productos para el hogar, juguetes, videojuegos y ropa. Algunos productos pertenecientes al *dataset* de Pepeganga pueden observarse en la Figura 3.1.

Figura 3.1: Ejemplos de productos del *dataset* de Pepeganga.

El conjunto de datos de Pepeganga S.A. presenta las siguientes características:

- Catálogo compuesto por 22558 productos diferentes.
- Largo promedio de la descripción de producto: 58 palabras.
- Un número total de 15 *GlobalCategory* distintas.
- Un número total de 440 *CategoryTree* distintas.

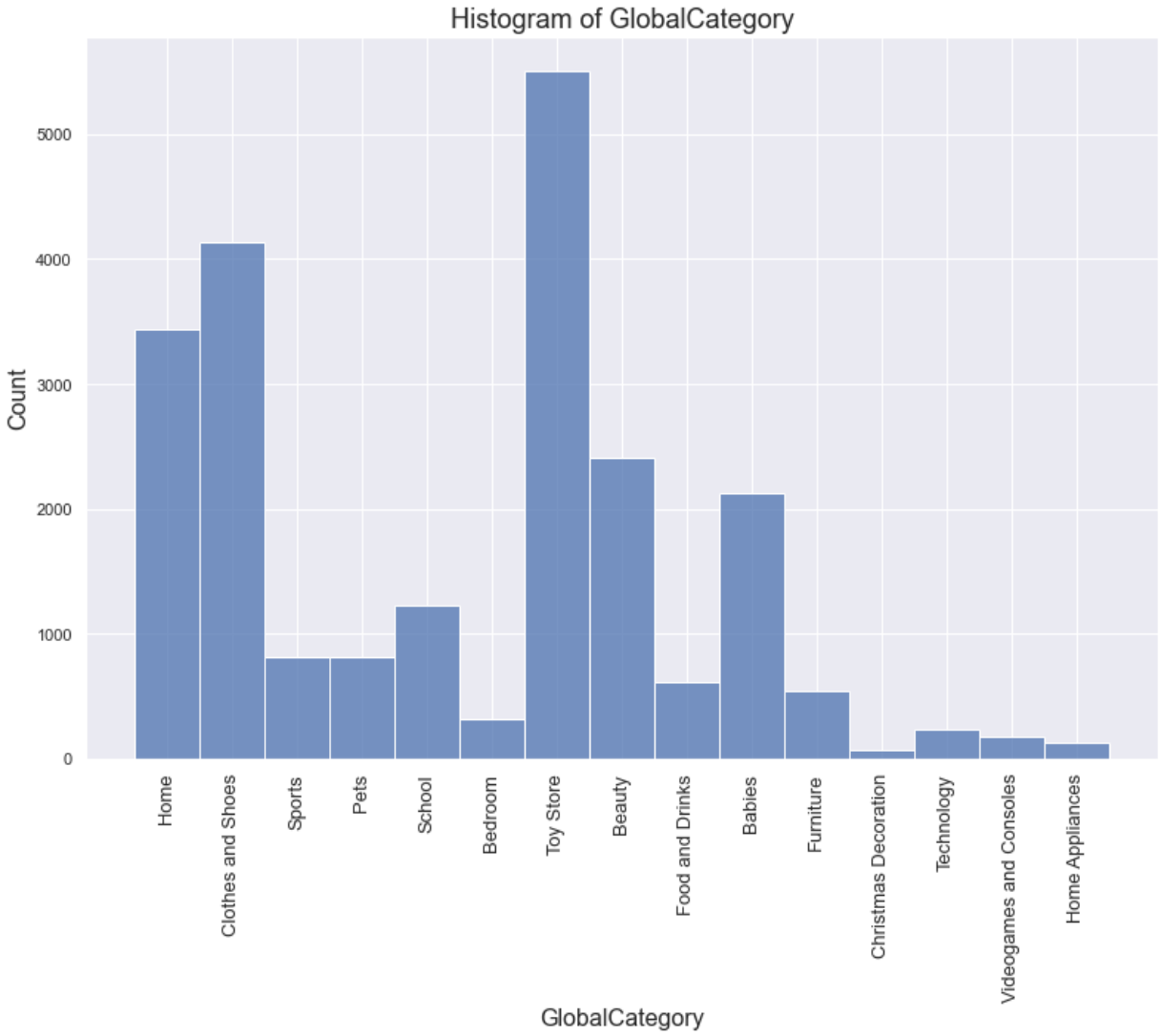


Figura 3.2: Histograma de productos por *GlobalCategory* del *dataset* de Pepeganga.

La distribución de los productos separados por *GlobalCategory* se pueden observar en la figura 3.2, cuyos datos específicos se encuentran en la tabla 3.1 a continuación:

Tabla 3.1: Cantidad de productos por *GlobalCategory* del *dataset* de Pepeganga.

GlobalCategory	**Cantidad**
Toy Store	5508
Clothes and Shoes	4131
Home	3443
Beauty	2404
Babies	2129
School	1227
Pets	812
Sports	809
Food and Drinks	617
Furniture	543
Bedroom	317
Technology	239
Videogames and Consoles	177
Home Appliances	134
Christmas Decoration	68

El *dataset* de Pepeganga contiene además 300 *queries* con imágenes reales de evaluación categorizadas utilizando la categoría de *GlobalCategory*. Estas *queries* se extraen desde *Facebook Marketplace*, y el sitio de compra y venta chileno *yapo.cl*, y se escogen acorde al contexto del *dataset*. La extracción de datos se realiza mediante el uso de las mismas herramientas de *web scraping* mencionadas anteriormente.

El sitio web de Pepeganga contiene descripciones de productos en español, por lo que estas se deben traducir al inglés usando la librería de *Python deep-translator*, la cual ofrece múltiples traductores conocidos. Para este caso, se utiliza *Google Translate*.

3.1.2. Dataset de IKEA

El *dataset* de IKEA está compuesto por electrodomésticos, muebles, colchones y objetos para el hogar, y algunos de sus productos pueden observarse en la Figura 3.3.

Figura 3.3: Ejemplos de productos del *dataset* de IKEA.

Este *dataset* presenta las siguientes características:

- Catálogo compuesto por 5606 productos diferentes.
- Largo promedio de la descripción de producto: 28 palabras.
- Un número total de 18 *GlobalCategory* distintas.
- Un número total de 602 *CategoryTree* distintas.

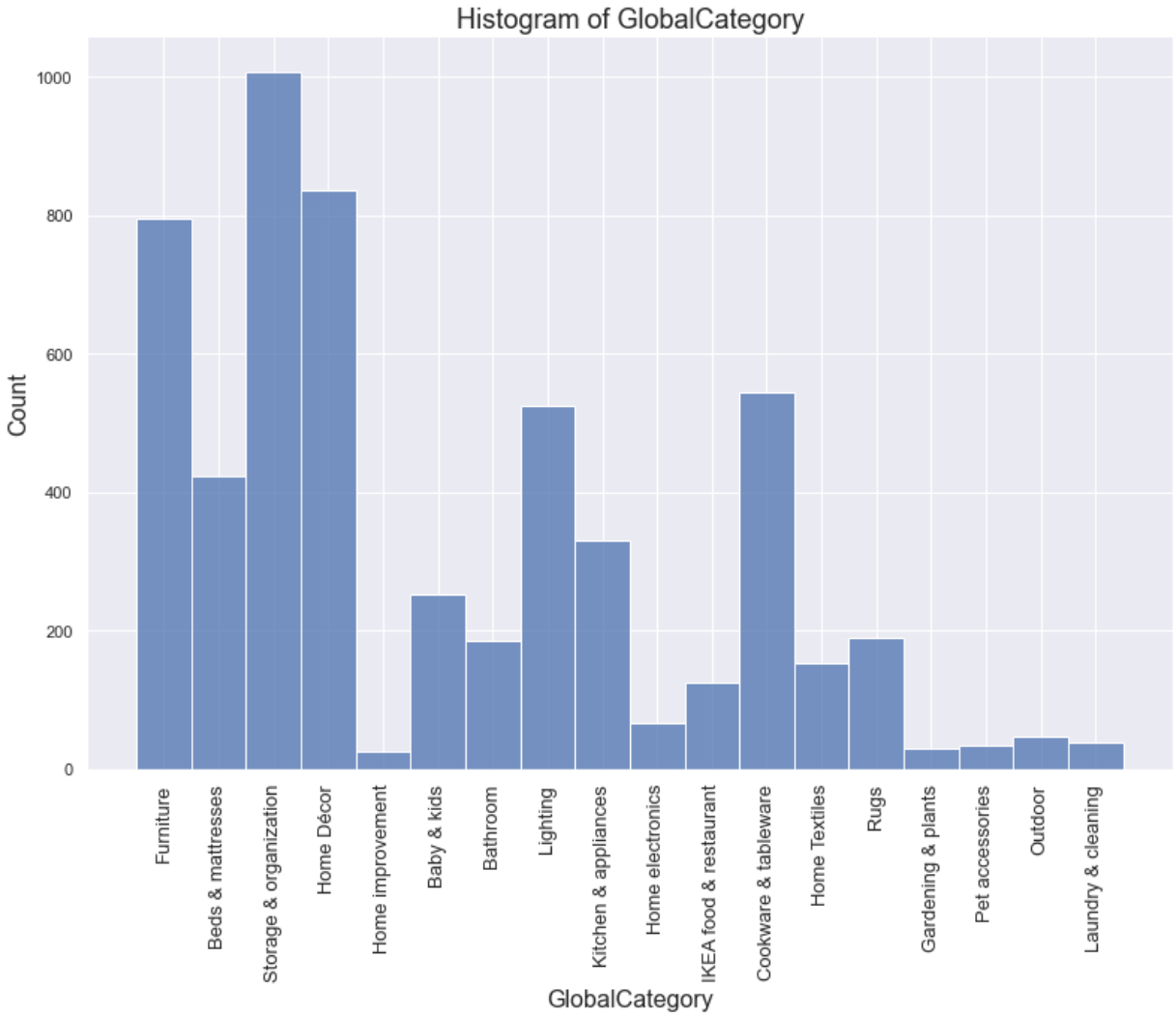


Figura 3.4: Histograma de productos por *GlobalCategory* del *dataset* de IKEA.

La distribución de los productos separados por *GlobalCategory* se pueden observar en la figura 3.4, cuyos datos específicos se encuentran en la tabla 3.2 a continuación:

Tabla 3.2: Cantidad de productos por *GlobalCategory* del *dataset* de IKEA.

GlobalCategory	**Cantidad**
Storage & Organization	1007
Home Décor	836
Furniture	794
Cookware & tableware	545
Lighting	524
Beds & mattresses	424
Kitchen & appliances	330
Baby & kids	252
Rugs	189
Bathroom	185
Home Textiles	153
IKEA food & restaurant	125
Home electronics	66
Outdoor	48
Laundry & cleaning	39
Pet accessories	34
Gardening & plants	30
Home improvement	25

Este *dataset* no contiene *queries* reales de evaluación, y las descripciones de los productos en el sitio web se encuentran en inglés.

3.1.3. Dataset de *World Market*

Este *dataset* contiene muebles para el hogar, de decoración, alfombras, regalos, cortinas, bebestibles y comida de distinta índole, y algunos de sus productos pueden observarse en la Figura 3.5.

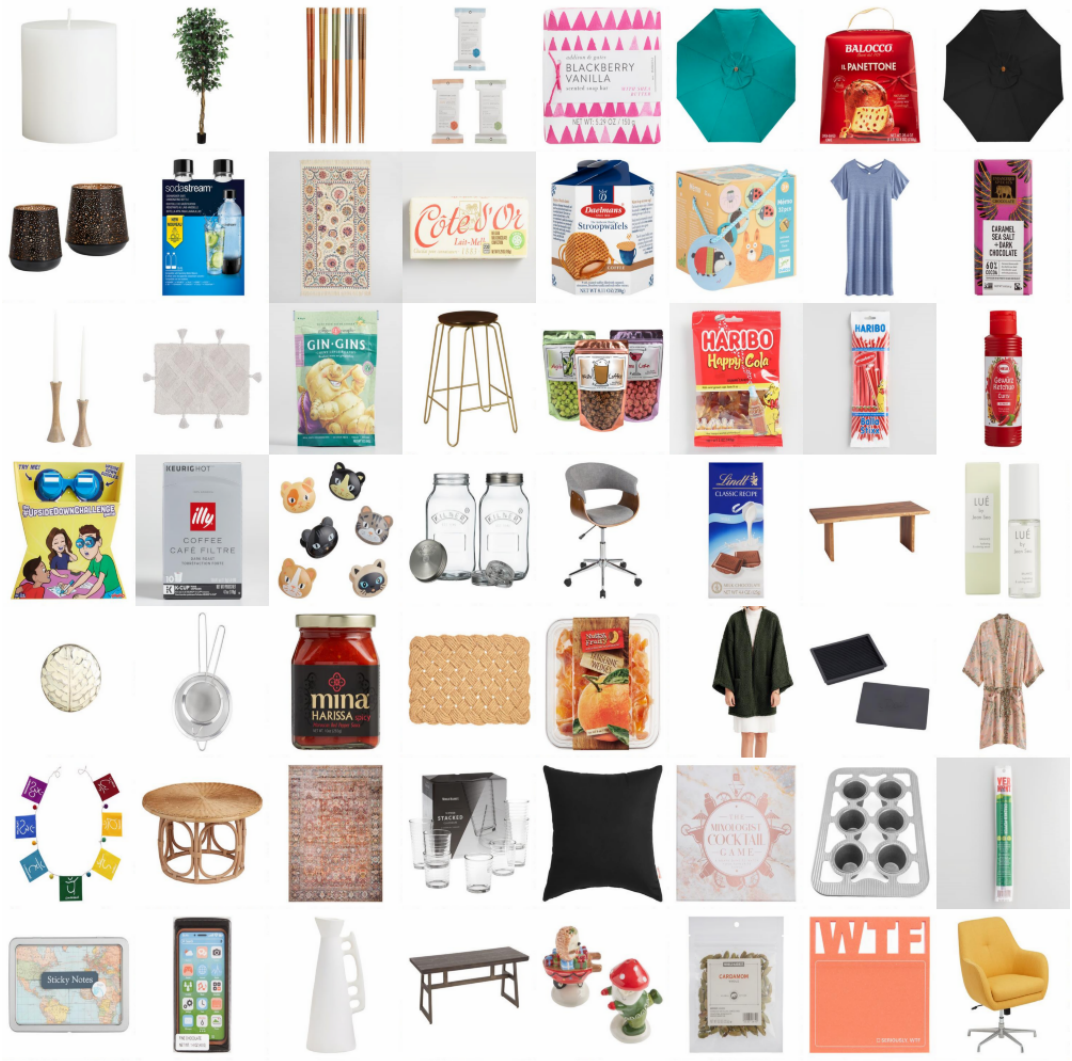


Figura 3.5: Ejemplos de productos del *dataset* de *World Market*.

El *dataset* de *World Market* presenta las siguientes características:

- Catálogo compuesto por 10056 productos diferentes.
- Largo promedio de la descripción de producto: 48 palabras.
- Un número total de 12 *GlobalCategory* distintas.
- Un número total de 356 *CategoryTree* distintas.

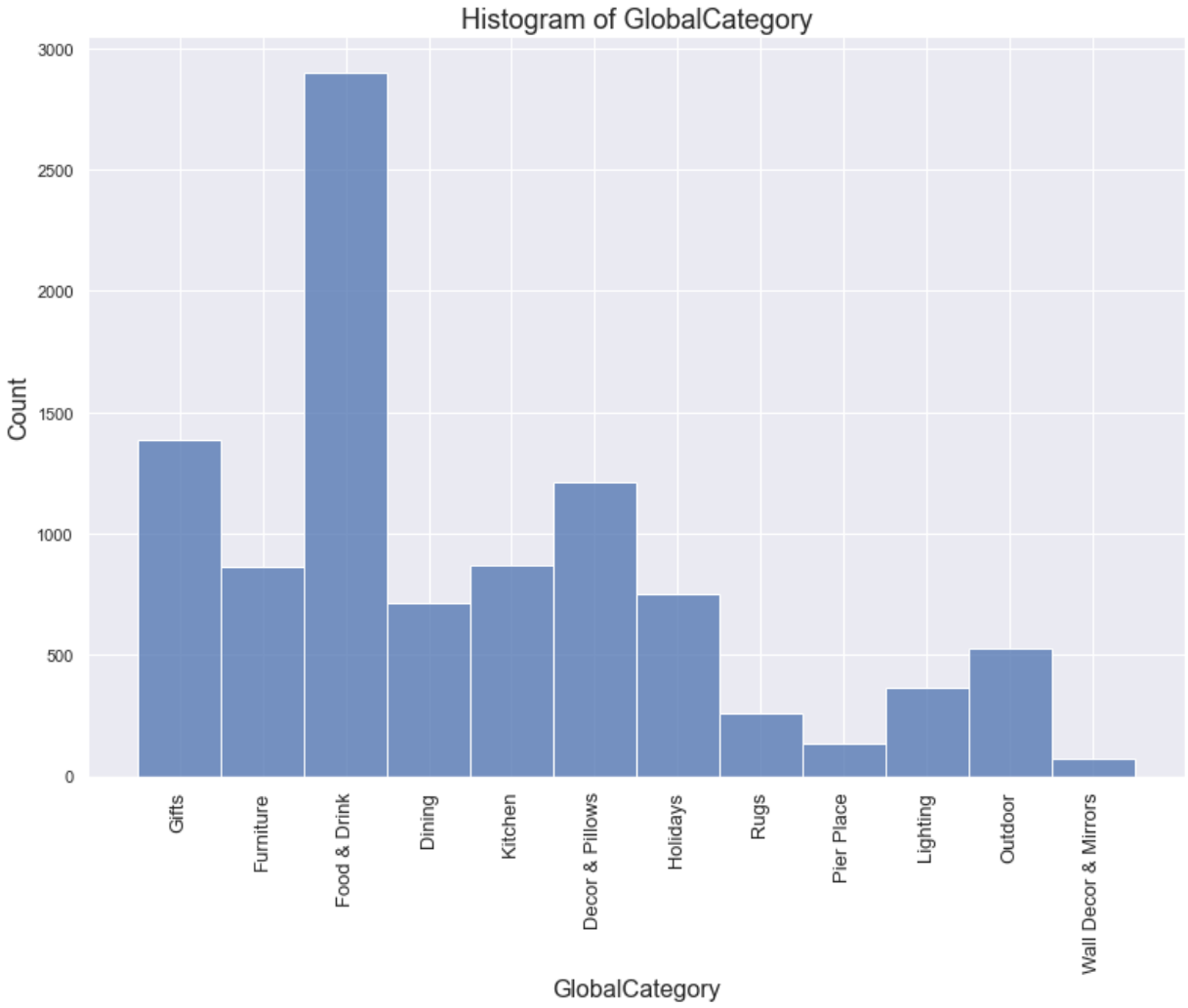


Figura 3.6: Histograma de productos por *GlobalCategory* del *dataset* de *World Market*.

La distribución de los productos separados por *GlobalCategory* se pueden observar en la figura 3.6, cuyos datos específicos se encuentran en la tabla 3.3 a continuación:

Tabla 3.3: Cantidad de productos por *GlobalCategory* del *dataset* de *World Market*.

GlobalCategory	**Cantidad**
Food & Drink	2902
Gifts	1389
Decor & Pillows	1211
Kitchen	867
Furniture	863
Holidays	753
Dining	715
Outdoor	524
Lighting	368
Rugs	260
Pier Place	132
Wall Decor & Mirror	72

Este *dataset* no contiene *queries* reales de evaluación, y las descripciones de los productos en el sitio web se encuentran en inglés.

3.1.4. Dataset de Homy

El *dataset* de Homy posee productos para el hogar, muebles, electrodomésticos, y productos de decoración, y algunos de sus productos pueden observarse en la Figura 3.7.

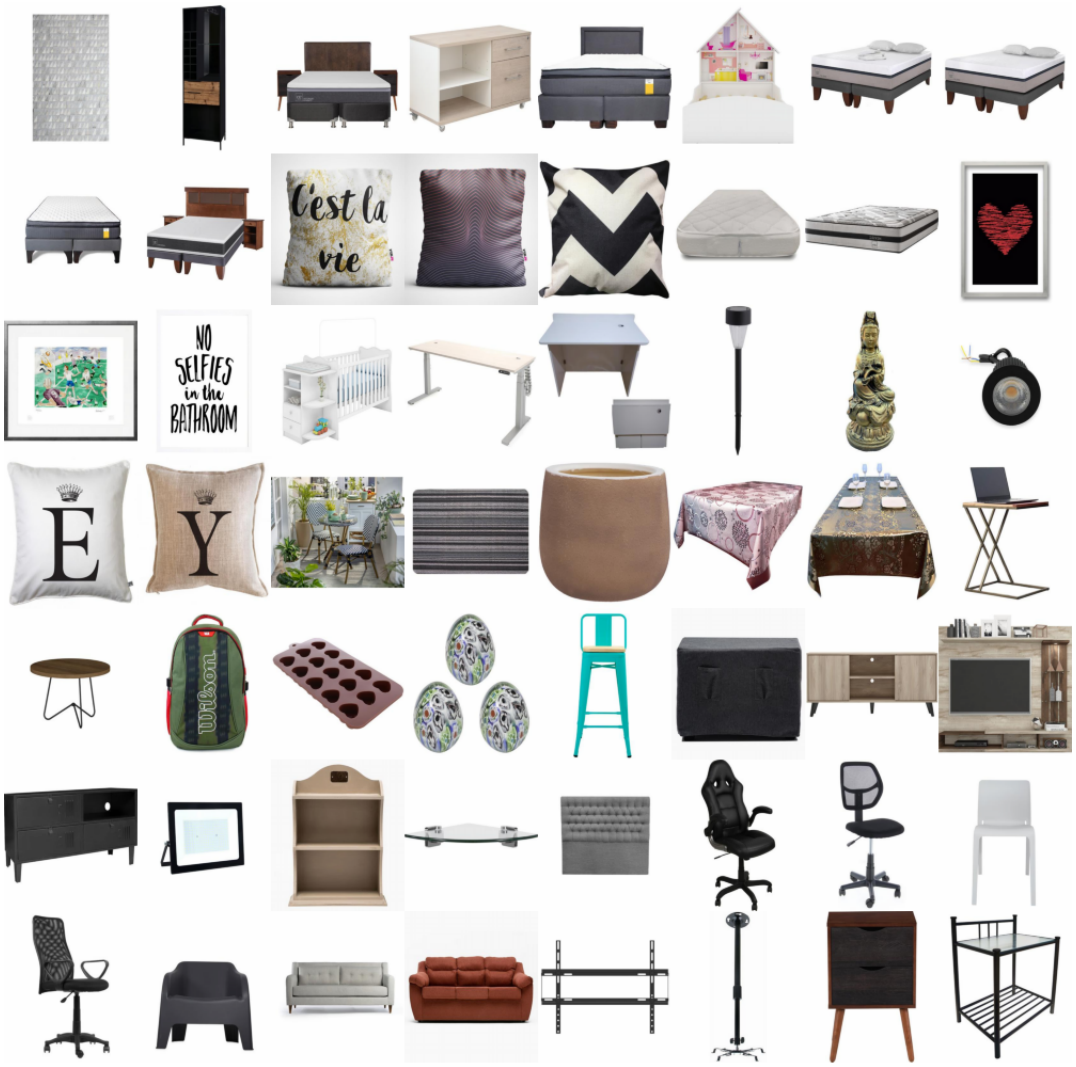


Figura 3.7: Ejemplos de productos del *dataset* de Homy.

El *dataset* de Homy presenta las siguientes características:

- Catálogo compuesto por 11181 productos diferentes.
- Largo promedio de la descripción de producto: 76 palabras.
- Un número total de 13 *GlobalCategory* distintas.
- Un número total de 311 *CategoryTree* distintas.

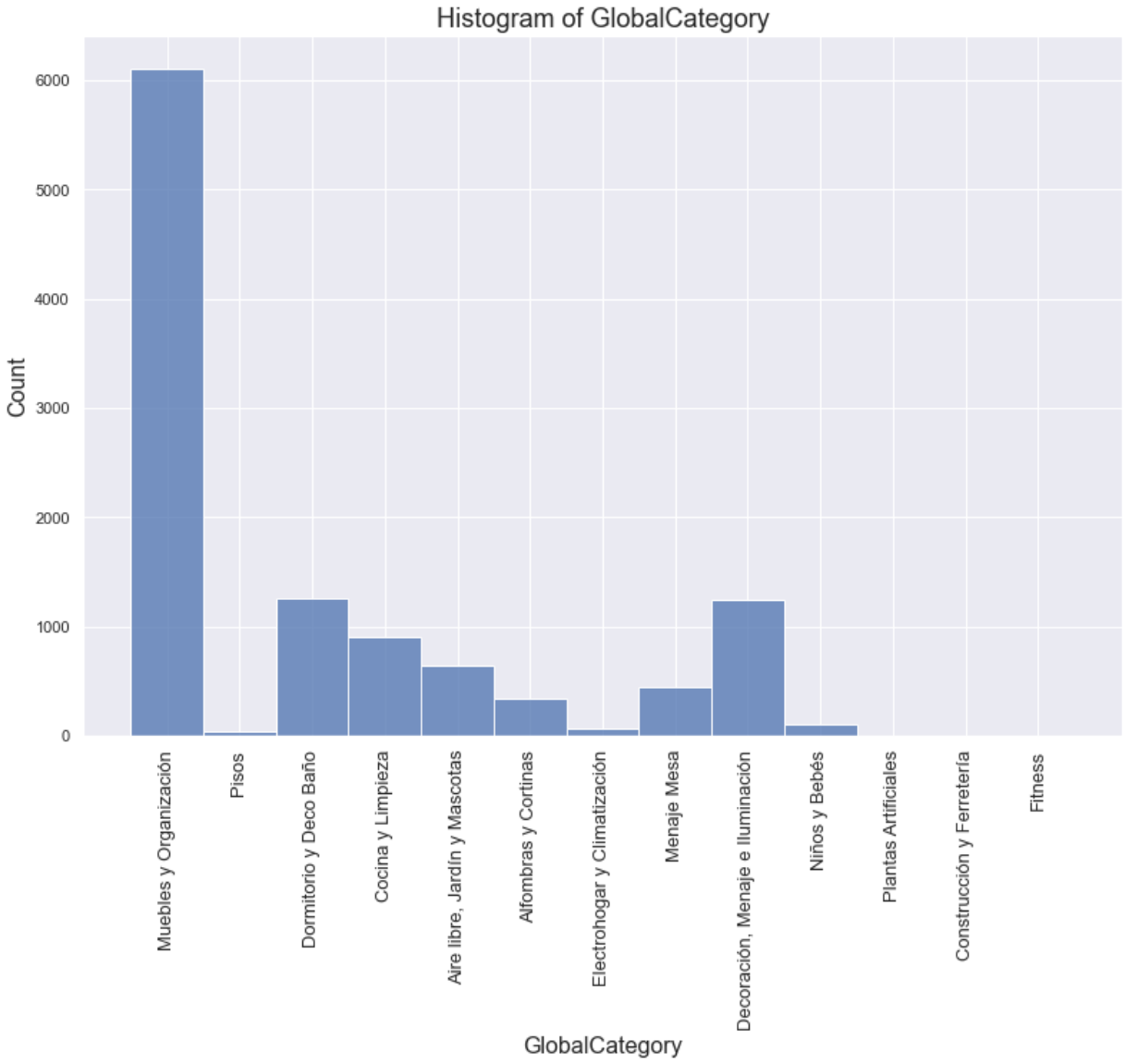


Figura 3.8: Histograma de productos por *GlobalCategory* del *dataset* de Homy.

La distribución de los productos separados por *GlobalCategory* se pueden observar en la figura 3.8, cuyos datos específicos se encuentran en la tabla 3.4 a continuación:

Tabla 3.4: Cantidad de productos por *GlobalCategory* del *dataset* de Homy.

GlobalCategory	**Cantidad**
Muebles y Organización	6100
Dormitorio y Deco Baño	1254
Decoración, Menaje e Iluminación	1245
Cocina y Limpieza	911
Aire libre, Jardín y Mascotas	640
Menaje Mesa	447
Alfombras y Cortinas	341
Niños y Bebés	113
Electrohogar y Climatización	69
Pisos	43
Construcción y Ferretería	9
Plantas Artificiales	8
Fitness	1

El *dataset* de Homy es el segundo que posee 256 *queries* reales de evaluación categorizadas mediante la categoría de *GlobalCategory*.

El sitio web de Homy contiene descripciones de productos en español, por lo que estas también se deben traducir al inglés utilizando *Google Translate* a través de la librería *deep-translator*.

3.1.5. Dataset de Cartier

Este *dataset* está compuesto por joyas, relojes, carteras, billeteras, entre otros, y algunos de sus productos pueden observarse en la Figura 3.9.

Figura 3.9: Ejemplos de productos del *dataset* de Cartier.

El *dataset* de Cartier presenta las siguientes características:

- Catálogo compuesto por 1759 productos diferentes.
- Largo promedio de la descripción de producto: 33 palabras.
- Un número total de 5 *GlobalCategory* distintas.
- Un número total de 143 *CategoryTree* distintas.

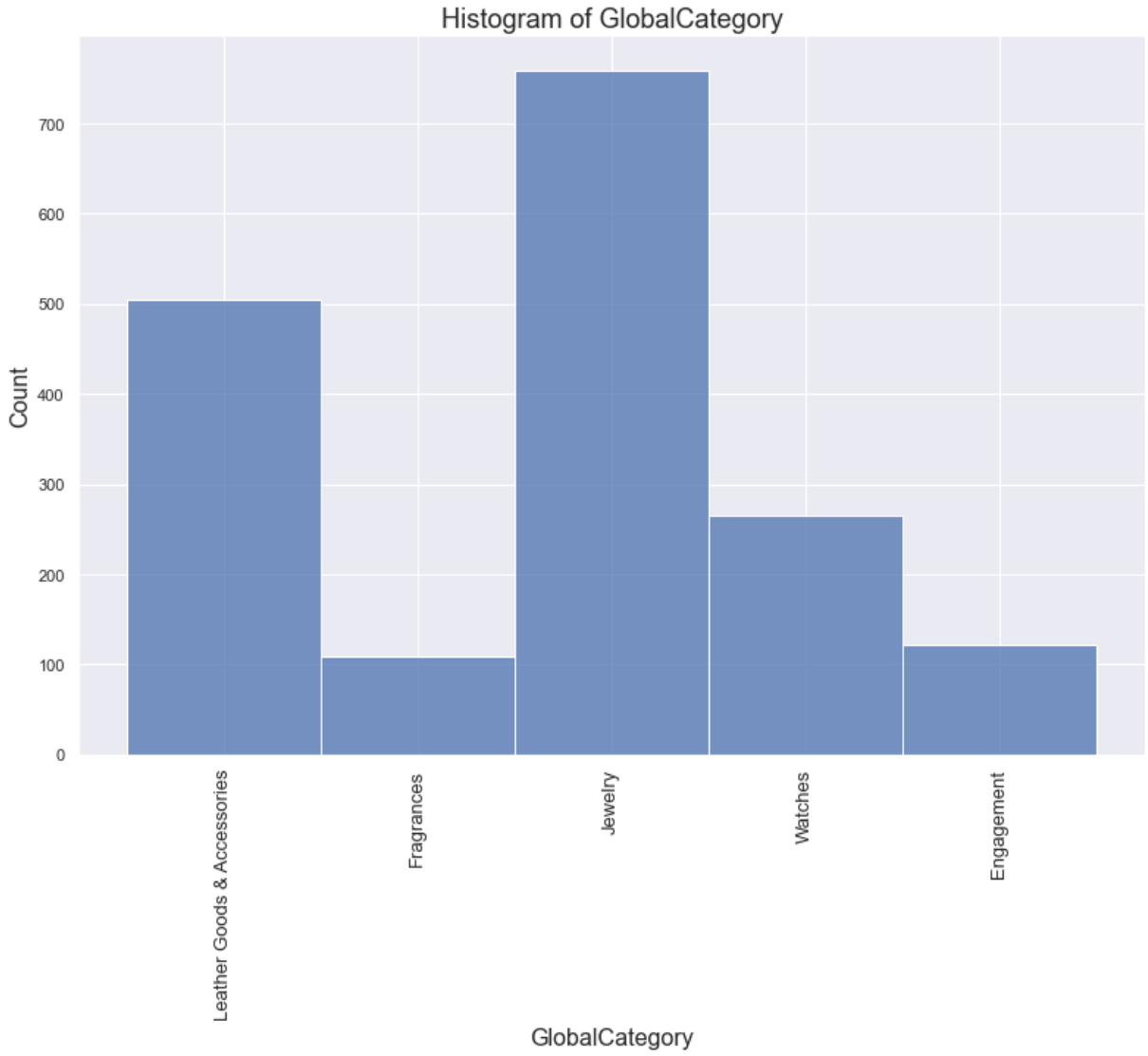


Figura 3.10: Histograma de productos por *GlobalCategory* del *dataset* de Cartier.

La distribución de los productos separados por *GlobalCategory* se pueden observar en la figura 3.10, cuyos datos específicos se encuentran en la tabla 3.5 a continuación:

Tabla 3.5: Cantidad de productos por *GlobalCategory* del *dataset* de Cartier.

GlobalCategory	**Cantidad**
Jewelry	759
Leather Goods & Accessories	505
Watches	265
Engagement	122
Fragrances	108

Este *dataset* no contiene *queries* reales de evaluación, y las descripciones de los productos en el sitio web se encuentran en inglés.

3.1.6. Dataset de UNIQLO

El *dataset* de UNIQLO posee productos de ropa, y algunos de sus productos pueden observarse en la Figura 3.11.

Figura 3.11: Ejemplos de productos del *dataset* de UNIQLO.

El *dataset* de UNIQLO presenta las siguientes características:

- Catálogo compuesto por 1425 productos diferentes.
- Largo promedio de la descripción de producto: 54 palabras.
- Un número total de 5 *GlobalCategory* distintas.
- Un número total de 281 *CategoryTree* distintas.

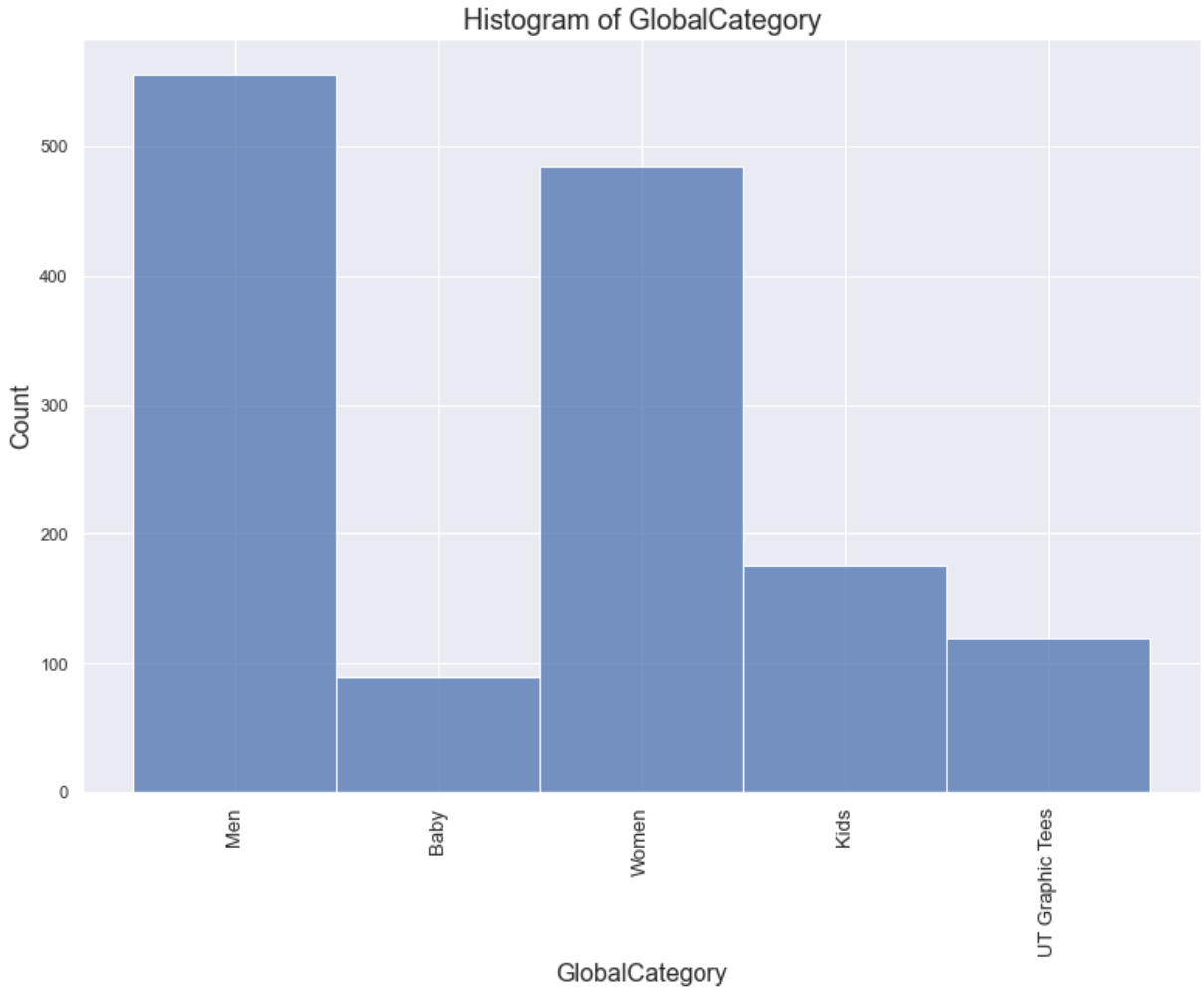


Figura 3.12: Histograma de productos por *GlobalCategory* del *dataset* de UNIQLO.

La distribución de los productos separados por *GlobalCategory* se pueden observar en la figura 3.12, cuyos datos específicos se encuentran en la tabla 3.6 a continuación:

Tabla 3.6: Cantidad de productos por *GlobalCategory* del *dataset* de UNIQLO.

GlobalCategory	**Cantidad**
Men	556
Women	484
Kids	176
UT Graphic Tees	119
Baby	90

Este *dataset* no contiene *queries* reales de evaluación, y las descripciones de los productos en el sitio web se encuentran en inglés.

3.2. Recuperación de imágenes con ResNet-50

Como *baseline* para la recuperación de imágenes en este trabajo se decide utilizar un modelo pre-entrenado con los datos de *ImageNet*, el cual presenta una arquitectura ResNet-50.

Para recuperar productos en base a una imagen, llamada *query*, esta es procesada por la ResNet-50, obteniéndose un vector de características de 2048 dimensiones. Luego, se calcula la distancia euclidiana entre este vector y el catálogo del *dataset* correspondiente, la cual está definida por:

$$d(X, Y) = \left(\sum_{k=1}^{n} |X_k - Y_k|^2 \right)^{1/2} \tag{3.1}$$

donde X corresponde al vector de características ingresado como *query*, e Y el vector de características de algún producto del catálogo. Finalmente, los productos recuperados se ordenan ascendentemente en base a la distancia euclideana, es decir, los productos cuyas imágenes poseen *embeddings* con menor distancia al *embedding* de la imagen ingresada como *query* son retornados primero.

3.3. Recuperación de productos mediante texto

Para apreciar y evaluar la diferencia de semántica existente entre las imágenes y las descripciones de los productos, se decide implementar una recuperación de productos mediante texto. Esta es definida de igual manera que la recuperación de imágenes con ResNet-50, sin embargo, los vectores de características son obtenidos en base a la descripción de cada producto. Además, en vez de utilizar la distancia euclidiana para encontrar los productos más cercanos, se decide utilizar la similitud coseno, la cual se define como:

$$S_C(X, Y) = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2} \sqrt{\sum_{i=1}^{n} Y_i^2}} \tag{3.2}$$

en donde X corresponde al vector de características obtenido en base a la descripción ingresada como *query*, e Y el vector de características de algún producto en particular, obtenido en base a su descripción. Así, la recuperación retorna en orden descendente los productos cuya similitud coseno sea mayor respecto al *embedding* ingresado como *query*.

Para este punto, mediante la librería de código abierto especializada en la representación de texto *huggingface*, se utilizan tres modelos clásicos para la generación de *embeddings* de texto ya entrenados, definidos a continuación:

- FastText: vectores de 300 dimensiones entrenados con los datos de Wikipedia 2017, *UMBC web-based corpus*, y *Statistical Machine Translation news dataset*.
- GloVe: vectores de 300 dimensiones entrenados en Wikipedia 2014 y Gigaword 5.

- Word2Vec: vectores de 300 dimensiones entrenados con datos de *Google News*.

Los vectores para cada palabra dentro de una descripción se promedian, generándose un vector para cada descripción. Se evalúa el modelo base original, correspondiente a una arquitectura ResNet-50 entrenada con los datos de *ImageNet*, y se contrastan los resultados con los obtenidos por cada modelo de texto mencionado anteriormente.

Más adelante en el desarrollo, se agregan cuatro modelos más recientes para crear representaciones de palabras, utilizando tanto *huggingface* como la librería *SentenceBert*, la cual ofrece facilidades para obtener vectores de oraciones directamente. Los modelos utilizados corresponden a los siguientes:

- MPNet-base: vectores de 768 dimensiones. El modelo fue entrenado con 1 billón de pares de oraciones, obtenidos a través de diversos *datasets* concatenados, utilizando comentarios de *Reddit*, *WikiAnswers*, textos *StackExchange*, entre otros.
- BERT-uncased: vectores de 768 dimensiones. El modelo fue entrenado con los datos de *BookCorpus*, un *dataset* consistente de 11038 libros no publicados, y con *English Wikipedia* (excluyendo listas, tablas y títulos).
- RoBERTa-large: vectores de 1024 dimensiones. Este modelo fue entrenado de la misma manera y con los mismos datos que MPNet-base.

3.4. Recuperación y visualización mediante UMAP

Un método auto-supervisado que permite la reducción de dimensionalidad en *embeddings*, preservando localidad, es UMAP (*Uniform Manifold Approximation and Projection*). De esta manera, motivado por la eficiencia generada al reducir las dimensiones de los vectores de características y la capacidad de UMAP para preservar la localidad de los datos, se implementa una recuperación utilizando tanto *embeddings* visuales como de texto, en donde nuevamente se decide utilizar la distancia euclidiana y la similitud coseno respectivamente.

Por último, dado que UMAP permite una reducción a 2 dimensiones, se opta por generar visualizaciones para poder observar los datos y su estructura. Además, de esta manera, se puede medir la capacidad representativa de estos *embeddings* mediante la generación de *clusters* en base a los mismos.

4. Métodos de ajuste de modelos

4.1. Ajuste de *embeddings* visuales dentro del catálogo

En base a la premisa de que las descripciones de los productos pueden agregar más semántica a los *embeddings* visuales, y así mejorar la recuperación de imágenes, se decide desarrollar un algoritmo que pueda captar la información entregada por los textos de un catálogo e incorporarla, mediante ajustes, en el *embedding* que representa la imagen de un producto.

De este modo, se define el algoritmo observado en la figura 4.1 para realizar el ajuste al *embedding* visual de un único producto en particular perteneciente al catálogo. Este algoritmo presenta tres variantes, diferenciadas únicamente en el último paso:

1. Se obtiene el vector de características visual de un producto del catálogo.
2. Se obtiene el vector de características de texto del mismo producto.
3. Utilizando el *embedding* de texto, se buscan los k vecinos más cercanos del catálogo en base a la similitud coseno.
4. Se obtienen los *embeddings* visuales de cada producto retornado en el paso anterior.
5. Se genera el *embedding* visual ajustado utilizando una de las tres variantes implementadas.

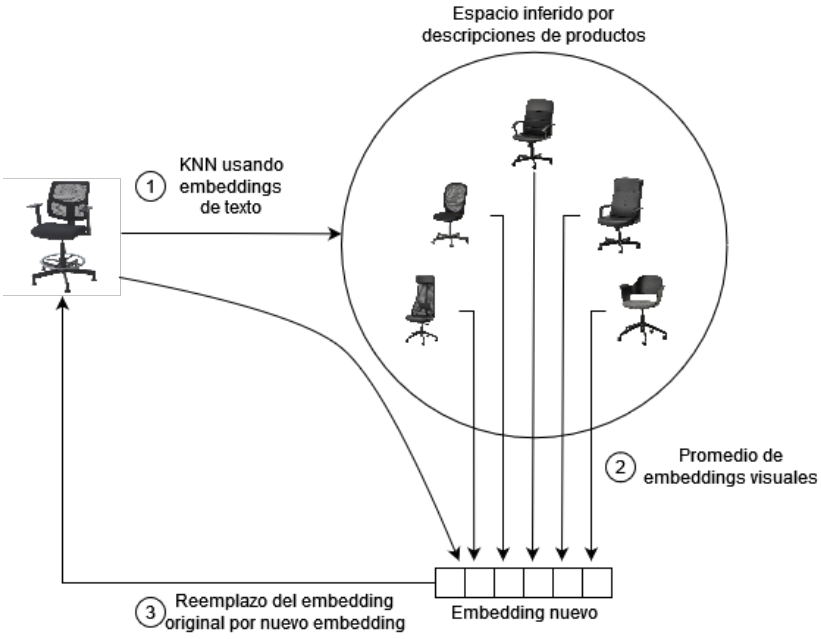


Figura 4.1: Algoritmo general de ajuste de *embeddings* visuales utilizando $k = 5$.

A continuación, se definen las tres variantes desarrolladas para obtener el *embedding* visual ajustado.

4.1.1. Promedio simple con *moving average*

Utilizando los *embeddings* visuales que se obtienen en el paso 4 del algoritmo definido anteriormente, se realiza un promedio simple entre estos, lo que significa promediar las componentes respectivas de cada vector. Luego, se calcula el *moving average* entre el *embedding* visual original del producto y el promedio de *embeddings* visuales. Es decir, definiendo un $\alpha \in [0.1, 0.9]$, los *embeddings* original y ajustado como E_{old} y E_{new} respectivamente, y el promedio de *embeddings* visuales de los k vecinos más cercanos como $\mu_{E_{knn}}$, se tiene que:

$$E_{new} = \alpha * E_{old} + (1 - \alpha) * \mu_{E_{knn}} \tag{4.1}$$

4.1.2. Promedio ponderado con similitud coseno

La segunda opción para generar el *embedding* ajustado es mediante un promedio ponderado, en donde las componentes respectivas de cada *embedding*, incluído el original, se multiplican por la respectiva similitud coseno normalizada del mismo. Esto provoca que el *embedding* original posea una ponderación mayor en el promedio.

4.1.3. Promedio ponderado con *softmax*

Como última opción, el *embedding* ajustado también se obtiene mediante un promedio ponderado. Sin embargo, los pesos son calculados a través de la fórmula de *softmax*, definida como:

$$W(c,T) = \frac{exp(\frac{c}{T})}{\sum_{i=1}^{n} exp(\frac{c_i}{T})} \tag{4.2}$$

donde c representa un vector con las k similitudes coseno más altas, incluída la similitud coseno máxima para el *embedding* original, y T un parámetro arbitrario entre $[0.5, 3.0]$.

4.2. *Query* adaptativa

Con la premisa de que un modelo con *embeddings* ajustados genera el mismo *embedding* para una *query* que un modelo no ajustado, se opta por desarrollar un algoritmo adaptativo para ajustar el *embedding* de un producto ingresado en tiempo real, es decir, el algoritmo decide cuando ajustar o no el *embedding* visual de una imagen que recibe como entrada. El algoritmo, luego de ingresar un producto como *query*, se puede observar en la Figura 4.2 y se define en los siguientes pasos:

1. Se obtiene el *embedding* de la *query* ingresada.
2. Se hace una recuperación de los tres productos más similares utilizando los *embeddings* originales (en caso de que los *embeddings* del catálogo hayan sido ajustados).
3. Si los tres productos retornados poseen la misma *GlobalCategory*, se procede con el algoritmo. En caso contrario, la *query* no se modifica y el algoritmo termina.
4. Si el catálogo posee *embeddings* ajustados, se obtienen estos para los tres productos obtenidos en el paso anterior. En caso contrario, se dejan los *embeddings* originales.
5. Se realiza un promedio simple para cada componente de cada *embedding*, incluyendo el de la *query* ingresada, obteniendo un *embedding* ajustado en tiempo real para la *query*.

Con este algoritmo, se espera mejorar los resultados que ya son relativamente buenos, y no alterar los que pueden tener un peor rendimiento.

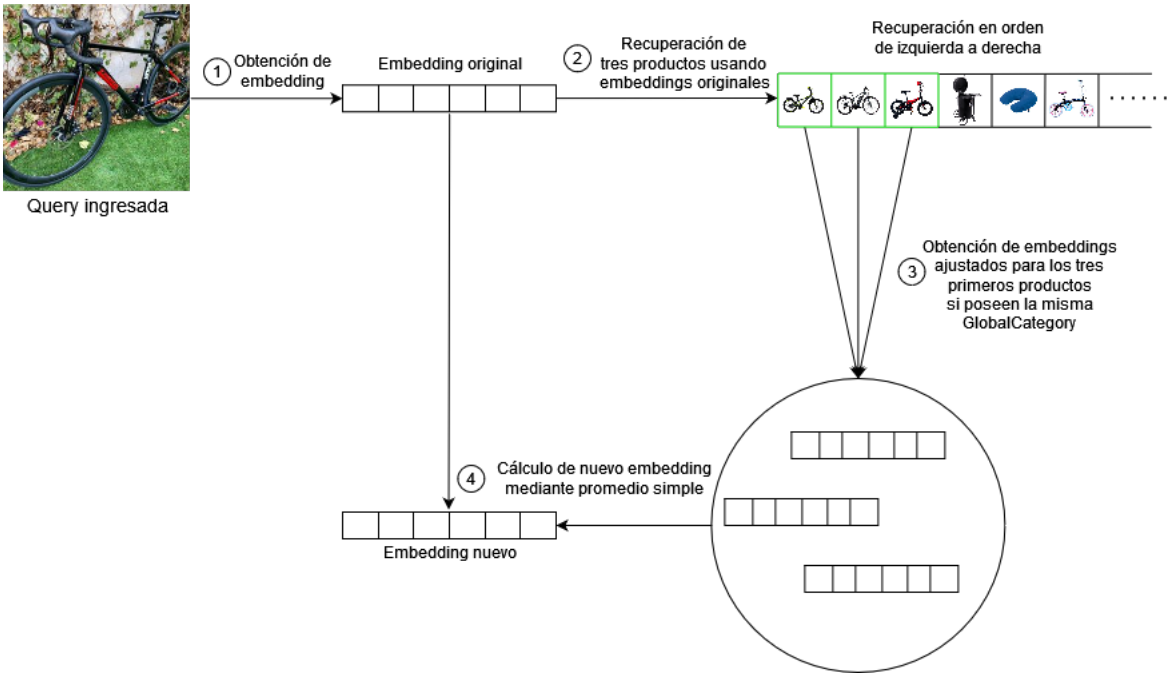


Figura 4.2: Algoritmo para adaptar *embedding* de *query* en tiempo real.

4.3. Entrenamiento y recuperación utilizando el modelo CLIP

En base a los resultados prometedores obtenidos por CLIP [1] se decide implementar una recuperación de imágenes utilizando el *encoder* visual de este modelo en vez del de ResNet-50. Debido a que CLIP busca generar un espacio de texto similar al espacio visual, se propone además usar el *encoder* de texto del mismo. De esta manera, se espera que los vectores de características de las descripciones se parezcan a los vectores de sus imágenes correspondientes.

Considerando los datos de cada catálogo, se decide además entrenar seis modelos (uno para cada catálogo) esperando un ajuste más adecuado de los textos e imágenes presentes en los distintos catálogos. La función de pérdida definida en CLIP, la cual llaman *symmetric loss function*, se define como:

$$\mathcal{L} = \frac{\textit{CrossEntropyLoss}(X, z) + \textit{CrossEntropyLoss}(Y, z)}{2} \tag{4.3}$$

en donde X es el valor real para el espacio visual, Y el valor real para el espacio de texto, y z un vector $z = [0, 1, ..., N-1]$, con N igual al tamaño del *batch* utilizado durante el entrenamiento.

5. Experimentos y resultados

5.1. Métricas utilizadas

5.1.1. *Mean Average Precision (mAP)*

La métrica mAP permite evaluar directamente la capacidad de un modelo para retornar productos relevantes en una recuperación semántica de imágenes. Todos los experimentos que consideran esta métrica dentro de su evaluación realizan un mAP@20, lo que significa que se toman en cuenta los primeros 20 productos recuperados. Para los *datasets* que contienen únicamente *queries* de catálogo, la evaluación del mAP@20 se hace sobre 100 *queries* previamente extraídas y separadas del catálogo. En cuanto a los *datasets* que también contienen *queries* reales de evaluación, además se obtienen los mAP@20 sobre estas, las cuales corresponden a 256 *queries* distintas para Homy, y 300 para Pepeganga.

5.1.2. *Adjusted Rand Index* y *Adjusted Mutual Information Score*

Las métricas de evaluación *Adjusted Rand Index* (ARI) y *Adjusted Mutual Information Score* (AMIS) son métricas utilizadas para evaluar *clusters*. En este caso, los experimentos que utilizan estas métricas buscan medir la capacidad representativa de los *embeddings* en 2 dimensiones, en donde un valor igual a 0 indica un etiquetado aleatorio en el proceso de *clustering*, y un valor igual a 1 representa un etiquetado perfecto.

5.2. Experimentos preliminares

5.2.1. Resultados y visualización de productos en 2 dimensiones

En estos experimentos se usa únicamente el *dataset* de Pepeganga, dado que son experimentos preliminares que permiten entender y aportar a la hipótesis de que las descripciones de los productos poseen una mayor semántica que las imágenes.

La técnica proporcionada por UMAP puede ser utilizada para la visualización de los datos mediante una reducción de *embeddings* a 2 dimensiones, permitiendo observar estos datos y su estructura para obtener intuición sobre estos en sí. Además, se puede medir la capacidad representativa de estos *embeddings* mediante la generación y evaluación de *clusters* en base a estos. Primero se realiza una visualización de los *embeddings* de texto generados por Word2Vec después de aplicar UMAP, definiendo los parámetros

$n\_neighbors = 15$ y $min\_dist = 0.1$, ya que son los parámetros que usa UMAP por defecto, con lo cual se obtiene el gráfico mostrado en la figura 5.1.

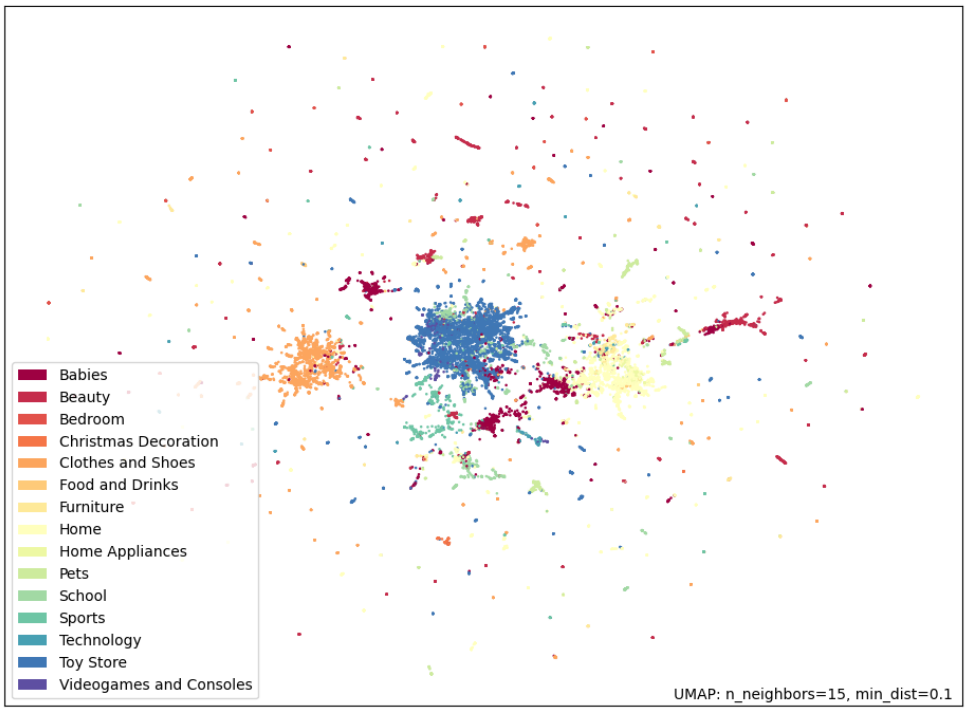


Figura 5.1: Visualización de *embeddings* de texto generados por Word2Vec.

Luego, se utilizan las métricas de evaluación de *clusters Adjusted Rand Index* (ARI) y *Adjusted Mutual Information Score* (AMIS) para medir la capacidad representativa de los *embeddings* en 2 dimensiones, con lo cual se obtienen los resultados observados en la tabla 5.1.

Tabla 5.1: Resultados de evaluación de *clusters* generados con *embeddings* de texto reducidos a 2 dimensiones utilizando UMAP.

Modelo	ARI	AMIS
FastText	**0.403**	0.420
GloVe	0.376	0.394
Word2Vec	0.398	**0.443**

Al igual que con los *embeddings* de texto, se decide realizar una visualización y evaluación de la capacidad representativa de los *embeddings* visuales mediante UMAP. Se utilizan los mismos parámetros de UMAP usados para la evaluación de los *embeddings*

de texto, con $n\_neighbors = 15$ y $min\_dist = 0.1$, obteniéndose los resultados observados en la figura 5.2

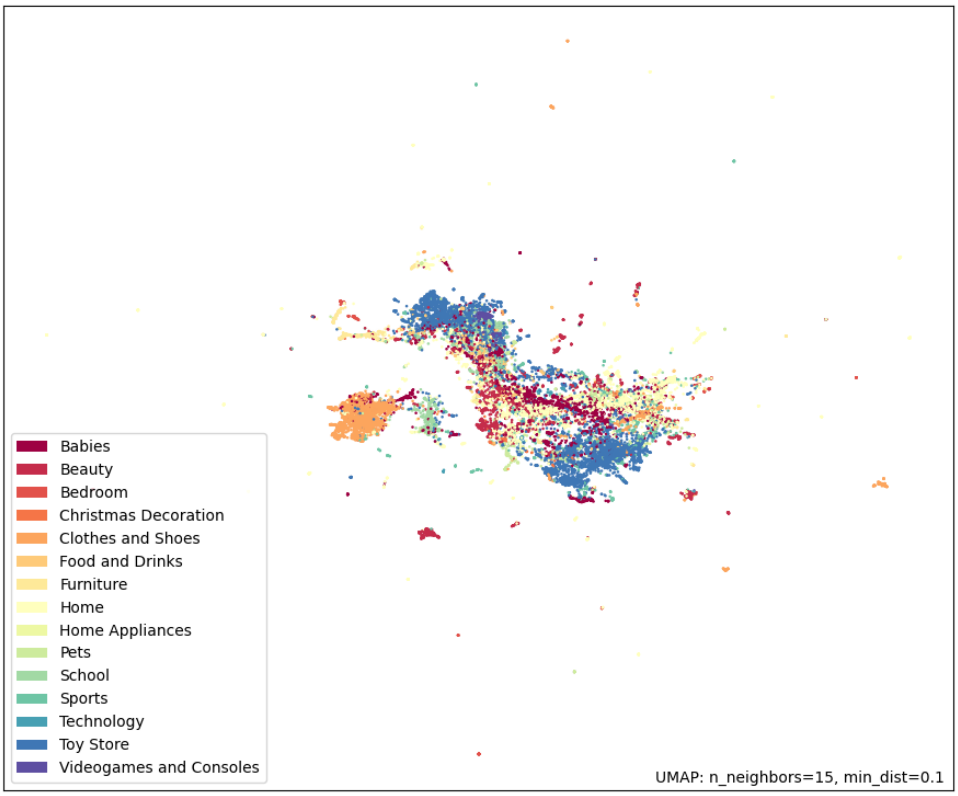


Figura 5.2: Visualización de *embeddings* visuales generados por una red ResNet-50.

Finalmente, se obtienen los resultados mostrados en la tabla 5.2 al aplicar las métricas de evaluación *Adjusted Rand Index* (ARI) y *Adjusted Mutual Information Score* (AMIS), los cuales son esperables considerando la drástica reducción de dimensionalidad a únicamente 2 dimensiones.

Tabla 5.2: Resultados de evaluación de *clusters* generados por *embeddings* visuales reducidos a 2 dimensiones utilizando UMAP.

Modelo	ARI	AMIS
ResNet-50	0.105	0.206

Se puede notar, tanto al observar las imágenes como viendo los resultados que entregan el *Adjusted Rand Index* y el *Adjusted Mutual Information Score*, como el espacio vectorial del texto se encuentra mejor definido que el espacio visual, formándose grupos

menos difusos para las categorías globales, e implicando que las representaciones de texto poseen una mayor semántica.

5.2.2. Recuperación de productos similares utilizando texto

Los experimentos de recuperación mediante texto buscan mostrar la diferencia semántica entre los espacios vectoriales: visual y textual. Para esto, se evalúa el mAP@20 de las descripciones de los 100 productos extraídos de cada catálogo para ser utilizados como *queries* sobre sus catálogos completos respectivos.

5.2.2.1. Pepeganga

Se genera en primera instancia una visualización con la recuperación de los 5 productos más similares utilizando únicamente las descripciones de 3 productos que se eligen al azar, las cuales se codifican usando el modelo GloVe.

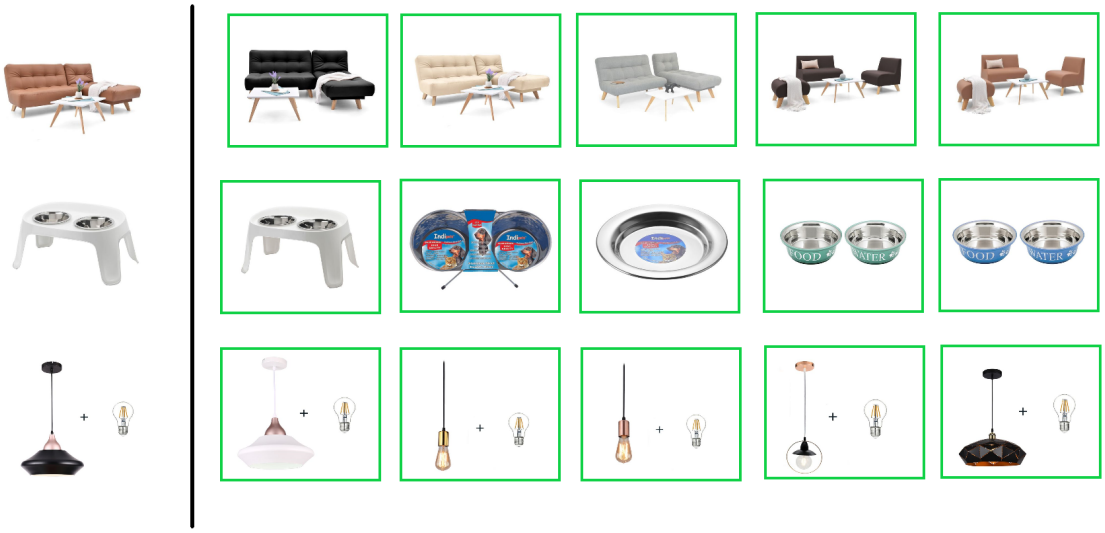


Figura 5.3: Productos con sus respectivos resultados más similares (Top 5) utilizando GloVe.

En la visualización representada en la Figura 5.3 los productos de la izquierda son los cuales se evalúa, y los de la derecha son los 5 más similares para cada producto evaluado respectivo. El cuadro verde que encierra los productos indica que las categorías globales de cada producto retornado son iguales a la categoría global de su consulta ingresada. Así, el resultado observado evidencia resultados positivos, en donde existe coherencia y una clara semántica presente en los textos.

Luego, se procede a evaluar los 100 productos de Pepeganga, con modelos más recientes para generar vectores de palabras, y comparándolos directamente con los resultados de Word2Vec. Además, para poder comparar cuantitativamente con un modelo visual, se añade el modelo *baseline* ResNet-50.

Tabla 5.3: Evaluación de la recuperación de productos sobre el catálogo de Pepeganga realizada con distintos modelos de texto y ResNet-50 base.

Modelo	Espacio	mAP (GC)	mAP (CT)
ResNet-50	Visual	87.13 %	67.83 %
Word2Vec	Textual	95.53 %	81.39 %
MPNet-base	Textual	**97.69 %**	**83.36 %**
RoBERTa-large	Textual	96.07 %	80.94 %
BERT-uncased	Textual	95.33 %	82.01 %

En la Tabla 5.3 se puede observar como todos los modelos de texto superan al modelo *baseline* tanto para el *GlobalCategory* como el *TreeCategory*. En particular, el modelo MPNet-base mejora los resultados en un **10.56 %** para el primero, y un **15.53 %** en el segundo.

5.2.2.2. UNIQLO

Para el resto de los catálogos, se realiza únicamente la evaluación de sus 100 *queries* respectivas. En particular, los resultados para UNIQLO observados en la Tabla 5.4 muestran que los modelos de texto no son capaces de superar el modelo *baseline* ResNet-50. Se cree que esto se debe a las descripciones similares que pueden tener prendas de ropa que a grandes rasgos son equivalentes, pero en que las clasificaciones son distintas. Por ejemplo, una camiseta de hombre y una camiseta de mujer pueden tener descripciones similares o iguales, sin embargo, poseen un *GlobalCategory* y un *CategoryTree* distintos.

Tabla 5.4: Evaluación de la recuperación de productos sobre el catálogo de UNIQLO realizada con distintos modelos de texto y ResNet-50 base.

Modelo	Espacio	mAP (GC)	mAP (CT)
ResNet-50	Visual	**70.66 %**	**42.16 %**
Word2Vec	Textual	58.02 %	39.07 %
MPNet-base	Textual	63.39 %	41.86 %
RoBERTa-large	Textual	61.01 %	40.88 %
BERT-uncased	Textual	57.62 %	32.56 %

5.2.2.3. Cartier

Los resultados de evaluar las *queries* del catálogo de Cartier se pueden apreciar en la Tabla 5.5. Al igual que en el catálogo de Pepeganga, todos los modelos de texto demuestran mejores resultados que el *baseline*. En específico, el modelo MPNet-base demuestra una mejora de un **9.43 %** sobre el *GlobalCategory*, mientras que el modelo RoBERTa-large aumenta el mAP@20 en un **15.20 %**.

Tabla 5.5: Evaluación de la recuperación de productos sobre el catálogo de Cartier realizada con distintos modelos de texto y ResNet-50 base.

Modelo	Espacio	mAP (GC)	mAP (CT)
ResNet-50	Visual	81.58 %	58.57 %
Word2Vec	Textual	90.87 %	61.89 %
MPNet-base	Textual	**91.01 %**	72.93 %
RoBERTa-large	Textual	90.77 %	**73.77 %**
BERT-uncased	Textual	90.91 %	68.67 %

5.2.2.4. IKEA

Nuevamente, como se puede ver en la Tabla 5.6, los resultados que se obtienen para IKEA con los modelos de texto son significativamente mejores. RoBERTa-large ofrece una mejora de un **14.31 %** para *GlobalCategory* y de un **6.86 %** para *CategoryTree*.

Tabla 5.6: Evaluación de la recuperación de productos sobre el catálogo de IKEA realizada con distintos modelos de texto y ResNet-50 base.

Modelo	Espacio	mAP (GC)	mAP (CT)
ResNet-50	Visual	71.36 %	50.50 %
Word2Vec	Textual	79.08 %	53.64 %
MPNet-base	Textual	81.33 %	55.26 %
RoBERTa-large	Textual	**85.67 %**	**57.36 %**
BERT-uncased	Textual	78.81 %	51.81 %

5.2.2.5. World Market

Al observar la tabla 5.7, se puede notar que los modelos de texto generan una recuperación notablemente mejor para el catálogo de *World Market*. Una vez más, MPNet-base y RoBERTa-large son los que obtienen mejores resultados, el primero incrementando el mAP@20 en un **17.42 %** para *GlobalCategory*, mientras que el segundo aumenta un **22.56 %** para *CategoryTree*.

Tabla 5.7: Evaluación de la recuperación de productos sobre el catálogo de *World Market* realizada con distintos modelos de texto y ResNet-50 base.

Modelo	Espacio	mAP (GC)	mAP (CT)
ResNet-50	Visual	68.37 %	48.48 %
Word2Vec	Textual	83.39 %	62.47 %
MPNet-base	Textual	**85.79 %**	68.46 %
RoBERTa-large	Textual	85.43 %	**71.04 %**
BERT-uncased	Textual	81.64 %	58.62 %

5.2.2.6. Homy

Por último, la tabla 5.8 muestra los resultados obtenidos para el catálogo de Homy. En este caso, si bien todos los modelos logran obtener mejores resultados para *GlobalCategory*, esta mejora no alcanza el 2 %, a excepción de MPNet-base, el cual logra obtener un **3.86 %** más. Sin embargo, ningún modelo de texto puede superar el valor de mAP@20 que obtiene el *baseline* para *CategoryTree*, lo cual puede deberse a que las descripciones de los productos no son suficientemente detalladas como para discernir entre productos que se encuentran categorizados dentro del mismo *GlobalCategory*.

Tabla 5.8: Evaluación de la recuperación de productos sobre el catálogo de Homy realizada con distintos modelos de texto y ResNet-50 base.

Modelo	Espacio	mAP (GC)	mAP (CT)
ResNet-50	Visual	83.56 %	**65.49 %**
Word2Vec	Textual	84.27 %	58.81 %
MPNet-base	Textual	**87.42 %**	62.65 %
RoBERTa-large	Textual	84.87 %	62.26 %
BERT-uncased	Textual	84.07 %	59.68 %

5.3. Experimentos de métodos propuestos

5.3.1. Recuperación de imágenes utilizando ResNet-50 con ajuste de *embeddings* visuales y *query* adaptativa

Para acotar la cantidad de experimentos realizados, se definen diferentes parámetros a usarse para los distintos métodos que promedian *embeddings* al momento de realizar el algoritmo de ajuste. En particular, se tiene que:

- **Promedio simple con moving average (mean)**: $k = [3, 5, 7, 10]$ y $\alpha = [0.1, 0.2, ..., 0.9]$.
- **Promedio ponderado con similitud coseno (sim)**: $k = [3, 4, 5, 6, 7, 8, 9, 10, 15]$.
- **Promedio ponderado con *softmax* (softmax)**: $k = [3, 5, 10]$ y $T = [0.5, 1.0, 2.0, 3.0]$.

donde k representa la cantidad de vecinos más cercanos en base al *embedding* de texto de un producto, α es el parámetro asociado al *moving average*, y T es el parámetro asociado a la función *softmax*. Además, para cada configuración posible, se calcula el mAP@20 tanto sin usar *query* adaptativa como usándola.

En las tablas siguientes, dentro del campo “Configuración de ajuste”, se define como “base” (*baseline*) al modelo ResNet-50 sin ajustar los *embeddings* del catálogo. Los algoritmos de ajuste de *embeddings* mediante promedio simple con moving average, promedio ponderado con similitud coseno, y promedio ponderado con *softmax* poseen el nombre de método simplificado “mean”, “sim” y “softmax” respectivamente. En adición, se agregan únicamente los modelos con los mejores resultados tanto en *GlobalCategory*

como en *CategoryTree*, lo cual a veces corresponde a un solo modelo. Esto se hace tanto para los casos con *query* adaptativa y los casos sin esta, lo que significa que al momento de ingresar una imagen al modelo, su *embedding* visual puede ser ajustado mediante el algoritmo ya descrito. Se incorpora también el resultado para el modelo "base" (*baseline*) con *query* adaptativa, que corresponde al modelo ResNet-50 entrenado en *ImageNet* y con la capacidad de adaptar la *query* ingresada.

Cabe destacar que el modelo de texto que se utiliza para realizar el ajuste de *embeddings* del catálogo y la adaptación de la *query* es RoBERTa-large. Si bien se realizan experimentos para todas las configuraciones de ajuste con todos los modelos de texto (RoBERTa-large, MPNet-base, BERT, Word2Vec), antes de incorporar la posibilidad de adaptar la *query* ingresada en tiempo real, se opta por acotar la cantidad de experimentos mediante la elección del mejor modelo de texto. Esta elección se lleva acabo de la siguiente manera:

1. Para cada catálogo y modelo de texto, se obtienen dos valores, correspondientes al promedio de mAP@20 para *GlobalCategory*, y el promedio de mAP@20 para *CategoryTree*, utilizando todas las posibles configuraciones para ajustar *embeddings*.
2. Luego se genera un único número para cada catálogo y modelo mediante el promedio de los dos valores mencionados anteriormente.
3. Finalmente, se genera un valor promedio para cada modelo de texto utilizando todos los catálogos, en donde el valor más grande corresponde al mejor modelo.

De este modo, RoBERTa es el modelo con mejores resultados, sin embargo, MPNet-base obtiene resultados altamente cercanos. Luego, se procede a realizar la evaluación sobre los *datasets*.

5.3.1.1. UNIQLO

A pesar de los resultados que se obtienen al realizar una recuperación utilizando únicamente texto en el catálogo de UNIQLO, la Tabla 5.9 permite observar mejoras, tanto al realizar un ajuste de *embeddings* como al adaptar la *query* ingresada en tiempo real. En particular, el solo hecho de adaptar la *query* sin ajustar los *embeddings* ya genera una mejora en el mAP@20 de un **3 %** para *GlobalCategory*, manteniendo igual el *CategoryTree*.

Además, la configuración con mejor mAP@20 para *GlobalCategory* mejora los resultados en un **4.12 %**. Sin embargo, para *CategoryTree* usando la misma configuración, el mAP@20 disminuye en un 0.67 %. Por otro lado, la configuración con mejor mAP@20 para *CategoryTree* aumenta en un **1.52 %** el mAP@20 para *GlobalCategory*, y en un 1.47 % para *CategoryTree*.

Tabla 5.9: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de UNIQLO, agregando ajuste de *embeddings* y *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
base	No	70.66 %	42.16 %
base	Sí	73.66 %	42.16 %
$k = 7$, $\alpha = 0.3$, method = mean	No	73.66 %	42.84 %
$k = 5$, $\alpha = 0.4$, method = mean	No	72.18 %	**43.63 %**
$k = 7$, $\alpha = 0.4$, method = mean	Sí	**74.78 %**	41.49 %
$k = 3$, $\alpha = 0.9$, method = mean	Sí	71.52 %	43.37 %

Los resultados de recuperación de imágenes para un producto aleatorio del *dataset*, utilizando las dos mejores configuraciones, se pueden observar en las Figuras 5.4 y 5.5, en donde el producto ingresado como *query* se encuentra encerrado dentro de un cuadrado azulado, y el orden de los productos retornados es de izquierda a derecha, y de arriba hacia abajo. Sin embargo, dada la naturaleza del *dataset*, cuesta trabajo evaluar cualitativamente el comportamiento de los modelos.

Baseline

ResNet-50 con query adaptativa
($k = 7$, $\alpha = 0.4$, method = mean)

Figura 5.4: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para ResNet-50 respecto al *GlobalCategory* dentro del catálogo de UNIQLO. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

ResNet-50 sin query adaptativa
($k = 5$, $\alpha = 0.4$, method = mean)

Figura 5.5: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para ResNet-50 respecto al *CategoryTree* dentro del catálogo de UNIQLO. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

5.3.1.2. Cartier

Los resultados visibles en la Tabla 5.10 que se obtienen en el catálogo de Cartier muestran una clara mejora al ajustar los *embeddings* y al adaptar la *query*. El modelo con mejor mAP@20 para *GlobalCategory* lo mejora el mAP@20 en un **3.82 %**, sin embargo, el mAP@20 para *CategoryTree* empeora significativamente usando esta configuración, disminuyendo en un 5.12 %. A pesar de esto, la configuración con mejor mAP@20 para la categorización de grano fino obtiene una mejora de un **5.53 %** en *CategoryTree* y de un **2.41 %** en *GlobalCategory*.

Tabla 5.10: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Cartier, agregando ajuste de *embeddings* y *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
base	No	81.58 %	58.57 %
base	Sí	82.28 %	58.54 %
$k = 15$, method = sim	No	**85.40 %**	53.45 %
$k = 7$, $\alpha = 0.6$, method = mean	No	83.98 %	62.42 %
$k = 3$, $\alpha = 0.1$, method = mean	Sí	85.22 %	62.57 %
$k = 5$, $\alpha = 0.4$, method = mean	Sí	83.99 %	**64.10 %**

Observando las recuperaciones para un producto aleatorio de las Figuras 5.6 y 5.7, correspondiente a unos aros de diamante, usando las dos mejores configuraciones, se puede apreciar inmediatamente como los modelos ajustados hacen desaparecer a los relojes y anillos de los primeros productos retornados, y además, como aparecen nuevos aros, demostrando una mejora en la semántica de la recuperación.

Baseline

ResNet-50 sin query adaptativa
($k = 15$, method = sim)

Figura 5.6: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para ResNet-50 respecto al *GlobalCategory* dentro del catálogo de Cartier. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

ResNet-50 con query adaptativa
($k = 5$, $k = 0.4$, method = mean)

Figura 5.7: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para ResNet-50 respecto al *CategoryTree* dentro del catálogo de Cartier. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

5.3.1.3. IKEA

Para el catálogo de IKEA, los resultados de la Tabla 5.11 dejan en evidencia que el adaptar la *query* en tiempo real sin realizar un ajuste previo de los *embeddings* no permite generar una mejor recuperación. A pesar de esto, las dos configuraciones con mejor mAP@20 tanto para *GlobalCategory* y *CategoryTree* presentan mejoras de mAP@20 en ambas categorizaciones, en donde la primera configuración sí adapta la *query*, mientras que la segunda no.

Tabla 5.11: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de IKEA, agregando ajuste de *embeddings* y *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
base	No	71.36 %	50.50 %
base	Sí	71.07 %	50.40 %
$k = 5$, $\alpha = 0.6$, method = mean	No	74.10 %	50.77 %
$k = 7$, $\alpha = 0.7$, method = mean	No	74.00 %	**53.73 %**
$k = 5$, $\alpha = 0.6$, method = mean	Sí	**75.13 %**	50.73 %
$k = 7$, $\alpha = 0.7$, method = mean	Sí	74.52 %	53.64 %

En las Figuras 5.8 y 5.9 se presentan las diferencias en la recuperación entre el modelo

baseline y las dos mejores configuraciones. El *baseline* tiende a retornar sillas luego de los primeros 3 productos, mientras que la mejor configuración para *GlobalCategory* logra mejorar dicha recuperación añadiendo más faroles cerca del inicio. Sin embargo, la mejor configuración para *CategoryTree* no presenta mejoras cualitativas de manera evidente.

Baseline

ResNet-50 con query adaptativa
($k = 5$, $\alpha = 0.6$, method = mean)

Figura 5.8: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para ResNet-50 respecto al *GlobalCategory* dentro del catálogo de IKEA. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

ResNet-50 sin query adaptativa
($k = 7$, $\alpha = 0.7$, method = mean)

Figura 5.9: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para ResNet-50 respecto al *CategoryTree* dentro del catálogo de IKEA. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

5.3.1.4. World Market

Observando la Tabla 5.12, nuevamente la incorporación tanto de ajuste de *embeddings* visuales como la adaptación de la *query* mejoran la recuperación significativamente, obteniéndose un incremento de mAP@20 de un **6.74 %** para *GlobalCategory* y de un **8.51 %** *CategoryTree* con las mejores configuraciones para cada categorización. Además, las dos configuraciones mejoran los resultados respecto al *baseline* para ambas categorizaciones.

Tabla 5.12: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de *World Market*, agregando ajuste de *embeddings* y *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
base	No	68.37 %	48.48 %
base	Sí	68.76 %	46.96 %
$k = 7$, method = sim	No	74.67 %	53.19 %
$k = 3$, $\alpha = 0.6$, method = mean	No	71.81 %	**56.99 %**
$k = 7$, method = sim	Sí	**75.11 %**	51.66 %
$k = 3$, $\alpha = 0.6$, method = mean	Sí	71.57 %	53.50 %

De las recuperaciones de imágenes observadas en las Figuras 5.10 y 5.11 se puede evidenciar que ambos modelos ajustados son capaces de incorporar más lámparas den-

tro de los primeros resultados en comparación al *baseline*, con una superioridad por parte de la mejor configuración de ajuste para *GlobalCategory*.

Baseline

ResNet-50 con query adaptativa
($k = 7$, method = sim)

Figura 5.10: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para ResNet-50 respecto al *GlobalCategory* dentro del catálogo de *World Market*. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

ResNet-50 sin query adaptativa
($k = 3$, $\alpha = 0.6$, method = mean)

Figura 5.11: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para ResNet-50 respecto al *CategoryTree* dentro del catálogo de *World Market*. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

5.3.1.5. Homy

Al igual que en el caso anterior, la Tabla 5.13 muestra que existen configuraciones que mejoran la recuperación utilizando *queries* de catálogo al ser evaluada tanto en grano fino como grano grueso, obteniéndose una mejora de mAP@20 de hasta un **2.29 %** en *GlobalCategory*, y de un **3.69 %** para *CategoryTree*.

Tabla 5.13: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Homy, agregando ajuste de *embeddings* y *query* adaptativa.

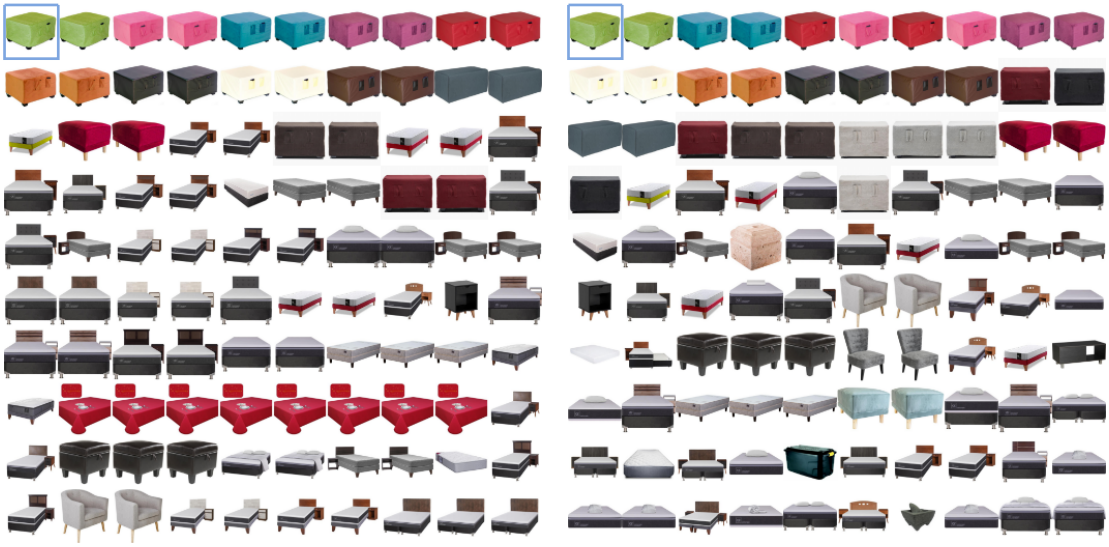
Configuración de ajuste	**Query adaptativa**	**mAP (GC)**	**mAP (CT)**
base	No	83.56 %	65.49 %
base	Sí	84.19 %	65.82 %
$k = 10$, $T = 0.5$, method = softmax	No	85.38 %	63.21 %
$k = 3$, $\alpha = 0.6$, method = mean	No	84.64 %	**69.18 %**
$k = 5$, method = sim	Sí	**85.85 %**	66.23 %
$k = 3$, $\alpha = 0.6$, method = mean	Sí	85.10 %	68.07 %

Las recuperaciones de las Figuras 5.12 y 5.13 para un *puff* perteneciente al *dataset* de Homy muestran que el *baseline* funciona bien en un inicio, pero que desde cierto punto empieza a retornar camas. Este comportamiento también se observa en los modelos ajustados, sin embargo, estos modelos tienden a retornar camas en posiciones menos significativas.

Baseline

ResNet-50 con query adaptativa
($k = 5$, method = sim)

Figura 5.12: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para ResNet-50 respecto al *GlobalCategory* dentro del catálogo de Homy. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.



Baseline

ResNet-50 sin query adaptativa
($k = 3$, $\alpha = 0.6$, method = mean)

Figura 5.13: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para ResNet-50 respecto al *CategoryTree* dentro del catálogo de Homy. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

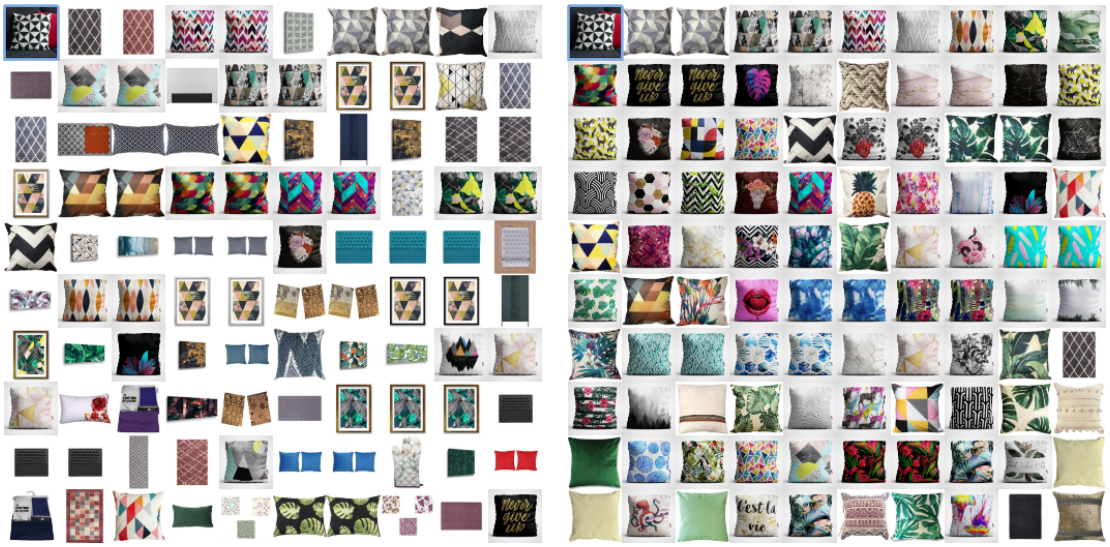
También se hace una evaluación al utilizar 256 *queries* de evaluación reales para comprobar si los métodos que se proponen funcionan en casos de uso cotidianos. Los resul-

tados que se observan en la Tabla 5.14 demuestran una mejora significativa, al menos en grano grueso, obteniéndose una mejora en el mAP@20 de hasta un 4.33 %.

Tabla 5.14: Evaluación de la recuperación de imágenes usando *queries* reales sobre el *dataset* de Homy, agregando ajuste de *embeddings* y *query* adaptativa.

Configuración de ajuste	**Query adaptativa**	**mAP (GC)**
base	No	62.83 %
base	Sí	63.13 %
$k = 7$, $\alpha = 0.9$, method = mean	No	62.73 %
$k = 5$, $T = 3.0$, method = softmax	Sí	**67.16 %**

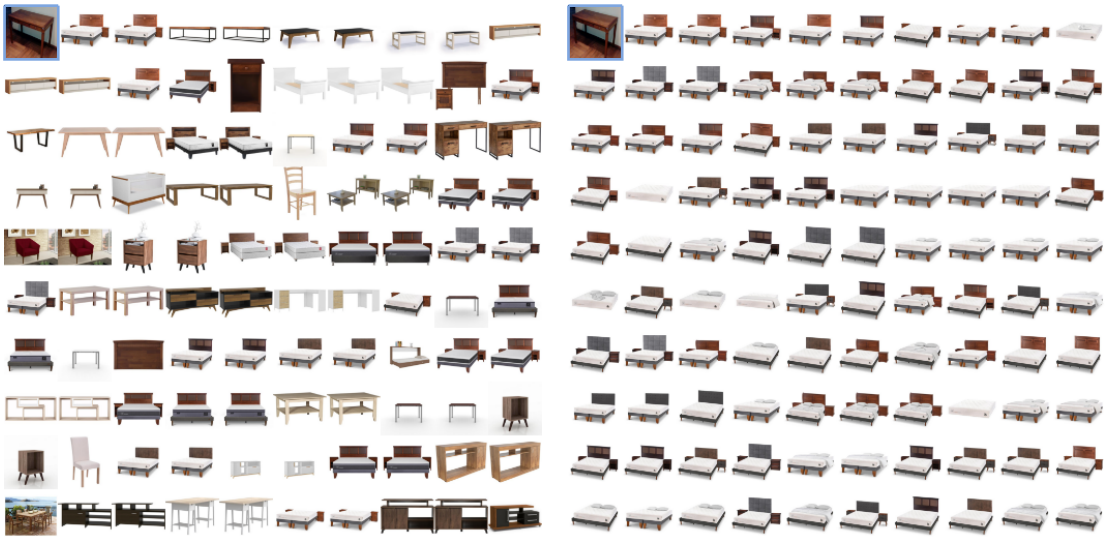
Además, como se puede observar en la Figura 5.14, la recuperación de imágenes para una foto real de un cojín mejora considerablemente. Sin embargo, para el segundo caso de uso real, correspondiente a un escritorio, los resultados parecen empeorar.



Baseline

ResNet-50 con query adaptativa
($k = 5$, $T = 3.0$, method = softmax)

Figura 5.14: Ejemplo de recuperación de una foto real de un cojín para el *baseline* y el modelo con mejor ajuste para ResNet-50 dentro del catálogo de Homy. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.



Baseline

ResNet-50 con query adaptativa
($k = 5, T = 3.0$, method = softmax)

Figura 5.15: Ejemplo de recuperación de una foto real de un mueble para el *baseline* y el modelo con mejor ajuste para ResNet-50 dentro del catálogo de Homy. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

5.3.1.6. Pepeganga

Nuevamente, al evaluar las 100 *queries* de catálogo sobre el *dataset* de Pepeganga, los algoritmos para ajustar *embeddings* visuales y el algoritmo de adaptación de *query* en tiempo real logran mejorar el *baseline* para ambas categorizaciones, lo cual se puede observar en la tabla 5.15.

Tabla 5.15: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Pepeganga, agregando ajuste de *embeddings* y *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
base	No	87.13 %	67.83 %
base	Sí	88.05 %	68.78 %
$k = 3$, $T = 0.5$, method = softmax	No	91.60 %	71.87 %
$k = 10$, $\alpha = 0.4$, method = mean	No	91.26 %	73.74 %
$k = 3$, $T = 0.5$, method = softmax	Sí	**92.75 %**	**75.02 %**

Utilizando la mejor configuración de ajuste para el catálogo de Pepeganga, se genera una recuperación para una mando de consola con un juego de fondo. En la Figura 5.16 se puede observar que si bien la recuperación no mejora notablemente, sí existe un leve aumento en la semántica al retornar una mayor cantidad de mandos.

Baseline

ResNet-50 con query adaptativa
($k = 3, T = 0.5$, method = softmax)

Figura 5.16: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para ResNet-50 respecto al *GlobalCategory* y *CategoryTree* dentro del catálogo de Pepeganga. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Por último, se evalúa la recuperación utilizando las 300 *queries* reales disponibles para el catálogo de Pepeganga, obteniéndose los resultados de la Tabla 5.16, los cuales, una vez más, demuestran una mejora en casos de uso reales para recuperación de imágenes al realizar una evaluación de grano grueso.

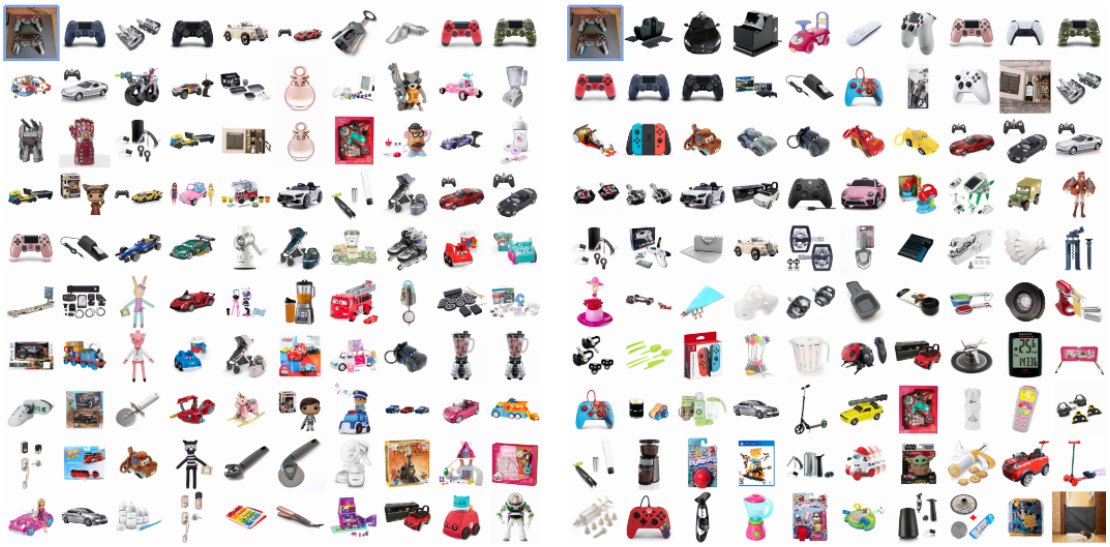
Tabla 5.16: Evaluación de la recuperación de imágenes usando *queries* reales sobre el *dataset* de Pepeganga, agregando ajuste de *embeddings* y *query* adaptativa.

Configuración de ajuste	**Query adaptativa**	**mAP (GC)**
base	No	57.21 %
base	Sí	58.56 %
$k = 10$, $\alpha = 0.4$, method = mean	No	59.82 %
$k = 10$, $T = 0.5$, method = softmax	Sí	**60.78 %**

En la Figura 5.17 se muestra una recuperación para una foto real de una consola con dos mandos encima de ella. El *baseline* retorna algunos controles dentro de los 9 primeros productos, en donde dos de ellos se encuentran en las 3 primeras posiciones. El mejor modelo ajustado no retorna mandos hasta la posición 6, sin embargo, la cantidad de mandos que retorna dentro de los 99 productos mostrados es significativamente mayor a la del *baseline*.

En cuanto a la Figura 5.18, la recuperación se hace para la foto de un árbol de navidad.

Cabe destacar que en el *dataset* de Pepeganga no existen árboles de navidad, pero sí existen otro tipo de productos navideños. Los resultados de la recuperación para ambos modelos resultan ser notablemente malos.



Baseline

ResNet-50 con query adaptativa
($k = 10$, $T = 0.5$, method = softmax)

Figura 5.17: Ejemplo de recuperación de una foto real de una consola con controles para el *baseline* y el modelo con mejor ajuste para ResNet-50 dentro del catálogo de Pepeganga. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

ResNet-50 con query adaptativa
($k = 10$, $T = 0.5$, method = softmax)

Figura 5.18: Ejemplo de recuperación de una foto real de un árbol de navidad para el *baseline* y el modelo con mejor ajuste para ResNet-50 dentro del catálogo de Pepeganga. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

5.3.2. Recuperación de imágenes utilizando CLIP con ajuste de *embeddings* visuales y *query* adaptativa

En estos últimos experimentos se evalúa la capacidad de CLIP para aumentar la semántica en la recuperación de imágenes, y medir cómo se comporta este modelo cuando se ajustan los *embeddings* visuales con los diferentes algoritmos implementados, además de utilizar el algoritmo de *query* adaptativa. Para llevar a cabo esta tarea, se utilizan los *embeddings* visuales que genera el *encoder* visual de CLIP, y en los casos en que se realice un ajuste de *embeddings* visuales dentro de cada catálogo, se usa además el *encoder* de texto de este mismo modelo.

Los experimentos de esta sección se encuentran divididos en dos partes: la primera consiste en evaluar el modelo CLIP-base, con los pesos disponibles públicamente luego de realizar el pre-entrenamiento, es decir, sin entrenar para alguna tarea específica. En la segunda parte se realiza la misma evaluación, pero esta vez se entrena, a partir del modelo pre-entrenado, un modelo distinto para cada catálogo. Este entrenamiento busca ajustar los *embeddings* generados por los *encoders* visual y de texto para cada catálogo, esperando obtener mejores resultados. El entrenamiento se lleva a cabo con diferentes configuraciones, sin embargo, la configuración de a continuación es la que presenta una mejor pérdida en el conjunto de prueba, y por ende, la que se usa para realizar los experimentos:

- Se entrena durante 20 épocas con un conjunto de entrenamiento igual al 80 % de los datos de un catálogo, y con un tamaño de *batch* de 256.
- El conjunto de prueba corresponde al 20 % de los datos de un catálogo.

- El conjunto de validación corresponde a las 100 *queries* que se extraen previamente, en conjunto a las *queries* reales para los casos que corresponda.
- Se utiliza la función de pérdida *symmetric loss*.
- Se utiliza el optimizador Adam, con un *learning rate* igual a $lr = 5e - 6$, y los parámetros $\beta_1 = 0.9$, $\beta_2 = 0.98$, $eps = 1e - 6$ y $weight\_decay = 0.2$.

Finalmente, durante el entrenamiento se guardan los mejores modelos para cada catálogo en base a la pérdida más baja que se obtiene con los datos de prueba.

Las tablas que se presentan a continuación contienen los mejores resultados de mAP@20 que se obtienen en la recuperación, para *GlobalCategory* y *CategoryTree*, añadiendo las opciones de ajustar *embeddings* visuales y adaptar la *query*. Además, se agrega el mAP@20 que obtiene el *baseline* ResNet-50 para una mejor comparación. Para estas tablas, la configuración de ajuste "base" corresponde al modelo CLIP sin ajuste de *embeddings*.

5.3.2.1. UNIQLO

Observando la Tabla 5.17 es evidente la amplia mejora que se obtiene al momento de utilizar el *encoder* visual de CLIP, sin realizar ningún tipo de ajuste de *embeddings* visuales ni *query* adaptativa, aumentando el mAP@20 para *GlobalCategory* en un **11.57 %** al compararlo con el *baseline*. Además, al introducir los algoritmos diseñados para el ajuste de *embeddings* esta mejora aumenta en un **12.52 %**. Sin embargo, el mAP@20 para *CategoryTree* solo incrementa en un **3.57 %** respecto al *baseline*.

Tabla 5.17: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de UNIQLO, utilizando los *embeddings* generados por CLIP sin entrenar, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
baseline (ResNet-50)	No	70.66 %	42.16 %
base	No	82.23 %	44.02 %
base	Sí	82.75 %	42.76 %
$k = 7$, $\alpha = 0.9$, method = mean	No	82.58 %	45.06 %
$k = 5$, $\alpha = 0.9$, method = mean	No	82.47 %	**45.73 %**
$k = 10$, $\alpha = 0.9$, method = mean	Sí	**83.18 %**	42.70 %
$k = 10$, $\alpha = 0.4$, method = mean	Sí	81.03 %	44.71 %

Nuevamente, observando las recuperaciones de las Figuras 5.19 y 5.20 para el mismo producto de UNIQLO que se utiliza anteriormente, debido a la naturaleza del catálogo.

Baseline

CLIP-base con query adaptativa
($k = 10$, $\alpha = 0.9$, method = mean)

Figura 5.19: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP-base respecto al *GlobalCategory* dentro del catálogo de UNIQLO. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

CLIP-base sin query adaptativa
($k = 5$, $\alpha = 0.9$, method = mean)

Figura 5.20: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP-base respecto al *CategoryTree* dentro del catálogo de UNIQLO. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Luego de realizar el entrenamiento utilizando los datos del catálogo de UNIQLO, la Tabla 5.18 muestra que la mejora en el mAP@20 de la mejor configuración se reduce

a un **11.13 %** para *GlobalCategory*, pero respecto a *CategoryTree*, este aumenta en un **6.22 %** al compararlo con el *baseline*.

Tabla 5.18: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de UNIQLO, utilizando los *embeddings* generados por CLIP entrenado, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
baseline (ResNet-50)	No	70.66 %	42.16 %
base	No	79.11 %	44.02 %
base	Sí	79.91 %	43.58 %
$k = 3$, $\alpha = 0.7$, method = mean	No	81.15 %	**48.38 %**
$k = 3$, $\alpha = 0.7$, method = mean	Sí	**81.79 %**	45.73 %
$k = 10$, $\alpha = 0.6$, method = mean	Sí	81.09 %	45.86 %

Las Figuras 5.21 y 5.22 muestran la recuperación de imágenes que obtienen estos modelos.

Baseline

CLIP-trained con query adaptativa
($k = 3$, $\alpha = 0.7$, method = mean)

Figura 5.21: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP entrenado respecto al *GlobalCategory* dentro del catálogo de UNIQLO. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

CLIP-base sin query adaptativa
($k = 5$, $\alpha = 0.9$, method = mean)

Figura 5.22: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP entrenado respecto al *CategoryTree* dentro del catálogo de UNIQLO. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

5.3.2.2. Cartier

Los resultados de recuperación sobre el catálogo de Cartier que se presentan en la Tabla 5.19 muestran una mejora del mAP@20 de un **2.03 %** para *GlobalCategory* usando la mejor configuración de ajuste de *embeddings*. Sin embargo, en este caso, el mAP@20 para *CategoryTree* disminuye en un 2.88 %. Ninguna configuración de ajuste mejora el resultado para *CategoryTree* que se obtiene al utilizar el *baseline*.

Tabla 5.19: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Cartier, utilizando los *embeddings* generados por CLIP sin entrenar, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
baseline (ResNet-50)	No	81.58 %	**58.57 %**
base	No	80.60 %	57.09 %
base	Sí	80.45 %	54.98 %
$k = 7$, $\alpha = 0.7$, method = mean	No	82.04 %	56.95 %
$k = 5$, $\alpha = 0.9$, method = mean	No	80.96 %	58.18 %
$k = 3$, $\alpha = 0.2$, method = mean	Sí	**83.61 %**	55.69 %
$k = 5$, $T = 3.0$, method = softmax	Sí	83.47 %	57.76 %

La recuperación de la Figura 5.23 demuestra una mejora evidente al utilizar el mejor modelo respecto al *GlobalCategory*, eliminando los relojes y retornando más aros. Sin embargo, la configuración de ajuste que se utiliza en la recuperación de la Figura 5.24 tiende a retornar una gran cantidad de collares, a pesar de que los primeros productos parecen ser más relevantes que los del *baseline*.

Baseline

CLIP-base con query adaptativa
($k = 3$, $\alpha = 0.2$, method = mean)

Figura 5.23: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP-base respecto al *GlobalCategory* dentro del catálogo de Cartier. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

CLIP-base sin query adaptativa
($k = 5$, $\alpha = 0.9$, method = mean)

Figura 5.24: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP-base respecto al *CategoryTree* dentro del catálogo de Cartier. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

En cuanto a la evaluación del modelo entrenado, se aumenta el mAP@20 para *GlobalCategory* hasta un **2.68 %** con la mejor configuración, sin embargo, nuevamente, no se puede superar el *baseline* para *CategoryTree*, siendo el mejor resultado incluso menor que el que se obtiene usando CLIP sin entrenamiento sobre el catálogo.

Tabla 5.20: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Cartier, utilizando los *embeddings* generados por CLIP entrenado, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
baseline (ResNet-50)	No	81.58 %	**58.57 %**
base	No	81.43 %	54.24 %
base	Sí	82.48 %	53.52 %
$k = 3$, $T = 3.0$, method = softmax	No	84.08 %	53.10 %
$k = 10$, $\alpha = 0.7$, method = mean	No	82.32 %	56.56 %
$k = 3$, $T = 3.0$, method = softmax	Sí	**84.26 %**	52.90 %
$k = 10$, $\alpha = 0.2$, method = mean	Sí	82.91 %	55.69 %

Nuevamente existe una mejora considerable en la recuperación de la Figura 5.25, pero gran parte de los productos retornados que se observan en la Figura 5.26 no parecen ser relevantes.

Baseline

CLIP-trained con query adaptativa
($k = 3, T = 3.0$, method = softmax)

Figura 5.25: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP entrenado respecto al *GlobalCategory* dentro del catálogo de Cartier. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

CLIP-trained sin query adaptativa
($k = 10, \alpha = 0.7$, method = mean)

Figura 5.26: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP entrenado respecto al *CategoryTree* dentro del catálogo de Cartier. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Se cree que estos resultados se deben a las descripciones ambiguas de los productos de Cartier, en donde no se presenta suficiente información para que el *encoder* de texto

de CLIP asocie la descripción de un producto con su respectiva imagen.

5.3.2.3. IKEA

Con el modelo sin entrenamiento sobre el catálogo de IKEA, los resultados de la Tabla 5.21 demuestran una mejora sustancial sobre las *queries* de catálogo al ajustar los *embeddings* visuales y adaptar la *query* en tiempo real, aumentando el mAP@20 de *GlobalCategory* en un **9.2 %** en comparación al *baseline*, y para *CategoryTree* en un **5.28 %** para las mejores configuraciones respectivas.

Tabla 5.21: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de IKEA, utilizando los *embeddings* generados por CLIP sin entrenar, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

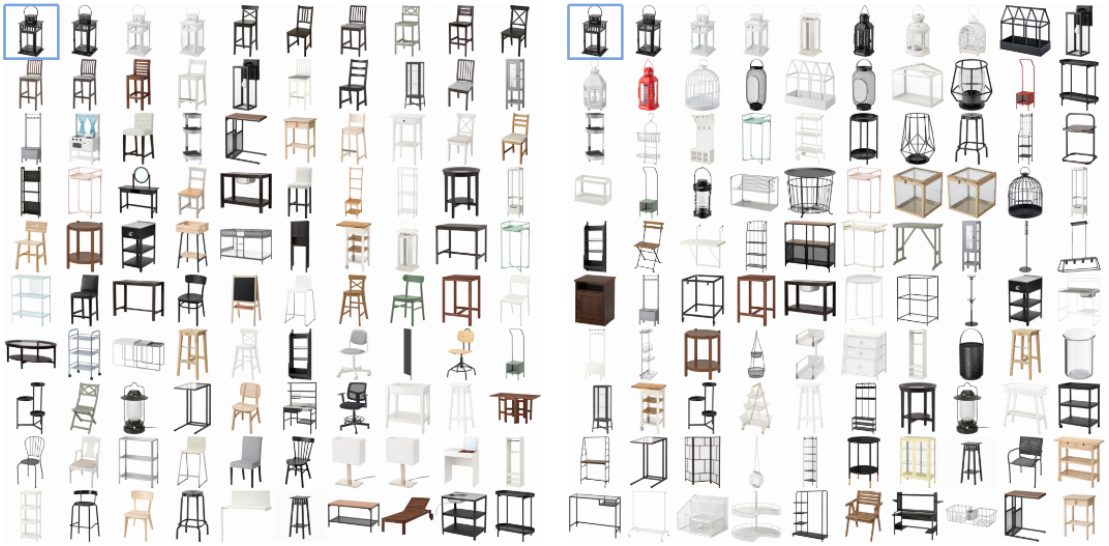
Configuración de ajuste	**Query adaptativa**	**mAP (GC)**	**mAP (CT)**
baseline (ResNet-50)	No	71.36 %	50.50 %
base	No	76.79 %	53.41 %
base	Sí	77.14 %	52.72 %
$k = 7$, $\alpha = 0.6$, method = mean	No	79.25 %	54.15 %
$k = 5$, $\alpha = 0.8$, method = mean	No	77.99 %	54.75 %
$k = 5$, $\alpha = 0.5$, method = mean	Sí	**80.56 %**	53.91 %
$k = 7$, $\alpha = 0.7$, method = mean	Sí	79.65 %	**55.78 %**

Los resultados de la recuperación para las mejores dos configuraciones observadas en las Figuras 5.27 y 5.28 presentan productos más relevantas, y son capaces de eliminar las sillas retornadas por el modelo *baseline*

Baseline

CLIP-base con query adaptativa
($k = 5$, $\alpha = 0.5$, method = mean)

Figura 5.27: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP-base respecto al *GlobalCategory* dentro del catálogo de IKEA. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.



Baseline

CLIP-base con query adaptativa
($k = 7$, $\alpha = 0.7$, method = mean)

Figura 5.28: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP-base respecto al *CategoryTree* dentro del catálogo de IKEA. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Por otro lado, con el modelo entrenado, esta diferencia se hace más pequeña para *GlobalCategory* y *CategoryTree*, obteniendo un mAP@20 de un **5.64 %** y de un **2.8 %**,

respectivamente, en comparación al *baseline*. Los resultados se pueden ver en la Tabla 5.22. A pesar de esto, todas las configuraciones presentes siguen superando el *baseline*.

Tabla 5.22: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Pepeganga, utilizando los *embeddings* generados por CLIP entrenado, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
baseline (ResNet-50)	No	71.36 %	50.50 %
base	No	76.08 %	52.43 %
base	Sí	76.58 %	50.67 %
$k = 5$, $\alpha = 0.9$, method = mean	No	76.76 %	52.58 %
$k = 3$, $\alpha = 0.9$, method = mean	No	75.99 %	**53.30 %**
$k = 5$, $\alpha = 0.8$, method = mean	Sí	**77.03 %**	53.24 %

A pesar de la disminución en la efectividad de la recuperación en comparación al modelo CLIP-base, las recuperaciones de imágenes de las Figuras 5.29 y 5.30 nuevamente son mejores y con mayor semántica.

Baseline

CLIP-trained con query adaptativa
($k = 5$, $\alpha = 0.8$, method = mean)

Figura 5.29: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP entrenado respecto al *GlobalCategory* dentro del catálogo de IKEA. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

CLIP-trained sin query adaptativa
($k = 3$, $\alpha = 0.9$, method = mean)

Figura 5.30: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP entrenado respecto al *CategoryTree* dentro del catálogo de IKEA. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

5.3.2.4. World Market

La recuperación sobre el catálogo de World Market mejora ampliamente al utilizar el modelo de CLIP-base, donde los resultados de la Tabla 5.23 demuestran una mejora del mAP@20 de un **12.83 %** para *GlobalCategory*, y de un **11.45 %** para *CategoryTree*. Esto con sus respectivas mejores configuraciones.

Tabla 5.23: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de *World Market*, utilizando los *embeddings* generados por CLIP sin entrenar, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
baseline (ResNet-50)	No	68.37 %	48.48 %
base	No	76.16 %	52.87 %
base	Sí	77.31 %	54.74 %
$k = 10$, $\alpha = 0.5$, method = mean	No	79.77 %	59.22 %
$k = 3$, $\alpha = 0.4$, method = mean	No	78.33 %	**59.93 %**
$k = 10$, $\alpha = 0.5$, method = mean	Sí	**81.20 %**	58.30 %
$k = 10$, $\alpha = 0.6$, method = mean	Sí	80.83 %	58.89 %

Las Figuras 5.31 y 5.32 poseen las comparaciones entre la recuperación de imágenes

que retorna el *baseline* y las que retornan los dos mejores modelos ajustados de CLIP para una lámpara del *dataset*. Las mejoras son evidentes, obteniendo en ambos casos práctica y únicamente lámparas.

Baseline

CLIP-base con query adaptativa
($k = 10$, $\alpha = 0.5$, method = mean)

Figura 5.31: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP-base respecto al *GlobalCategory* dentro del catálogo de *World Market*. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

CLIP-base sin query adaptativa
($k = 3$, $\alpha = 0.4$, method = mean)

Figura 5.32: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP-base respecto al *CategoryTree* dentro del catálogo de *World Market*. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Respecto al modelo entrenado, en la Tabla 5.24 se observa que el mAP@20 aumenta hasta un **14.83 %** para *GlobalCategory*. Sin embargo, el mejor resultado de *CategoryTree* no logra superar el de CLIP-base. De todos modos, todos los modelos presentes, con y sin configuración de *embeddings*, logran superar el modelo *baseline*.

Tabla 5.24: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de *World Market*, utilizando los *embeddings* generados por CLIP entrenado, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
baseline (ResNet-50)	No	68.37 %	48.48 %
base	No	75.08 %	52.53 %
base	Sí	75.81 %	52.77 %
$k = 10$, $\alpha = 0.2$, method = mean	No	80.70 %	58.25 %
$k = 5$, $\alpha = 0.3$, method = mean	No	79.39 %	58.96 %
$k = 10$, $T = 3.0$, method = softmax	Sí	**83.20 %**	57.84 %
$k = 5$, $\alpha = 0.3$, method = mean	Sí	81.81 %	**59.35 %**

De igual manera que para CLIP-base, las recuperaciones de las Figuras 5.33 y 5.34 mejoran notablemente.

Baseline

CLIP-trained con query adaptativa
($k = 10$, $T = 3.0$, method = softmax)

Figura 5.33: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP entrenado respecto al *GlobalCategory* dentro del catálogo de *World Market*. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

CLIP-trained con query adaptativa
($k = 5$, $\alpha = 0.3$, method = mean)

Figura 5.34: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP entrenado respecto al *CategoryTree* dentro del catálogo de *World Market*. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

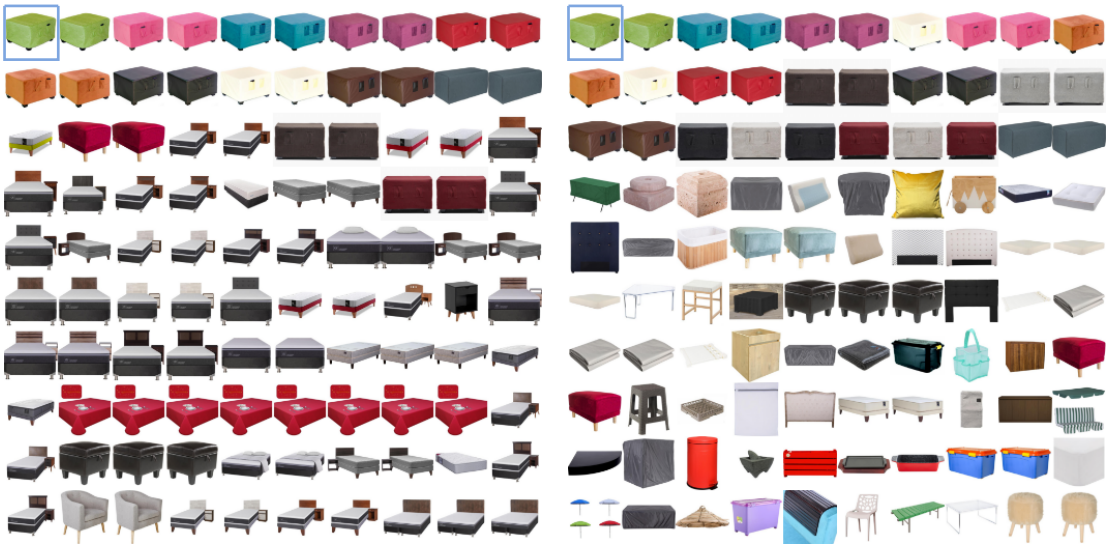
5.3.2.5. Homy

Las evaluaciones de recuperación usando *queries* de catálogo observadas en la Tabla 5.25, tanto para *GlobalCategory* como *CategoryTree*, no son significativamente mayores al modelo *baseline*, logrando una mejora en el mAP@20 de un **3.54 %** para el primero, y de un **3.43 %** para el segundo.

Tabla 5.25: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Homy, utilizando los *embeddings* generados por CLIP sin entrenar, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
baseline (ResNet-50)	No	83.56 %	65.49 %
base	No	84.60 %	67.29 %
base	Sí	84.65 %	64.95 %
$k = 7$, $\alpha = 0.4$, method = mean	No	87.02 %	**68.92 %**
$k = 7$, $\alpha = 0.4$, method = mean	Sí	**87.10 %**	67.17 %

A pesar de la pequeña mejora que se obtiene al utilizar CLIP, la recuperación si presenta una ganancia para algunos productos, como por ejemplo, el *puff* perteneciente a Homy y que se ingresa como *query*. Las recuperaciones de imágenes para este producto se pueden observar en las Figuras 5.35 y 5.36.



Baseline

CLIP-base con query adaptativa
($k = 7$, $\alpha = 0.4$, method = mean)

Figura 5.35: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP-base respecto al *GlobalCategory* dentro del catálogo de Homy. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

CLIP-base sin query adaptativa
($k = 7$, $\alpha = 0.4$, method = mean)

Figura 5.36: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP-base respecto al *CategoryTree* dentro del catálogo de Homy. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Los resultados no cambian notablemente al entrenar CLIP, obteniéndose la misma mejora en el mAP@20 para *GlobalCategory*, y una mejora de un **6.1 %** para *CategoryTree*.

Tabla 5.26: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Homy, utilizando los *embeddings* generados por CLIP entrenado, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
baseline (ResNet-50)	No	83.56 %	65.49 %
base	No	84.68 %	69.76 %
base	Sí	85.00 %	69.29 %
$k = 7$, $\alpha = 0.5$, method = mean	No	86.47 %	**71.59 %**
$k = 10$, $\alpha = 0.4$, method = mean	No	85.99 %	72.40 %
$k = 7$, $\alpha = 0.5$, method = mean	Sí	**87.10 %**	71.53 %

Los resultados observados en las Figuras 5.37 y 5.38 son similares a los que entrega CLIP-base, obteniéndose una leve mejora.

Baseline

CLIP-trained con query adaptativa
($k = 7$, $\alpha = 0.5$, method = mean)

Figura 5.37: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP entrenado respecto al *GlobalCategory* dentro del catálogo de Homy. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

CLIP-trained sin query adaptativa
($k = 7$, $\alpha = 0.5$, method = mean)

Figura 5.38: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP entrenado respecto al *CategoryTree* dentro del catálogo de Homy. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Se evalúa también la capacidad de CLIP sin entrenar para generar una buena recuperación usando las 256 *queries* reales disponibles, y al comparar el modelo *baseline*

con el mejor resultado usando el modelo de CLIP, se obtiene una sustancial mejora en el mAP@20 de un **13.2 %**, demostrando la alta capacidad de CLIP para la tarea de recuperación en casos de uso reales.

Tabla 5.27: Evaluación de la recuperación de imágenes usando *queries* reales sobre el *dataset* de Homy, utilizando los *embeddings* generados por CLIP sin entrenar, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

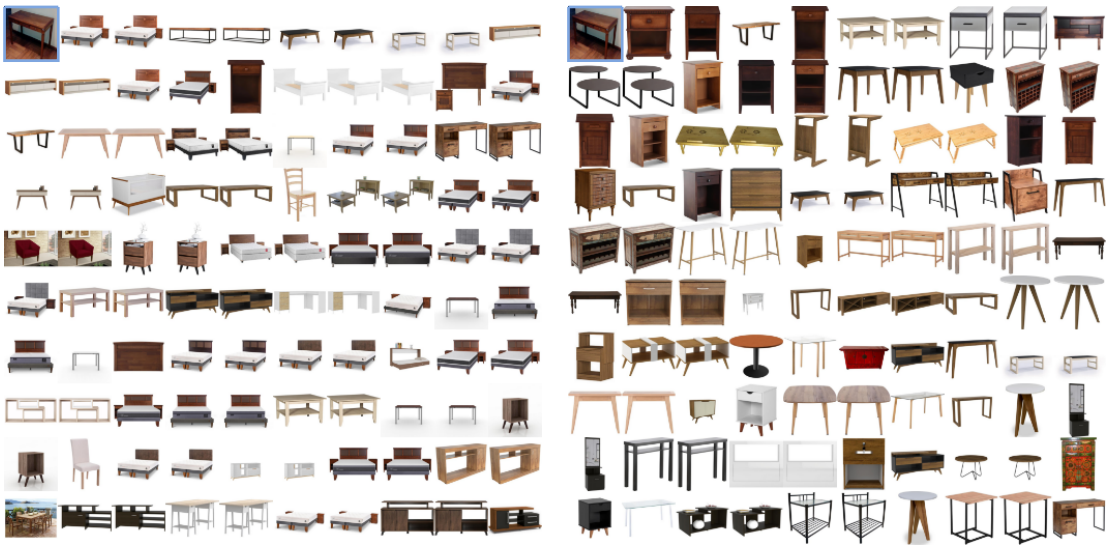
Configuración de ajuste	**Query adaptativa**	**mAP (GC)**
baseline (ResNet-50)	No	62.83 %
base	No	73.77 %
base	Sí	74.64 %
$k = 3$, $T = 0.5$, method = softmax	No	75.08 %
$k = 10$, $\alpha = 0.2$, method = mean	Sí	**76.03 %**

La recuperación para la foto real del cojín es notablemente mejor para el modelo CLIP con *embeddings* ajustados y *query* adaptativa, retornando únicamente cojines, cuyos patrones o formas no necesariamente son similares al producto ingresado como *query*, lo cual se puede observar en la Figura 5.39. Esta mejora se puede evidenciar también en la recuperación para el caso del escritorio presente en la Figura 5.40, donde las camas dejan de aparecer entre los productos recuperados, y se presentan únicamente muebles, escritorios, cómodas y mesas.

Baseline

CLIP-base con query adaptativa
($k = 10, \alpha = 0.2$, method = mean)

Figura 5.39: Ejemplo de recuperación de una foto real de un cojín para el *baseline* y el modelo con mejor ajuste para CLIP-base dentro del catálogo de Homy. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.



Baseline

CLIP-base con query adaptativa
($k = 10, \alpha = 0.2$, method = mean)

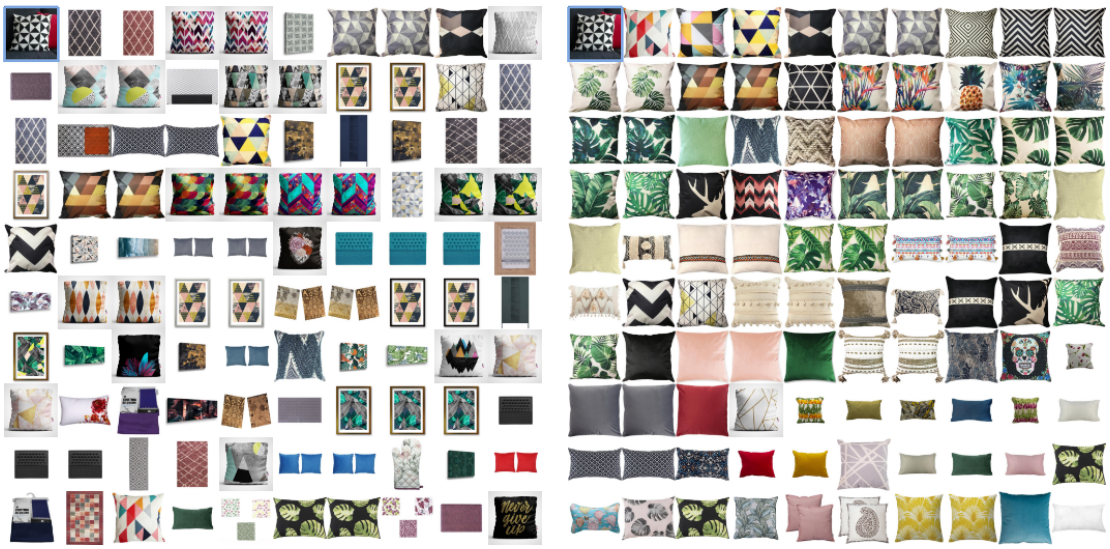
Figura 5.40: Ejemplo de recuperación de una foto real de un mueble para el *baseline* y el modelo con mejor ajuste para CLIP-base dentro del catálogo de Homy. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Al entrenar CLIP con el catálogo de Homy, las mejoras de los resultados anteriores se reducen. Se cree que esto ocurre porque el modelo fue entrenado con imágenes de

catálogo, perdiendo la capacidad de generalizar al ingresar como entrada imágenes con ruido. Sin embargo, los resultados que se observan en la Tabla 5.28 siguen superando el *baseline*, y los resultados de la recuperación de la Figura 5.41 también presenta mejoras, afirmación que no se puede realizar para la recuperación de la Figura 5.38.

Tabla 5.28: Evaluación de la recuperación de imágenes usando *queries* reales sobre el *dataset* de Homy, utilizando los *embeddings* generados por CLIP entrenado, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

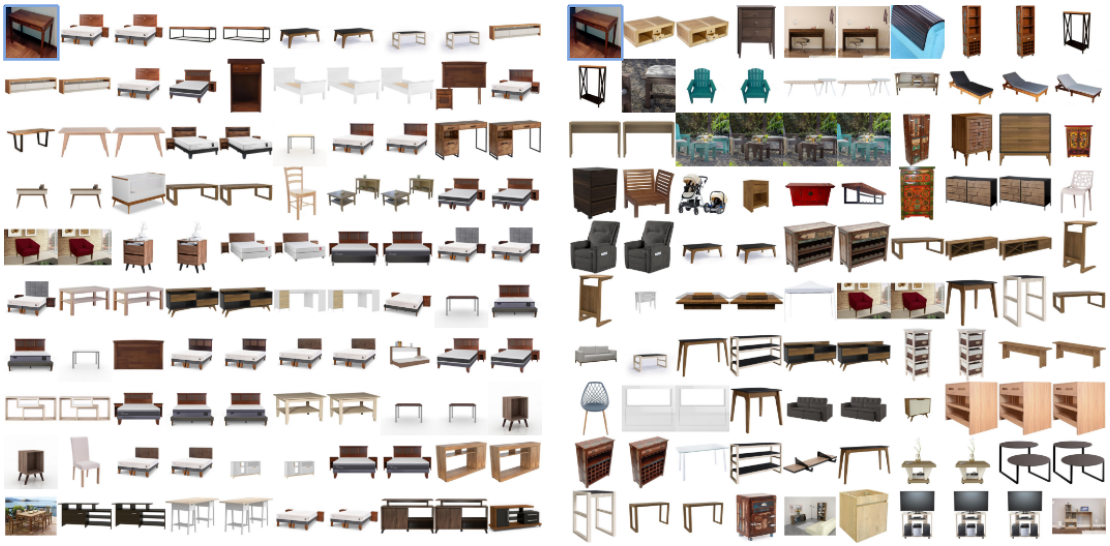
Configuración de ajuste	**Query adaptativa**	**mAP (GC)**
baseline (ResNet-50)	No	62.83 %
base	No	64.23 %
base	Sí	64.36 %
$k = 3$, $T = 3.0$, method = softmax	No	64.29 %
$k = 10$, $T = 0.5$, method = softmax	Sí	**65.76 %**



Baseline

CLIP-trained con query adaptativa
($k = 10$, $T = 0.5$, method = softmax)

Figura 5.41: Ejemplo de recuperación de una foto real de un cojín para el *baseline* y el modelo con mejor ajuste para CLIP entrenado dentro del catálogo de Homy. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.



Baseline

CLIP-trained con query adaptativa
($k = 10$, $T = 0.5$, method = softmax)

Figura 5.42: Ejemplo de recuperación de una foto real de un mueble para el *baseline* y el modelo con mejor ajuste para CLIP entrenado dentro del catálogo de Homy. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

5.3.2.6. Pepeganga

Al evaluar la recuperación usando las 100 *queries* de catálogo de Pepeganga, los valores de mAP@20 que se muestran en la Tabla 5.29, tanto para *GlobalCategory* como *CategoryTree* mejoran para todas las configuraciones presentes, logrando una mejora de un **6.76 %** para el primero, y de un **9.64 %** para el segundo.

Tabla 5.29: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Pepeganga, utilizando los *embeddings* generados por CLIP sin entrenar, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
baseline (ResNet-50)	No	87.13 %	67.83 %
base	No	91.44 %	72.40 %
base	Sí	92.67 %	72.40 %
$k = 3$, $\alpha = 0.3$, method = mean	No	93.12 %	74.51 %
$k = 5$, $\alpha = 0.3$, method = mean	No	92.61 %	75.44 %
$k = 3$, $\alpha = 0.3$, method = mean	Sí	**93.89 %**	76.25 %
$k = 5$, $\alpha = 0.2$, method = mean	Sí	93.82 %	**77.47 %**

La semántica presente en las recuperaciones de las Figuras 5.43 y 5.44 se evidencian

inmediatamente. Primero, los objetos retornados en un inicio son efectivamente productos relevantes, y segundo, la mayoría del resto de los objetos retornados entran dentro de la misma *GlobalCategory*, al estar relacionados con videojuegos.

Baseline

CLIP-base con query adaptativa
($k = 3$, $\alpha = 0.3$, method = mean)

Figura 5.43: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP-base respecto al *GlobalCategory* dentro del catálogo de Pepeganga. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Baseline

CLIP-base con query adaptativa
($k = 5$, $\alpha = 0.2$, method = mean)

Figura 5.44: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP-base respecto al *CategoryTree* dentro del catálogo de Pepeganga. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Como se puede observar en la Tabla 5.30, los resultados anteriores se ven incrementados al entrenar el modelo con datos de Pepeganga únicamente para *GlobalCategory*, aumentando la mejora de mAP@20 en un **7.06 %** sobre el *baseline*. Sin embargo, la mejora para *CategoryTree* se reduce a un **7.10 %**.

Tabla 5.30: Evaluación de la recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Pepeganga, utilizando los *embeddings* generados por CLIP entrenado, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

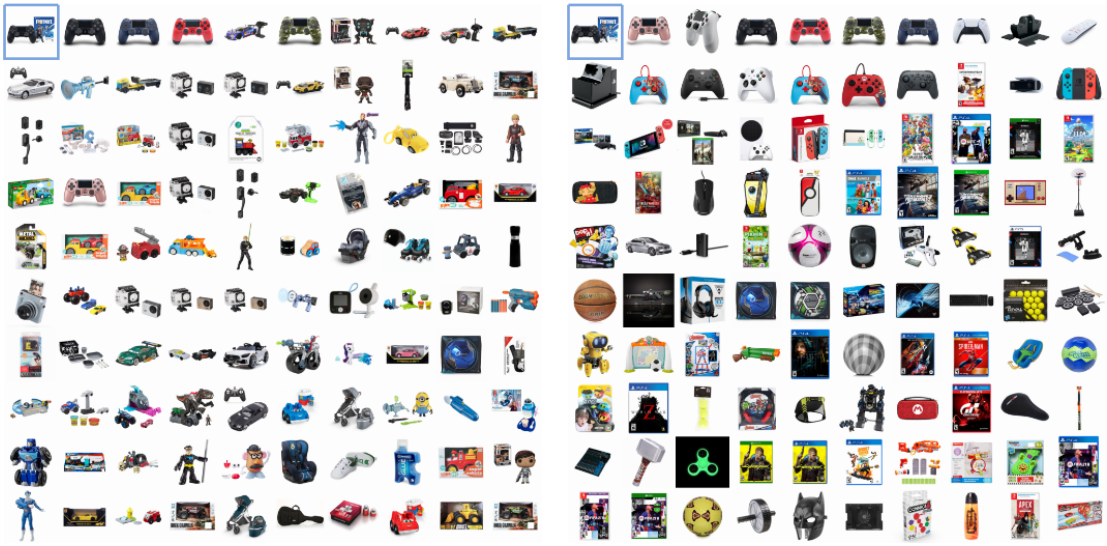
Configuración de ajuste	Query adaptativa	mAP (GC)	mAP (CT)
baseline (ResNet-50)	No	87.13 %	67.83 %
base	No	90.12 %	70.63 %
base	Sí	90.97 %	72.33 %
$k = 6$, method = sim	No	93.60 %	73.44 %
$k = 10$, $\alpha = 0.5$, method = mean	No	92.26 %	73.49 %
$k = 5$, $\alpha = 0.2$, method = mean	Sí	**94.19 %**	74.43 %
$k = 6$, method = sim	Sí	94.13 %	**74.93 %**

Observando las Figuras 5.45 y 5.46 se puede apreciar que en la recuperación de imágenes, al menos para el producto ingresado como *query*, mejora, al retornar una mayor cantidad de mandos para consola.

Baseline

CLIP-trained con query adaptativa
($k = 5$, $\alpha = 0.2$, method = mean)

Figura 5.45: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP entrenado respecto al *GlobalCategory* dentro del catálogo de Pepeganga. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.



Baseline

CLIP-trained con query adaptativa
($k = 6$, method = sim)

Figura 5.46: Ejemplo de recuperación de un producto para el *baseline* y el modelo con mejor ajuste para CLIP entrenado respecto al *CategoryTree* dentro del catálogo de Pepeganga. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

Para casos de uso cotidianos, el modelo CLIP sin entrenar es evaluado con 300 *queries* reales. La Tabla 5.31 muestra una sorpresiva mejora de hasta un **24.11 %** en el mAP@20

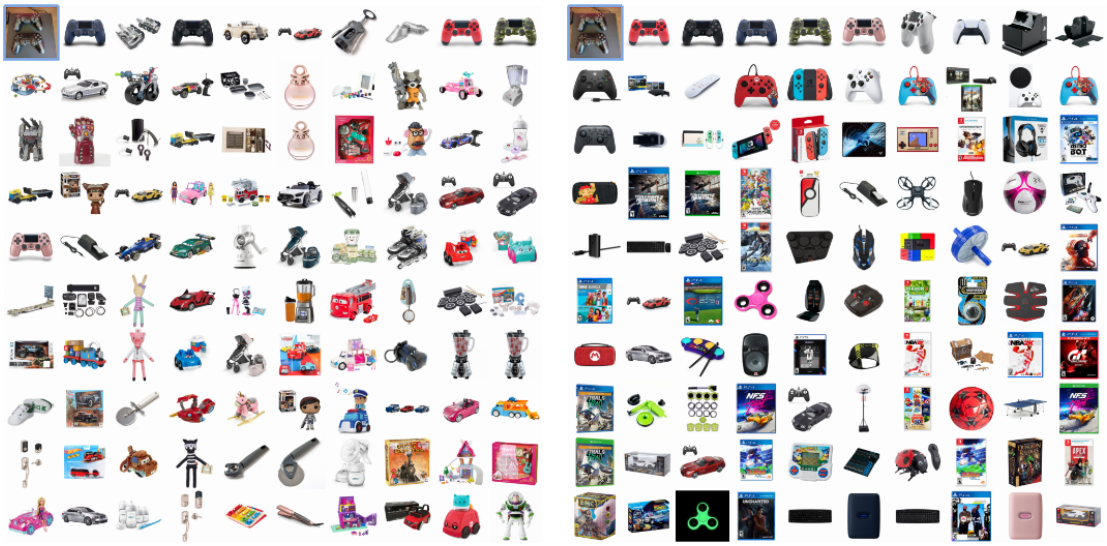
resultante de la mejor configuración de ajuste.

Tabla 5.31: Evaluación de la recuperación de imágenes usando *queries* reales sobre el *dataset* de Pepeganga, utilizando los *embeddings* generados por CLIP sin entrenar, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

Configuración de ajuste	**Query adaptativa**	**mAP (GC)**
baseline (ResNet-50)	No	57.21 %
base	No	77.04 %
base	Sí	77.98 %
$k = 10$, $\alpha = 0.2$, method = mean	No	80.76 %
$k = 7$, $\alpha = 0.2$, method = mean	Sí	**81.32 %**

Para la recuperación de la foto real del mando de consola, observado en la Figura 5.47, se puede notar como el modelo ajustado de CLIP-base es capaz de aumentar la semántica, obteniendo dentro de las primeras posiciones todos los productos relevantes observados en dicha recuperación, y los productos que no son relevantes tienden a caer dentro de la misma *GlobalCategory*.

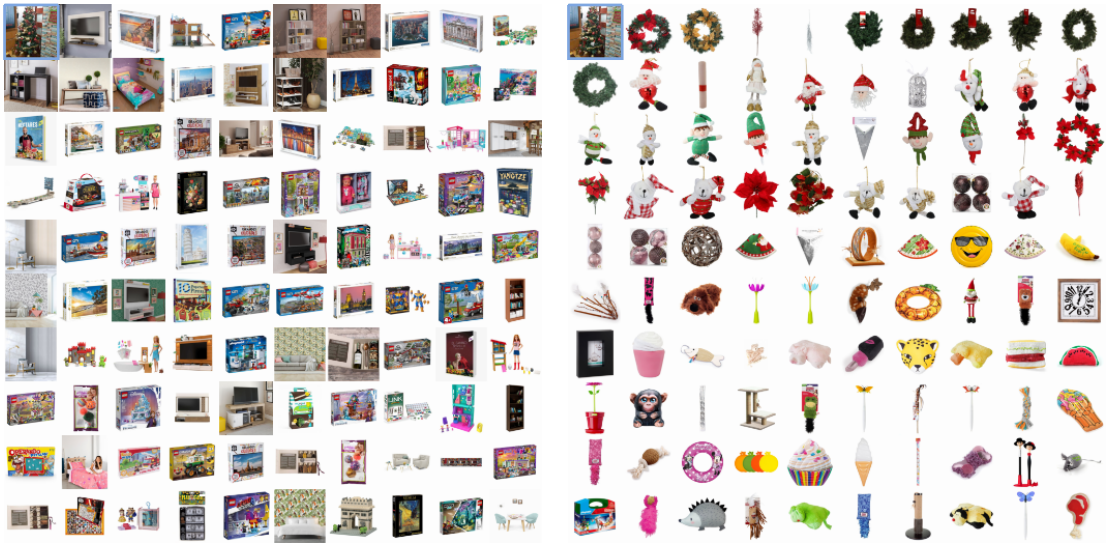
En el caso de la foto del árbol de navidad, el aumento de semántica es aún mayor y sorpresivo. El catálogo de Pepeganga no posee árboles de navidad, sin embargo, cerca de la mitad de los productos retornados, y que además se encuentran dentro de las primeras posiciones, son productos navideños.



Baseline

CLIP-base con query adaptativa
($k = 7$, $\alpha = 0.2$, method = mean)

Figura 5.47: Ejemplo de recuperación de una foto real de una consola con controles para el *baseline* y el modelo con mejor ajuste para CLIP-base dentro del catálogo de Pepeganga. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.



Baseline

CLIP-base con query adaptativa
($k = 7$, $\alpha = 0.2$, method = mean)

Figura 5.48: Ejemplo de recuperación de una foto real de un árbol de navidad para el *baseline* y el modelo con mejor ajuste para CLIP-base dentro del catálogo de Pepeganga. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

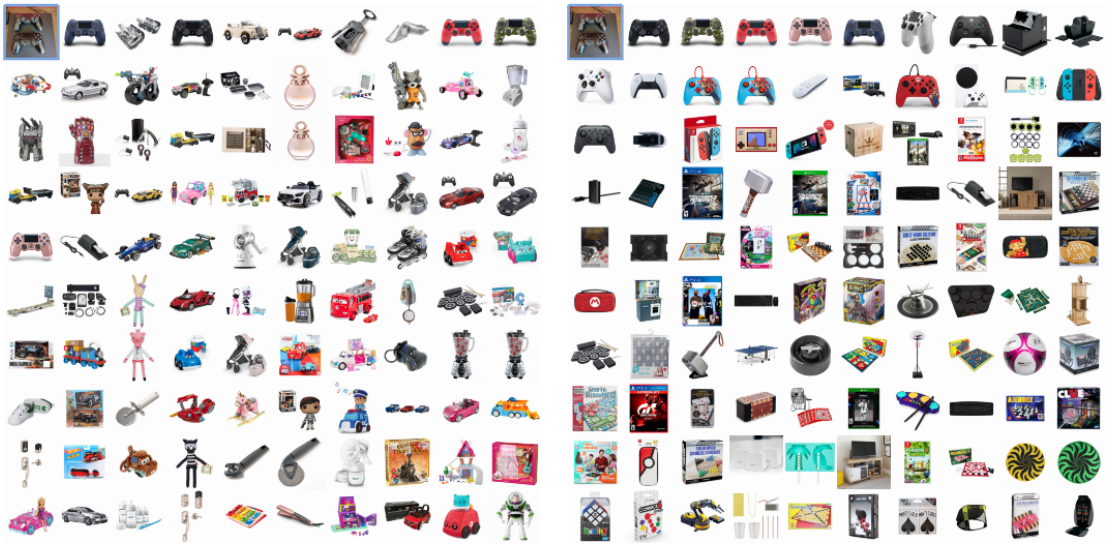
Respecto a la evaluación sobre el modelo CLIP entrenado, al igual que ocurre con el catálogo de Homy, la diferencia con el modelo CLIP sin entrenar nuevamente disminuye,

obteniéndose una mejora de mAP@20 sobre el *baseline* de un **19.01 %**. A pesar de esto, las mejores son significativas para todos los modelos presentes en la Tabla 5.32.

Tabla 5.32: Evaluación de la recuperación de imágenes usando *queries* reales sobre el *dataset* de Pepeganga, utilizando los *embeddings* generados por CLIP entrenado, modificados mediante algoritmos de ajuste de *embeddings* y utilizando *query* adaptativa.

Configuración de ajuste	Query adaptativa	mAP (GC)
baseline (ResNet-50)	No	57.21 %
base	No	71.75 %
base	Sí	71.95 %
$k = 10$, $\alpha = 0.2$, method = mean	No	**76.22 %**
$k = 10$, $\alpha = 0.2$, method = mean	Sí	75.86 %

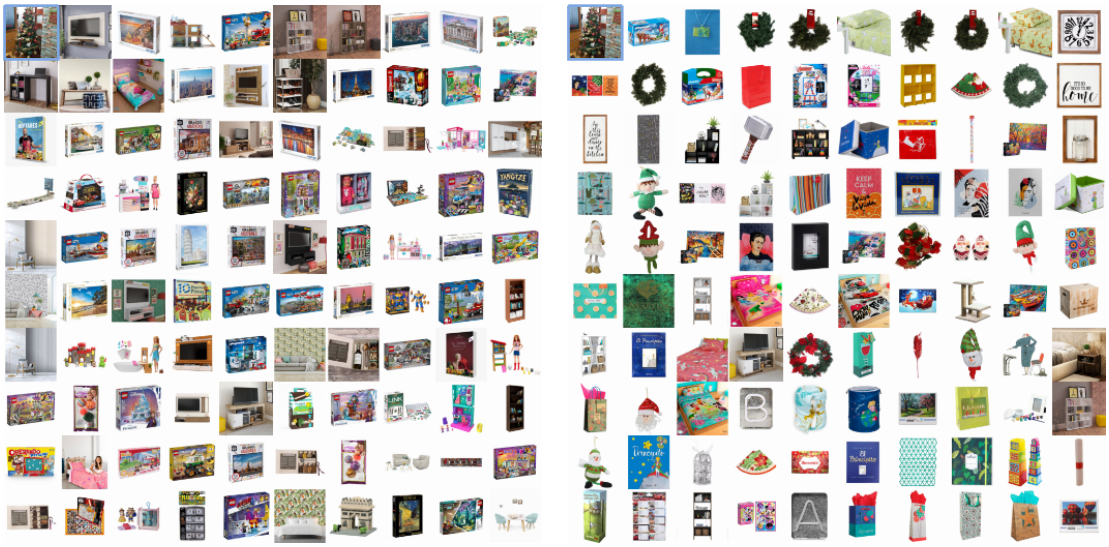
Ambas recuperaciones que obtiene el modelo CLIP entrenado y se observan en las Figuras 5.49 y 5.50, tanto para el mando de consola como para el árbol de navidad, mejoran respecto al *baseline*, sin embargo, esta mejora no se acerca a la que se obtiene con el modelo CLIP-base.



Baseline

CLIP-trained sin query adaptativa
($k = 10$, $\alpha = 0.2$, method = mean)

Figura 5.49: Ejemplo de recuperación de una foto real de una consola con controles para el *baseline* y el modelo con mejor ajuste para CLIP entrenado dentro del catálogo de Pepeganga. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.



Baseline

CLIP-trained sin query adaptativa
($k = 10$, $\alpha = 0.2$, method = mean)

Figura 5.50: Ejemplo de recuperación de una foto real de un árbol de navidad para el *baseline* y el modelo con mejor ajuste para CLIP entrenado dentro del catálogo de Pepeganga. La imagen de consulta es la encerrada en un cuadrado azulado, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo.

5.4. Sumario

Para lograr entender de manera más general la mejora que puede significar el uso de ajuste de *embeddings*, de *query* adaptativa, y del modelo CLIP, se decide hacer una evaluación más global. Esta consiste únicamente en promediar los valores de mAP@20 para todos los *datasets*, separando los modelos sin ajustar de los ajustados, y los que utilizan *query* adaptativa de los que no hacen uso de esta. Además, esto se realiza independientemente para *queries* de catálogo y para *queries* reales, así como para los modelos ResNet-50 y CLIP.

La Tabla 5.33 muestra los mAP@20 promedio que se obtienen al realizar consultas con imágenes de catálogo y utilizando como modelo base la ResNet-50. Se puede notar que, en general, conviene realizar un ajuste de *embeddings* y utilizar una *query* adaptativa al momento de usar un modelo ResNet-50 entrenado en *ImageNet* como base.

Tabla 5.33: Valores de mAP@20 promedio sobre todos los *datasets* usando ResNet-50 como base y *queries* de catálogo.

Tipo de modelo	Query adaptativa	mAP promedio (GC)	mAP promedio (CT)
ResNet-50	No	77.11 %	55.51 %
ResNet-50	Sí	78.00 %	55.44 %
ResNet-50 ajustado	No	80.22 %	57.92 %
ResNet-50 ajustado	Sí	**80.69 %**	**58.78 %**

En la Tabla 5.34 se evidencia que, con el entrenamiento realizado sobre CLIP, no se obtienen mejores resultados en la recuperación de imágenes, tanto a grano grueso como grano fino, al compararlo con CLIP-base y utilizar imágenes de catálogo. A pesar de esto, la mejora sobre el *baseline* es amplia para ambos modelos, y el aporte del ajuste de *embeddings* y del uso de una *query* adaptativa se justifica para obtener mejores resultados.

Tabla 5.34: Valores de mAP@20 promedio sobre todos los *datasets* usando CLIP como base y *queries* de catálogo.

Tipo de modelo	Query adaptativa	mAP promedio (GC)	mAP promedio (CT)
baseline (ResNet-50)	No	77.11 %	55.51 %
CLIP-base	No	81.97 %	57.85 %
CLIP-base	Sí	82.50 %	57.09 %
CLIP-entrenado	No	81.08 %	57.27 %
CLIP-entrenado	Sí	81.79 %	57.03 %
CLIP-base ajustado	No	83.60 %	**60.15 %**
CLIP-base ajustado	Sí	**84.62 %**	59.65 %
CLIP-entrenado ajustado	No	83.32 %	60.04 %
CLIP-entrenado ajustado	Sí	84.30 %	59.69 %

Respecto a la recuperación de imágenes usando como *query* fotos reales o imágenes con ruido, se puede observar que tanto la *query* adaptativa como el ajuste de *embeddings* constituye una herramienta útil al momento de buscar una leve mejora en dicha recuperación, como se puede apreciar en la Tabla 5.35.

Tabla 5.35: Valores de mAP@20 promedio sobre los *datasets* de Pepeganga y Homy usando ResNet-50 como base y *queries* reales.

Tipo de modelo	Query adaptativa	mAP promedio (GC)
ResNet-50	No	60.02 %
ResNet-50	Sí	60.85 %
ResNet-50 ajustado	No	61.28 %
ResNet-50 ajustado	Sí	**63.97 %**

Por otro lado, cuando se utiliza CLIP como modelo base, el uso de una *query* adaptativa y ajuste de *embeddings* es sustancial al momento de querer obtener una mejora en la semántica de la recuperación de imágenes, lo cual se puede observar en la Tabla 5.36.

Tabla 5.36: Valores de mAP@20 promedio sobre los *datasets* de Pepeganga y Homy usando CLIP como base y *queries* reales.

Tipo de modelo	Query adaptativa	mAP promedio (GC)
baseline (ResNet-50)	No	60.02 %
CLIP-base	No	75.41 %
CLIP-base	Sí	76.31 %
CLIP-entrenado	No	67.99 %
CLIP-entrenado	Sí	68.16 %
CLIP-base ajustado	No	77.92 %
CLIP-base ajustado	Sí	**78.68 %**
CLIP-entrenado ajustado	No	70.26 %
CLIP-entrenado ajustado	Sí	70.81 %

Independiente del catálogo de comercio electrónico, el efecto del ajuste de *embeddings*, el uso de una *query* adaptativa, y el uso del modelo CLIP, para *queries* de catálogo y *queries* reales, también se puede visualizar en los diagramas de caja de las Figuras 5.51, 5.52 y 5.53, observándose la clara mejora al momento de introducir estas propuestas en la recuperación de imágenes.

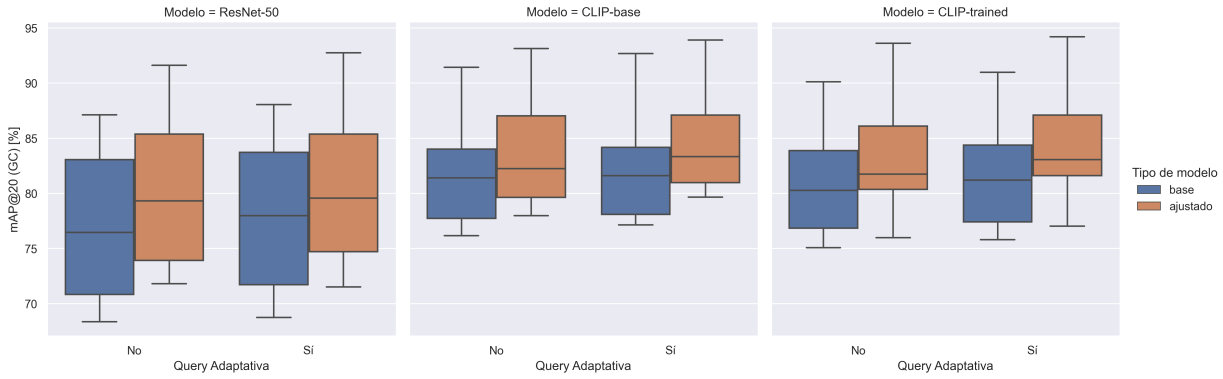


Figura 5.51: Comparación de las distribuciones de mAP@20 promedio para *GlobalCategory* utilizando *queries* de catálogo.

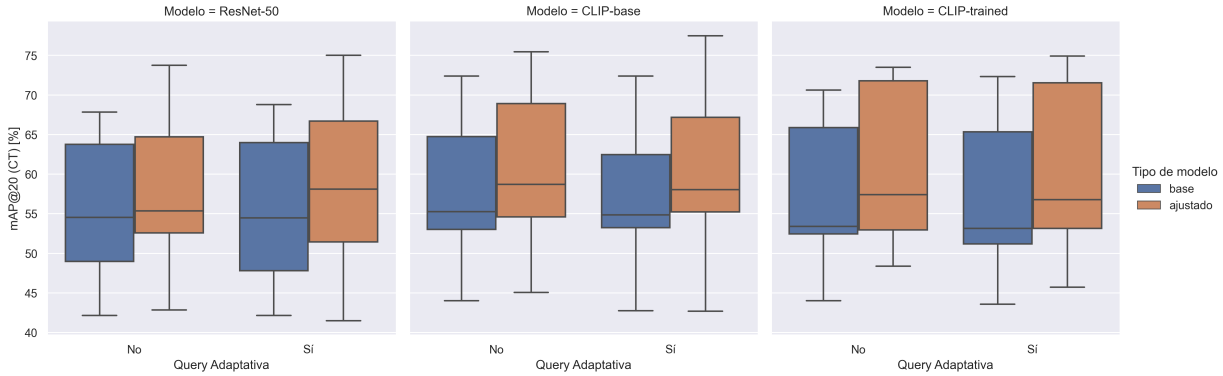


Figura 5.52: Comparación de las distribuciones de mAP@20 promedio para *CategoryTree* utilizando *queries* de catálogo.

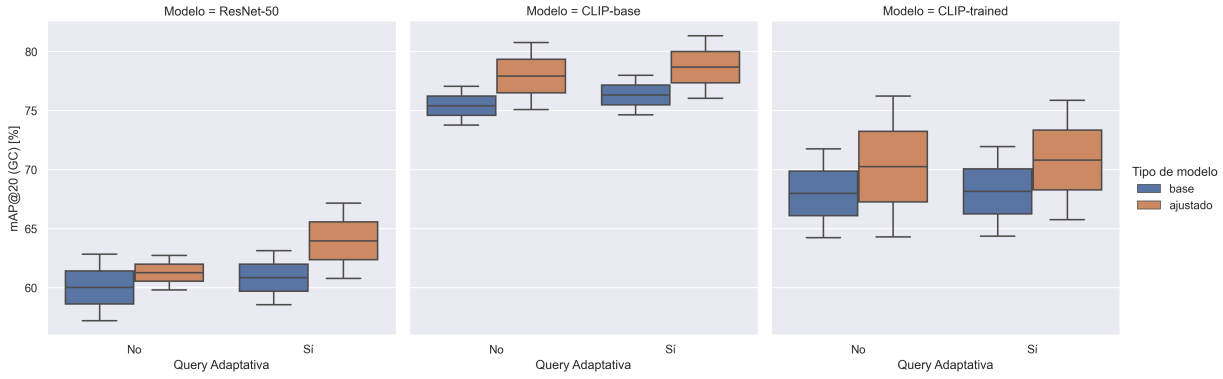


Figura 5.53: Comparación de las distribuciones de mAP@20 promedio para *GlobalCategory* utilizando *queries* reales.

Además, los gráficos de barra de las Figuras 5.54, 5.55 y 5.56 permiten visualizar la diferencia entre el modelo *baseline* y el modelo propuesto, ya sea con ajuste de *embeddings*, *query* adaptativa, o usando CLIP, con mejores resultados para cada *dataset*, independiente del modelo base.

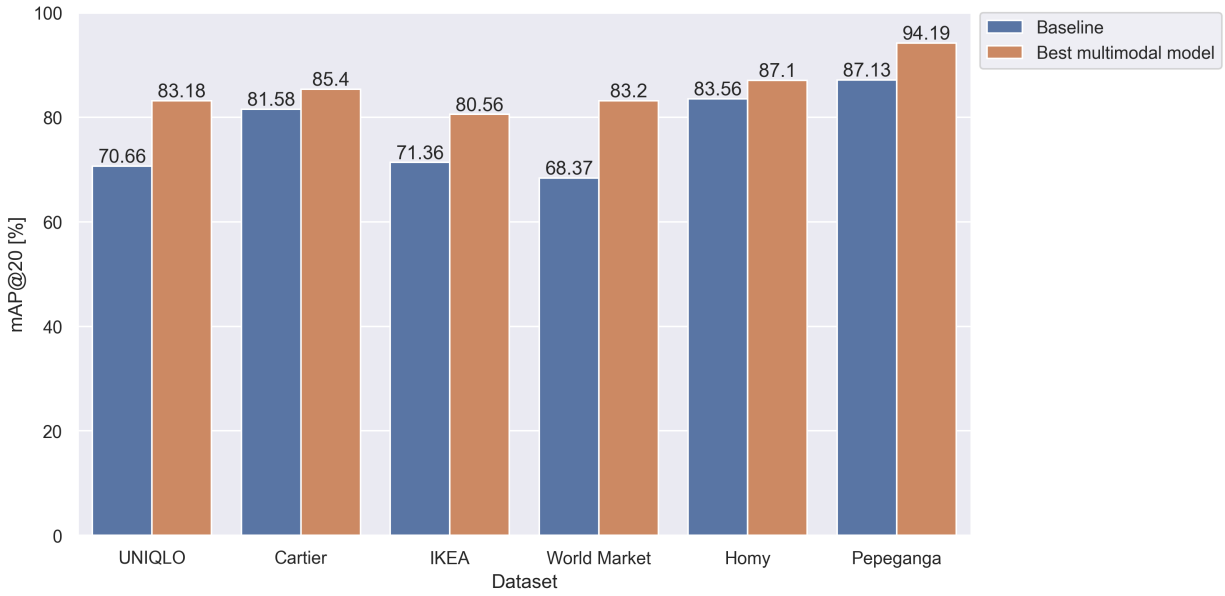


Figura 5.54: Comparación de mAP@20 para *GlobalCategory* en cada *dataset* entre el modelo base y el mejor modelo ajustado utilizando *queries* de catálogo.

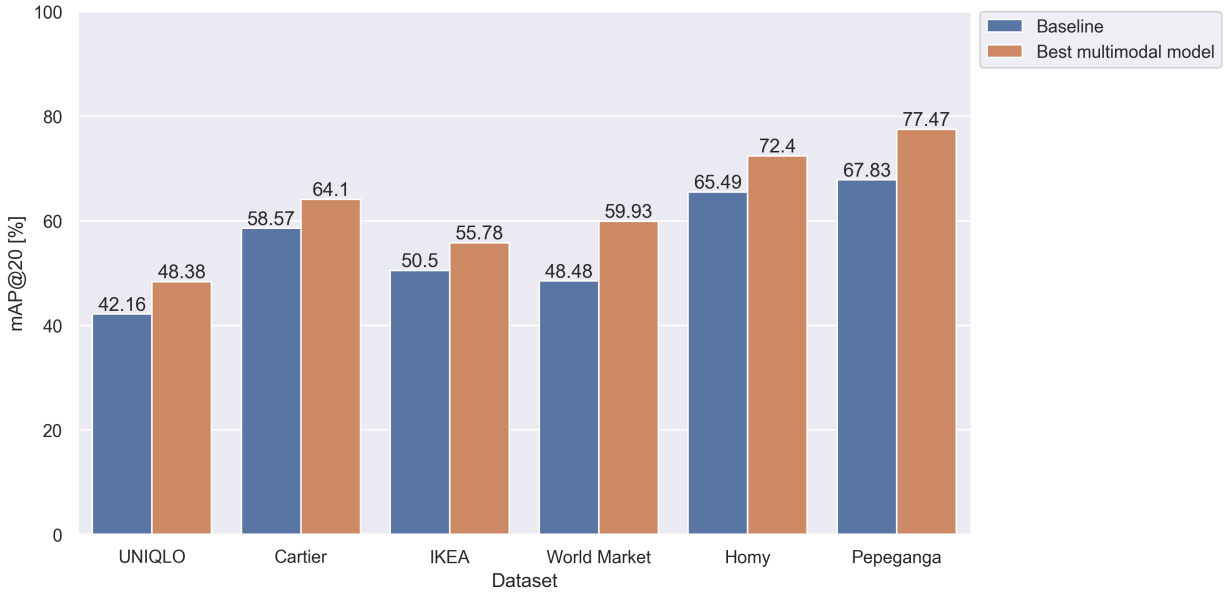


Figura 5.55: Comparación de mAP@20 para *CategoryTree* en cada *dataset* entre el modelo base y el mejor modelo ajustado utilizando *queries* de catálogo.

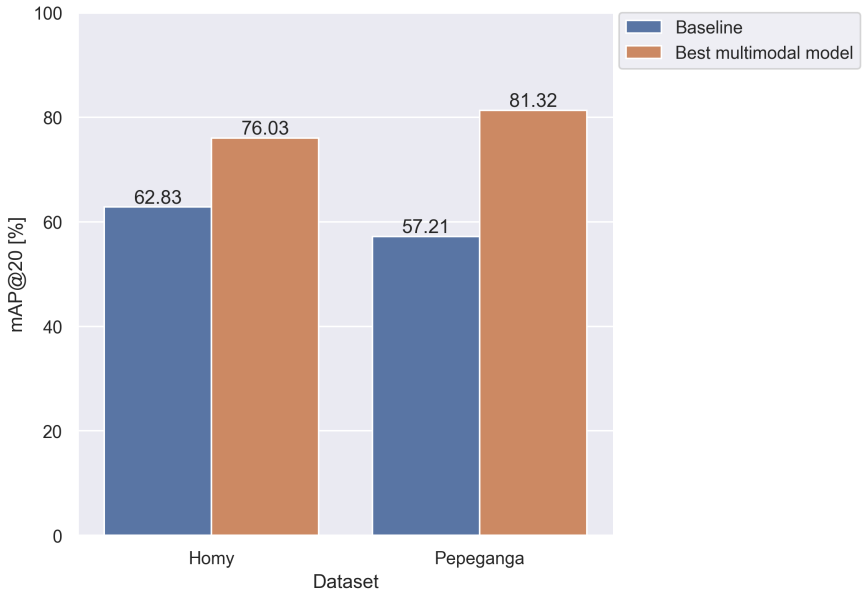


Figura 5.56: Comparación de mAP@20 para *GlobalCategory* en cada *dataset* entre el modelo base y el mejor modelo ajustado utilizando *queries* reales.

Por otra parte, también se realiza una evaluación para medir la capacidad de UMAP al momento de generar recuperaciones de imágenes con semántica, entrenando con los *embeddings* que genera la ResNet-50, tanto sin ajustar, como ajustados. Sin embargo, los resultados no son favorables al momento de comparar con el *baseline*. El desarrollo y los resultados se encuentran detallados en el Anexo A.1. A pesar de esto, CLIP puede beneficiar enormemente la recuperación de imágenes generada por UMAP si se utilizan sus representaciones visuales al momento de entrenar, debido a la mejor semántica presente en los *embeddings* de CLIP, y por ende, mejor topología local.

Por último, como un enfoque inicial en el trabajo, se implementan redes neuronales cuya función consiste en aprender a representar espacios visuales como espacios de texto. Se generan múltiples redes con distintas configuraciones, y se evalúa la recuperación de imágenes usando tanto *queries* de catálogo como *queries* con fotos reales e imágenes con ruido. El desarrollo y los resultados se pueden observar en el Anexo A.2.

6. Conclusiones

En este trabajo se diseñan, implementan y evalúan técnicas y modelos autosupervisados para aumentar la semántica en la recuperación de imágenes basada en contenido mediante la utilización de información visual y textual. Para llevar a cabo esto, primero se crean seis *datasets* de catálogos de comercio electrónico de distinta índole, sobre los cuales se realizan estudios preliminares para entender la diferencia de semántica existente entre las representaciones visuales y las de texto, en los cuales se comparan la recuperación de imágenes usando las imágenes de productos como consulta y la recuperación usando sus respectivos textos. Por último, en esta parte también se generan visualizaciones en dos dimensiones de ambas representaciones mediante UMAP, para así poder observar los datos y su estructura, y medir la capacidad representativa de los *embeddings* a través de la generación y evaluación de clusters en base a estos. Los resultados que se obtienen sugieren un aporte significativo en la incorporación de texto para poder aumentar la semántica en la recuperación de imágenes.

Se proponen distintos enfoques para mejorar la recuperación de imágenes basada en contenido. El primero, que no utiliza las descripciones de los productos, consiste en usar las representaciones visuales generadas por UMAP para diferentes dimensiones en la implementación de la recuperación de imágenes. Esta propuesta se evalúa únicamente en el *dataset* de Pepeganga y se decide no continuar experimentando con ella debido a que los resultados no presentan una mejora respecto al *baseline* que define la ResNet-50. También se generan múltiples redes neuronales para transformar representaciones visuales en representaciones de texto, sin embargo, la capacidad de generalización de estas redes es significativamente baja en comparación al modelo *baseline*, dejando a este método como inviable.

Después se diseñan e implementan algoritmos que buscan ajustar los *embeddings* visuales en base a las descripciones de los productos. Además, se implementa un algoritmo para ajustar el *embedding* de una *query* ingresada en tiempo real. Estos métodos superan el *baseline* definido en todos los *datasets* de evaluación a grano grueso y grano fino, tanto al utilizar *queries* de catálogo como al usar *queries* reales y con ruido. Sin embargo, los métodos no siempre pueden generar una mejora significativa en casos donde la recuperación de imágenes que se obtiene usando el *baseline* para un producto ya presenta una baja efectividad.

Para solucionar el problema anterior, se propone el último método, que consiste en utilizar un modelo multi-modal, llamado CLIP. El entrenamiento de este modelo busca aprender un espacio conjunto tanto para las imágenes como el texto. El modelo CLIP de

por sí ya genera resultados notablemente mejores sin modificaciones a sus *embeddings* ni con el uso de una *query* adaptativa. Sin embargo, al observar los resultados de los experimentos, se puede apreciar que estas mejoras incrementan aún más al utilizar estos dos métodos de ajuste recién mencionados. Además, la capacidad de generalizar de CLIP es notablemente mayor a la de ResNet-50, obteniendo una amplia mejora en la recuperación al momento de utilizar fotos reales e imágenes con ruido.

7. Trabajos futuros

Respecto al ajuste de *embeddings* de catálogo, se pueden diseñar e implementar otros algoritmos que permitan generar un nuevo *embedding* visual más representativo y con más semántica, mejorando así el resultado de la recuperación de imágenes.

En cuanto al algoritmo para adaptar la *query* ingresada en tiempo real, nuevamente se pueden buscar distintas opciones para generar el nuevo *embedding* visual, considerando que en este trabajo se utiliza un único algoritmo. Además, se puede estudiar y probar con más condiciones para decidir si adaptar el *embedding* de una *query* o no.

Dadas las diferencias presentes en los resultados de la recuperación de imágenes entre el modelo CLIP-base y CLIP entrenado con imágenes de catálogo, sería interesante entrenar CLIP utilizando, además de imágenes de catálogo, imágenes reales de productos con descripciones asociadas a ellas mediante mecanismos que faciliten la alimentación del modelo con este tipo de datos.

Por último, un trabajo futuro podría estudiar la capacidad de CLIP dentro del contexto de comercio electrónico, pero enfocado en la recuperación de imágenes basada en dibujos.

Bibliografía

[1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," *arXiv preprint arXiv:2103.00020*, 2021.

[2] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[5] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[6] P. Torres and J. M. Saavedra, "Compact and effective representations for sketch-based image retrieval," *arXiv preprint arXiv:2104.10278*, 2021.

[7] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[8] G. Yauney, J. Hessel, and D. Mimno, "Domain-specific lexical grounding in noisy visual-textual documents," *arXiv preprint arXiv:2010.16363*, 2020.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[11] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *The journal of machine learning research*, vol. 3, pp. 1137–1155, 2003.

[12] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[13] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information*

science, vol. 41, no. 6, pp. 391–407, 1990.

[14] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[16] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[17] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

[18] S. Nagel, "Cc-news," *URL: http://web. archive. org/save/http://commoncrawl. org/2016/10/newsdatasetavailable*, 2016.

[19] A. Gokaslan and V. Cohen, "Url http://web. archive. org/save/http://skylion007. github. io."

[20] T. H. Trinh and Q. V. Le, "A simple method for commonsense reasoning," *arXiv preprint arXiv:1806.02847*, 2018.

[21] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mpnet: Masked and permuted pretraining for language understanding," *arXiv preprint arXiv:2004.09297*, 2020.

[22] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[23] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," *arXiv preprint arXiv:2006.07733*, 2020.

A. Anexo

A.1. Recuperación de imágenes utilizando UMAP y ajuste de *embeddings* visuales

Estos experimentos se realizan pensando en los buenos resultados que se obtienen en los estudios preliminares, y en la ganancia que se puede obtener respecto a la optimización de la recuperación de imágenes al utilizar *embeddings* de tamaños drásticamente más pequeños.

Se evalúa la capacidad de UMAP para reducir las dimensiones manteniendo, o mejorando, la semántica presente en los *embeddings* visuales. Para esto, se utiliza el catálogo de Pepeganga. Primero, se realiza una recuperación evaluando el mAP@20 para las 100 *queries* de catálogo, usando el modelo base con los *embeddings* originales que se obtienen de la ResNet-50 entrenada en *ImageNet*. Luego, al tomar en cuenta la mejora que se logra al evaluar tanto sobre *queries* de catálogo como *queries* reales cuando se realiza un ajuste de *embeddings* visuales, se hace una nueva recuperación usando la mejor configuración para *GlobalCategory* en el catálogo de Pepeganga. En ambos casos, se incluye el mAP@20 que se obtiene al utilizar el modelo *baseline* (ResNet-50 sin UMAP y sin ajuste de *embeddings*).

Por último, UMAP permite escoger un número de épocas para entrenar la reducción de dimensiones, razón por la cual los experimentos se realizan usando tanto las épocas por defecto que se definen internamente al usar la librería de Python especializada para UMAP, como la máxima cantidad de épocas recomendadas, equivalente a 500.

A.1.1. *Embeddings* visuales sin ajustar

Al utilizar la cantidad de épocas por defecto que define UMAP al momento de recibir los datos de entrada la recuperación de imágenes empeora notablemente en algunos casos, como se puede observar en la Tabla A.1, donde el mejor resultado se obtiene al reducir las dimensiones a 8, reduciendo el mAP@20 del *baseline* en un 2.56 % para *GlobalCategory* y un 8.22 % para *CategoryTree*.

Tabla A.1: Evaluación de recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Pepeganga, en distintas dimensiones, utilizando los *embeddings* generados por ResNet-50 para entrenar UMAP con épocas por defecto.

Modelo	UMAP	Dimensiones	mAP (GC)	mAP (CT)
ResNet-50	No	2048	**87.13 %**	**67.83 %**
ResNet-50	Sí	32	63.32 %	35.87 %
ResNet-50	Sí	16	79.77 %	55.61 %
ResNet-50	Sí	8	84.57 %	59.61 %
ResNet-50	Sí	4	83.64 %	59.23 %

Por otro lado, cuando el entrenamiento de UMAP se hace durante 500 épocas, los resultados para las distintas dimensiones son más cercanos entre sí. Nuevamente el mejor resultado se logra al reducir a 8 dimensiones, sin embargo, este disminuye el mAP@20 obtenido por el *baseline* para *GlobalCategory* en un 2.85 %, y en un 6.43 % para *CategoryTree*. Los resultados se pueden observar en la Tabla A.2

Tabla A.2: Evaluación de recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Pepeganga, en distintas dimensiones, utilizando los *embeddings* generados por ResNet-50 para entrenar UMAP en 500 épocas.

Modelo	UMAP	Dimensiones	mAP (GC)	mAP (CT)
ResNet-50	No	2048	**87.13 %**	**67.83 %**
ResNet-50	Sí	32	80.15 %	56.12 %
ResNet-50	Sí	16	84.18 %	61.14 %
ResNet-50	Sí	8	84.28 %	61.40 %
ResNet-50	Sí	4	83.00 %	60.08 %

A.1.2. *Embeddings* visuales ajustados

La configuración de ajuste de *embeddings* que se usa corresponde a la configuración que obtiene el mejor resultado de recuperación utilizando *queries* de catálogo dentro del *dataset* de Pepeganga. Sin embargo, cuando se usan las épocas por defecto que define UMAP, los resultados que se logran siguen siendo significativamente peores que el *baseline*.

Tabla A.3: Evaluación de recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Pepeganga, en distintas dimensiones, utilizando los *embeddings* generados por ResNet-50 y modificados mediante algoritmos de ajuste de *embeddings* para entrenar UMAP en épocas por defecto.

Configuración de ajuste	UMAP	Dim.	mAP (GC)	mAP (CT)
base	No	2048	87.13 %	67.83 %
$k = 3$, $T = 0.5$, method = softmax	No	2048	**91.60 %**	**71.87 %**
$k = 3$, $T = 0.5$, method = softmax	Sí	32	76.23 %	51.29 %
$k = 3$, $T = 0.5$, method = softmax	Sí	16	83.42 %	59.41 %
$k = 3$, $T = 0.5$, method = softmax	Sí	8	82.98 %	58.24 %
$k = 3$, $T = 0.5$, method = softmax	Sí	4	80.72 %	56.79 %

Cuando se realiza un entrenamiento de UMAP con 500 épocas, el mejor mAP@20 que se obtiene, tanto para *GlobalCategory* como *CategoryTree*, no se diferencia considerablemente que cuando se realiza con las épocas por defecto. A pesar de esto, se observa una menor diferencia de resultados en cuanto a la dimensión usada.

Tabla A.4: Evaluación de recuperación de imágenes usando *queries* de catálogo sobre el *dataset* de Pepeganga, en distintas dimensiones, utilizando los *embeddings* generados por ResNet-50 y modificados mediante algoritmos de ajuste de *embeddings* para entrenar UMAP en 500 épocas.

Configuración de ajuste	UMAP	Dim.	mAP (GC)	mAP (CT)
base	No	2048	87.13 %	67.83 %
$k = 3$, $T = 0.5$, method = softmax	No	2048	**91.60 %**	**71.87 %**
$k = 3$, $T = 0.5$, method = softmax	Sí	32	82.97 %	59.10 %
$k = 3$, $T = 0.5$, method = softmax	Sí	16	83.35 %	59.47 %
$k = 3$, $T = 0.5$, method = softmax	Sí	8	83.90 %	59.77 %
$k = 3$, $T = 0.5$, method = softmax	Sí	4	81.66 %	58.24 %

En base a los resultados que se obtienen al evaluar las *queries* de catálogo sobre el *dataset* de Pepeganga, se decide no continuar experimentando con UMAP para el resto de catálogos ni con *queries* reales.

A.2. Redes neuronales *feed-forward fully-connected* para transformación de *embeddings* visuales

Considerando la semántica que tiene el texto descriptivo de un producto, surge la idea de implementar una red neuronal *feed-forward fully-connected* capaz de aprender el es-

pacio vectorial generado por las imágenes de un catálogo, para así poder obtener, en base a este, un espacio parecido al que se genera con las descripciones del mismo catálogo. En este caso en particular, se utilizan los *embeddings* de texto generados por RoBERTa y los visuales que genera la ResNet-50.

Se construyen variadas redes neuronales, y las características que pueden poseer son las siguientes:

- Número de capas ocultas dentro del rango $[1, 4]$.
- Con y sin *dropout*, con probabilidad $p = 0.2$.
- Con y sin *Batch Normalization*.

En adición, se diseñaron e implementaron redes neuronales con las mismas características ya mencionadas, pero donde el espacio a generar como *output* corresponde al espacio reducido mediante la utilización de UMAP para las dimensiones 128 y 8. La estructura general de las redes neuronales se puede observar en la Figura A.1.

El entrenamiento se realiza con los datos del catálogo de Pepeganga, donde el 80 % de los datos son datos de entrenamiento, el 20 % de prueba, y las 100 *queries* que se extraen previamente se usan para evaluar. La elección del mejor modelo se realiza en base a la evaluación sobre las 100 *queries* de catálogo en todos los modelos generados, y luego se toma el que mejor mAP@20 obtiene para cada configuración de dimensión. Por último, la función de pérdida utilizada se basa en la definida en BYOL [23], correspondiente a un *mean squared error*. Considerando a $\bar{X}$ como el *target*, e $\bar{Y}$ como el *output* real de la red, la función de pérdida está definida como:

$$\mathcal{L} = ||\bar{Y} - \bar{X}||_2^2 \tag{A.1}$$

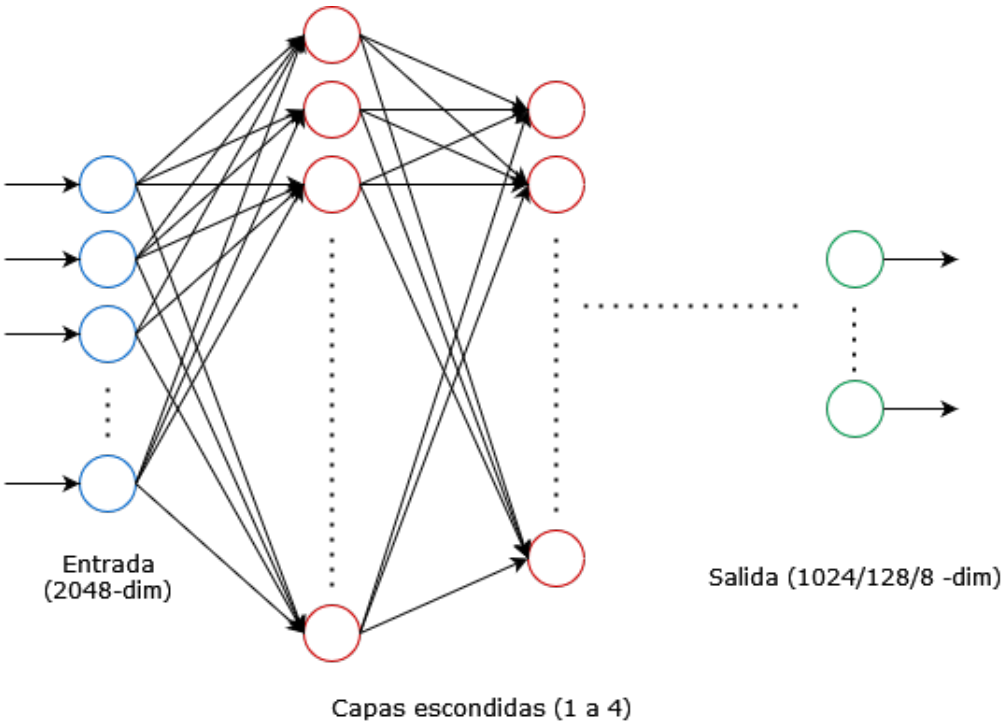


Figura A.1: Estructura general de red neuronal *fully-connected* para transformar *embeddings* visuales a textuales.

Los mejores modelos para cada dimensión se pueden observar en la tabla A.5, junto al *baseline* ResNet-50, en donde el modelo VTNN-N (*VisualTextNeuralNetwork-N*) corresponde al entrenado para generar una salida de N dimensiones.

Tabla A.5: Evaluación de la recuperación de imágenes usando *queries* de catálogo en el *dataset* de Pepeganga a través de los *embeddings* generados por una red neuronal *feed-forward fully-connected* entrenada, y el *baseline* ResNet-50 base.

Modelo	**Dimensiones**	**mAP (GC)**	**mAP (CT)**
ResNet-50	2048	87.13 %	67.83 %
VTNN-1024	1024	**89.94 %**	**69.52 %**
VTNN-128	128	84.66 %	48.74 %
VTNN-8	8	83.08 %	49.1 %

Por último, el mejor modelo para cada dimensión de salida fue evaluado con el mAP@20 sobre las 300 *queries* reales del mismo catálogo. Los resultados se pueden observar en la tabla A.6.

Tabla A.6: Evaluación de la recuperación de imágenes usando *queries* reales en el *dataset* de Pepeganga a través de los *embeddings* generados por una red neuronal *feed-forward fully-connected* entrenada, y el *baseline* ResNet-50 base.

Modelo	**Dimensiones**	**mAP (GC)**
ResNet-50	2048	**57.21 %**
VTNN-1024	1024	49.97 %
VTNN-128	128	45.07 %
VTNN-8	8	46.04 %

Si bien los resultados sobre las 100 *queries* de catálogo son prometedores, especialmente para el modelo VTNN-1024, al realizar una evaluación utilizando las 300 *queries* reales los resultados son considerablemente peores que el *baseline*, lo cual se debe a que la red neuronal no es capaz de generalizar, ya que se entrena utilizando únicamente imágenes de catálogo sin ruido, siendo inservible para casos de uso real. Por esta última razón, se decide no realizar más experimentos con este método.