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

MODELADO DE REDES NEURONALES DE GRAFOS PARA OPTIMIZAR LA RECUPERACIÓN SEMÁNTICA DE IMÁGENES BASADA EN CONTENIDO EN EL COMERCIO ELECTRÓNICO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

ALFREDO HUMBERTO ESCOBAR URREA

PROFESOR GUÍA:
JOSÉ SAAVEDRA RONDO

PROFESOR CO-GUÍA:
NILS MURRUGARRA LLERENA

MIEMBROS DE LA COMISIÓN:
SEBASTIÁN FERRADA ALIAGA
IVÁN SIPIRÁN MENDOZA

SANTIAGO DE CHILE
2024

Resumen

La búsqueda de productos en comercios electrónicos requiere de mayor precisión y eficiencia a medida que los catálogos crecen y la cantidad de consumidores aumenta. Una de las prestaciones que las aplicaciones de eCommerce proveen es la habilidad de tomar o subir una fotografía que retrate un objeto, para luego entregar una lista de productos del catálogo similares a este.

Una de las limitantes que presenta esta opción es la dificultad para intuir características de artículos potencialmente deseadas mas no plasmadas en una fotografía, tales como materiales, dimensiones, entre otras.

Se busca diseñar una metodología para entregarle a los clientes resultados más relevantes al momento de ellos hacer uso de estas opciones de búsqueda, retornando productos con características no fácilmente representables en una imagen, similares a aquellas de los productos más similares visualmente a la imagen provista.

Es por ello que se plantea un entrenamiento de vectores de características visuales de los productos de catálogos de eCommerce, en función de los vectores de características de texto de sus respectivas descripciones de artículo. Esto se hace mediante el uso de redes neuronales de grafos, en donde se define un grafo completo de similitud en donde hay un nodo por cada producto del catálogo, y con aristas cuyos pesos están determinados por la similitud entre las descripciones textuales de los artículos correspondientes.

A lo largo de este trabajo se proponen distintas alternativas de soluciones a problemáticas como lo son la definición de "similitud" entre vectores, o la metodología a seguir para seleccionar pares de nodos aleatorios a considerar para el análisis de espacios vectoriales en un momento dado del entrenamiento.

Los resultados obtenidos están cuantitativamente por debajo de otras metodologías ya existentes. Sin embargo, un análisis cualitativo demuestra que se tiene éxito al propagar características textuales de las descripciones de los artículos a lo largo del espacio vectorial correspondiente a las características visuales.

Tabla de contenido

1. Introducción

1.1. Contexto y Formulación del Problema

A lo largo de los últimos años, y en especial durante la pandemia por COVID-19, se ha visto un auge en la venta de productos por medios digitales [2]. Diversas tiendas *online* necesitan de estrategias informáticas para que sus clientes puedan encontrar productos potencialmente deseados. Es por esto que los motores de búsqueda de dichas tiendas requieren, más que nunca, una mayor precisión y eficiencia.

Una de las formas con que un motor de búsqueda puede abordar este problema es mediante la realización de búsquedas visuales. Para esto, la aplicación o sitio web de la tienda le solicita al usuario fotografías que plasmen un producto similar al deseado. Dichas imágenes pueden ser utilizadas con el fin de mostrar artículos similares al de la imagen.

Esto es particularmente útil cuando el usuario no se ve capaz de describir textualmente las características del producto buscado. Se facilita el *match* o coincidencia de artículos entre aquellos representados en fotografías entregadas por los mismos usuarios y productos similares a presentar.

Las metodologías empleadas para realizar esta tarea de búsqueda y recomendación solían basarse en extracción de características (o *features*) de bajo nivel en imágenes. Un ejemplo de esto es el histograma de orientaciones [6], que contabiliza la frecuencia de gradientes de dirección por cada "celda" (región de tamaño arbitrario) de la imagen a comparar.

En la actualidad, tras el desarrollo del área de *Deep Learning*, esta extracción de características (*feature extraction*) es llevada a cabo por redes neuronales convolucionales (CNN). Estas últimas generan vectores de *embeddings*, esto es, representaciones en espacios vectoriales de menor dimensionalidad que la de los *features* originalmente extraídos, generados de tal forma que conservan relaciones significativas entre los *features*.

Las tiendas suelen utilizar metodologías basadas en *content-based image retrieval* (CBIR) con el propósito de satisfacer la necesidad. Metodologías como ResNet [3] y CLIP [10] se encargan de buscar imágenes similares a las entregadas por el usuario, mediante la extracción de sus *features* y la posterior búsqueda de elementos con *embeddings* similares.

El problema que existe con la solución nombrada recae en la entrega de artículos de venta que, si bien poseen *embeddings* visuales similares –de acuerdo a las CNNs– a aquellas del producto de la imagen entregada, no pertenecen a la misma categoría del catálogo, o bien difieren en características que no son prominentes en una representación visual (como por ejemplo, el material del producto).

Las CNNs en cuestión suelen ser entrenadas con *datasets* de gran tamaño que, si bien permiten obtener resultados visualmente similares a la imagen ingresada (o *query*), no siempre son adecuados al resultado esperado, a causa de la generalización de *embeddings* de las redes neuronales.

Para adaptar de mejor manera esta metodología al contexto de búsqueda visual en eCommerce, se presenta la posibilidad de aprovechar no solo las imágenes de las fichas de cada producto, sino también el texto contenido en estas mediante vectores de *embeddings* que representen sus *features* respectivas.

Al establecer un espacio vectorial para *embeddings* de texto, en conjunto con el espacio para *embeddings* visuales, es posible mostrar productos cuyas descripciones textuales son similares a la del producto en la *query*, aunque puedan diferir visualmente. Estos artículos resultan más deseables para el cliente que un producto similar visualmente a la *query*, pero con características o categorías distintas.

Por ejemplo, si un usuario provee una imagen de su chaqueta marrón, una búsqueda basada en características exclusivamente visuales podría entregar productos como una polera manga larga marrón, mientras que al usuario le sería más útil que se le presente una chaqueta que no necesariamente coincida con el color marrón.

Ha sido propuesto por Guillermo Martínez un modelo reajustado de recuperación de imágenes denominado VETE (*Visual Embeddings boosted by TExts*) [5], basado en el uso combinado del espacio vectorial de *embeddings* de texto y el de *embeddings* visuales, que aumenta la semántica de la recuperación al aprovechar información visual y textual de productos en catálogos de comercio electrónico a través de modelos auto-supervisados.

VETE aprovecha el texto contenido en las fichas de los productos para encontrar artículos que coincidan en características no apreciables visualmente. Se define tanto un espacio vectorial de texto –en el cual residen *embeddings* correspondientes a las descripciones textuales de las fichas–, como un espacio vectorial visual –en donde encontramos *embeddings* de las imágenes de los artículos. Por cada producto del catálogo, se buscan los k productos con los *embeddings* textuales más cercanos al de este, y luego el modelo computa un *embedding* visual en base a un promedio ponderado de los *embeddings* visuales correspondientes a cada uno de los k productos obtenidos.

La técnica descrita busca generar una mejor agrupación de vecindades en el espacio visual, en donde las características no visuales puedan ser consideradas al momento de medir distancia o similitud entre los elementos de este espacio. VETE ha demostrado mejores resultados que búsquedas visuales sin este modelo [5]. Es por esto que se considera valiosa la oportunidad de profundizar y extender esta estrategia mediante metodologías que puedan aportar en la definición de las vecindades en los espacios de texto y visual.

Un trabajo pendiente tras la realización de esta solución consiste en la formalización de la localidad del espacio de texto mediante el uso de redes neuronales de grafos (GNNs). Podemos entender a las GNNs como transformaciones de grafos. Más específicamente, toman como *input* a elementos de grafos –como vértices, aristas y descriptores globales– con *embeddings* asociados, con el objetivo de aprender *embeddings* basados en estos elementos y retornarlos como *output*. Cabe mencionar que la estructura del grafo (*e.g.,* conectividad entre vértices) no es modificada tras esta transformación, sino los *embeddings* asociados a los vértices, aristas y/o descriptores globales.

Generalizar el espacio de texto de los catálogos de eCommerce mediante GNNs y representar a las descripciones similares de artículos (o de una misma categoría) como nodos vecinos o cercanos (y como nodos lejanos a las descripciones que difieren bastante), permitiría estudiar los resultados de un modelo que formaliza la estrategia de VETE, puesto que este computa promedios ponderados de *embeddings* obtenidos directamente de la extracción de *features* de imágenes.

En particular, el uso de redes neuronales de grafos permitiría un modelamiento más concreto de categorías de productos (e.g.: productos de aseo) y similitudes de productos intercategoría (e.g.: instrumentos de medición para contextos de construcción y contextos escolares), mediante la modulación de los pesos asociados a las aristas entre los nodos.

1.2. Objetivos

Objetivo General

Diseñar un modelo basado en VETE –con ajuste según el texto de los catálogos– haciendo uso de GNNs, que aumente la precisión de la búsqueda visual en catálogos de eCommerce.

Objetivos Específicos

1. Diseñar y entrenar un modelo de redes neuronales de grafos que permita ajustar *embeddings* del espacio visual de acuerdo a *embeddings* del espacio de texto correspondiente.
2. Plantear una función de pérdida y lograr que su respectivo valor de pérdida disminuya a lo largo del entrenamiento.
3. Diseñar funciones de probabilidad para la selección aleatoria de *batches* de pares de productos que permita obtener pares con puntajes de similitud muy altos y muy bajos (en relación al rango obtenido de puntajes de similitud), para luego ser utilizados en la función de pérdida.
4. Determinar la precisión del modelo planteado al realizar búsquedas con sets de evaluación, de acuerdo a la métrica *mean Average Precision* (mAP).
5. Determinar la precisión del modelo planteado al ver cuántas coincidencias de categorías globales se obtienen para determinados productos de ejemplo.

2. Estado del Arte

En esta sección se proveerá el marco conceptual de redes neuronales, de distintas metodologías para la obtención de vectores de *embeddings* tanto de texto como visuales, y de técnicas basadas en ajustes multi-modales – es decir, que aprovechan información vectorial de distinta naturaleza, como lo textual y lo visual. Se hace énfasis a VETE, modelo en el que se basa esta propuesta para la obtención de nuevos *embeddings* visuales según texto.

Puesto que el modelo a diseñar toma como base el ajuste de *embeddings* planteado en VETE, y busca producir resultados comparables con los de este, se hace necesario utilizar las mismas metodologías de generación de *embeddings* usadas por VETE. De este modo, las diferencias en rendimiento de precisión serán atribuidas al diseño del modelo propuesto en sí, y no a factores externos (como lo podría ser el uso de un generador de *embeddings* menos o más representativo de texto o imágenes).

2.1. Redes Neuronales

Una red neuronal es un modelo computacional basado en "neuronas" (Figura 1), unidades asociadas a un valor de "sesgo" y agrupadas en "capas". Se definen conexiones –con pesos asociados– entre las neuronas de una capa y la siguiente. Al entregar información como *input* a la primera capa de todo el set de capas de neuronas (es decir, la red), esta es operada con los valores de sesgo y peso de la red, desde aquellos en las primeras capas hasta las últimas, hasta entregar un *output*.

Las redes neuronales tienen asociadas funciones de pérdida, que reciben el *output* retornado por la red y entregan un valor de pérdida que determina qué tan distinto es este *output* de lo esperado. En función de este valor de pérdida, los valores de la red son ajustados, para nuevamente procesar la información y obtener un *output* distinto.

El proceso descrito busca disminuir el valor de pérdida obtenido, y así aprender valores de sesgo y peso que permitan obtener *outputs* similares a los deseados.

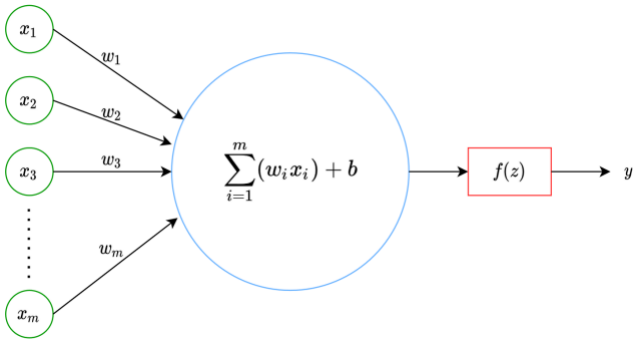


Figura 1: Estructura general de una neurona de una red neuronal

2.2. UMAP

UMAP [7] (*Uniform Manifold Approximation and Projection*) es una técnica que permite reducir la dimensión de vectores de *embeddings*, y así disminuir tiempos de ejecución y uso de memoria mientras se preserva la estructura del espacio vectorial.

Funciona mediante la construcción de una representación topológica (del *input*) "difusa" y de alta dimensionalidad, tras lo cuál se optimiza una representación de menor dimensionalidad de forma que conserve las relaciones globales y locales de los datos.

Construído desde un *framework* teórico basado en la geometría de Riemann y la topología algebráica, UMAP resulta un algoritmo práctico, escalable, y aplicable a datos reales.

2.3. Representación Vectorial de la Semántica

2.3.1. Word2Vec

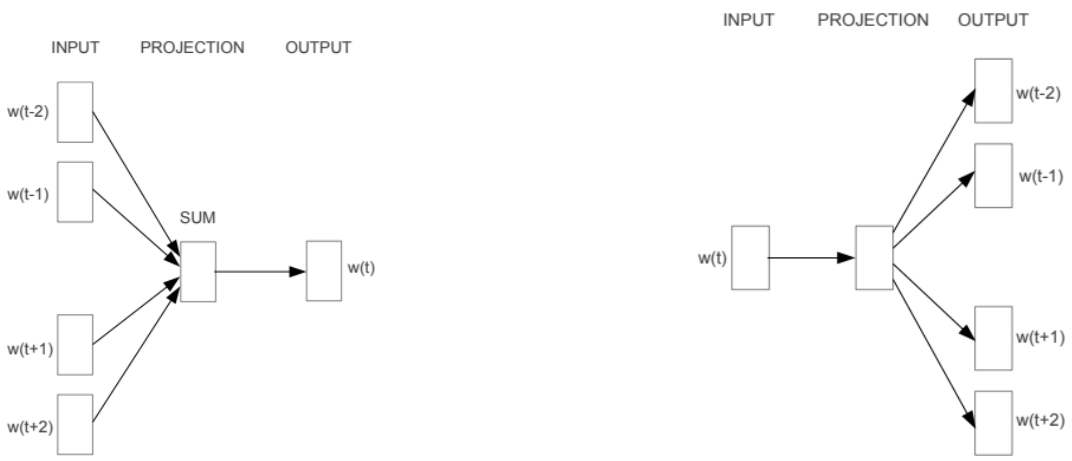
Word2Vec [8] (*Word to Vector*) es uno de los modelos que permite generar *embeddings* en base a texto.

Este modelo surge tras notar que el progreso con técnicas simples (como el uso de las secuencias de N-gramas) se estanca en tareas como el reconocimiento automático de voz y la traducción automática, sin importar cuántos datos adicionales se incorporen al entrenamiento. Word2Vec, entonces, busca entregar vectores de texto de alta dimensionalidad, comparables en base a la similitud de sus palabras (a diferencia de los modelos de N-gramas, en donde no hay noción de similitud de palabras), entrenables con datasets de muy alto tamaño.

Word2Vec propone dos nuevas arquitecturas de complejidad log-linear que manejan eficientemente grandes datasets tras remover capas no lineales escondidas. Estas arquitecturas, representadas en la Figura 2, serán descritas a continuación.

La primera de estas arquitecturas, CBOW (*Continuous Bag-of-Words*), intenta predecir palabras basándose en su "contexto", consistente de 4 palabras "históricas" y 4 palabras "futuras". La distribución continua de este contexto es la que le da el nombre a la arquitectura.

La segunda, *Continuous Skip-gram*, realiza la predicción opuesta: al tomar a una palabra como input, intenta predecir las palabras correspondientes a su contexto con cierto rango antes y después de esta.



(a) CBOW (b) Continuous Skip-gram

Figura 2: Representaciones de las arquitecturas de Word2Vec

2.3.2. RoBERTa

Al igual que Word2Vec, RoBERTa [4] (*Robustly Optimized BERT Pretraining Approach*) es un modelo de representación de lenguaje. Está basado en BERT [1] (*Bidirectional Encoder Representations from Transformers*), el cuál está diseñado para pre-entrenar representaciones bidireccionales profundas a partir de texto sin etiquetar, mediante el condicionamiento conjunto del contexto hacia la izquierda y derecha de las palabras en todas las capas.

Gracias a lo anterior, el modelo pre-entrenado puede ser ajustado con sólo una capa de output adicional, para así crear modelos avanzados para una gran variedad de objetivos, sin muchas modificaciones específicas a la tarea a realizar.

RoBERTa nace tras el análisis del efecto de hiperparámetros claves previamente ignorados en la formulación de BERT, tales como el tamaño de los *batches* y del dataset utilizado. Además, actualiza la máscara (previamente estática) de forma dinámica al entrenar al modelo con cada secuencia.

Gracias a estas modificaciones, RoBERTa logra igualar o sobrepasar la precisión tanto de BERT, como de modelos publicados posterior a este, en *benchmarks* como GLUE (General Language Understanding Evaluation), SQuAD (Stanford Question Answering Dataset) y RACE (ReAding Comprehension from Examinations).

2.4. Representación Vectorial de Imágenes

2.4.1. ResNet

La arquitectura ResNet [3] (*Residual Network*) es utilizada en tareas de visión por computadora, como lo es la generación de *embeddings* visuales. Esto es posible mediante un *framework* de entrenamiento residual profundo que permite que capas apiladas encajen en un *mapeado* residual, al utilizar como entrada de una capa a la salida de la capa previa.

En este contexto, un *mapeado* residual se refiere a una función residual $F(x)$ para un *input* (o "*identity mapping*") x, dado un $H(x)$ como el *output* deseado que una o más capas buscan aprender para el *input* x. La relación entre estos se define como:

$$F(x) + x = H(x) \tag{1}$$

$$F(x) = H(x) - x \tag{2}$$

Entonces, en lugar aprender directamente $H(x)$ para un conjunto de capas, un "bloque residual" (Figura 3) modela lo descrito en la ecuación 2 para su aprendizaje, permitiendo así "saltarse" capas.

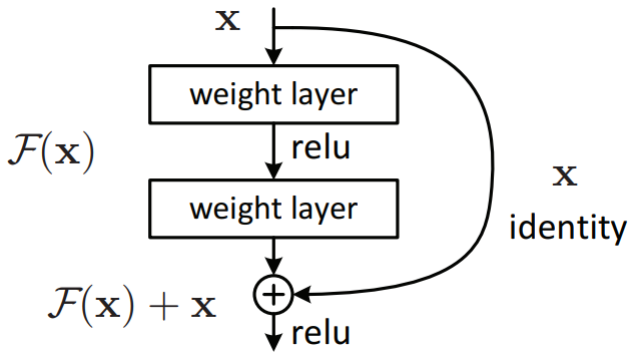


Figura 3: Bloque residual de ResNet

Esta técnica ayuda a solucionar problemas de saturación de exactitud y "explosión" (o desvanecimiento) del gradiente.

2.4.2. CLIP

CLIP [10] (*Contrastive Language-Image Pretraining*) es un modelo multi-modal que busca sortear las limitantes de los sistemas de visión por computadora, como el hecho de que son entrenados para predecir un set fijo de categorías predeterminadas, al hacer uso de lenguaje natural para aprender sobre las imágenes.

Se realiza un pre-entrenamiento de dos codificadores: uno para las imágenes, y el otro para texto. Tras esta etapa, ambos codificadores debiesen entregar *embeddings* similares al darles como input, de forma correspondiente, una imagen y una descripción textual de esta (Figura 4). Este modelo demuestra que el hacer coincidir pares de imágenes y texto que las describe es un método eficiente y escalable para obtener representaciones punteras de imágenes.

CLIP resulta práctico para su objetivo planteado, mas hasta el desarrollo de VETE no había sido probado para tareas de recuperación de imágenes.

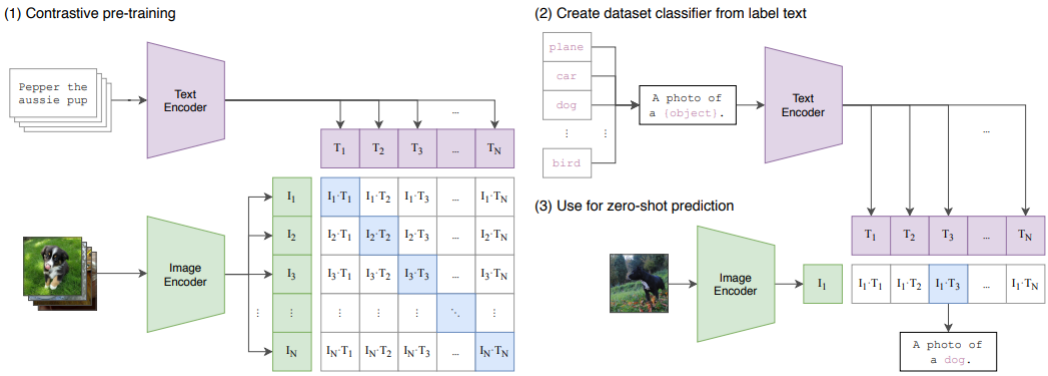


Figura 4: Resumen de la metodología implementada para CLIP

2.5. VETE

De forma similar a CLIP, VETE [5] (*Visual Embeddings boosted by TExts*) define un espacio vectorial para *embeddings* de texto y otro para los de imágenes (Figura 5). Este modelo tiene como objetivo mejorar la búsqueda visual de artículos en catálogos de eCommerce, por lo que esta vez los textos a entregar al modelo son las fichas de los productos representados en las imágenes correspondientes.

En el estudio de VETE se realizan experimentos con *embeddings* de texto obtenidos mediante Word2Vec, RoBERTa, entre otros, y con *embeddings* visuales obtenidos mediante ResNet-50 (esto es, con 50 capas de profundidad) y CLIP –usando, con este último, su *encoder* de texto, de tal manera de obtener espacios vectoriales similares. Se hace uso también de UMAP para la reducción de la dimensionalidad de los vectores de *embeddings*.

En base al ajuste de *embeddings* visuales en dos etapas (entrenamiento y búsqueda, detallado a continuación), se logran consistentemente mejores resultados que con los *baselines* definidos para la recuperación de imágenes. Dichos *baselines* consisten en la búsqueda sobre el espacio vectorial definido por los embeddings visuales entregados por ResNet-50 y por CLIP, sin un ajuste adicional dependiente de espacios vectoriales de embeddings de texto [5].

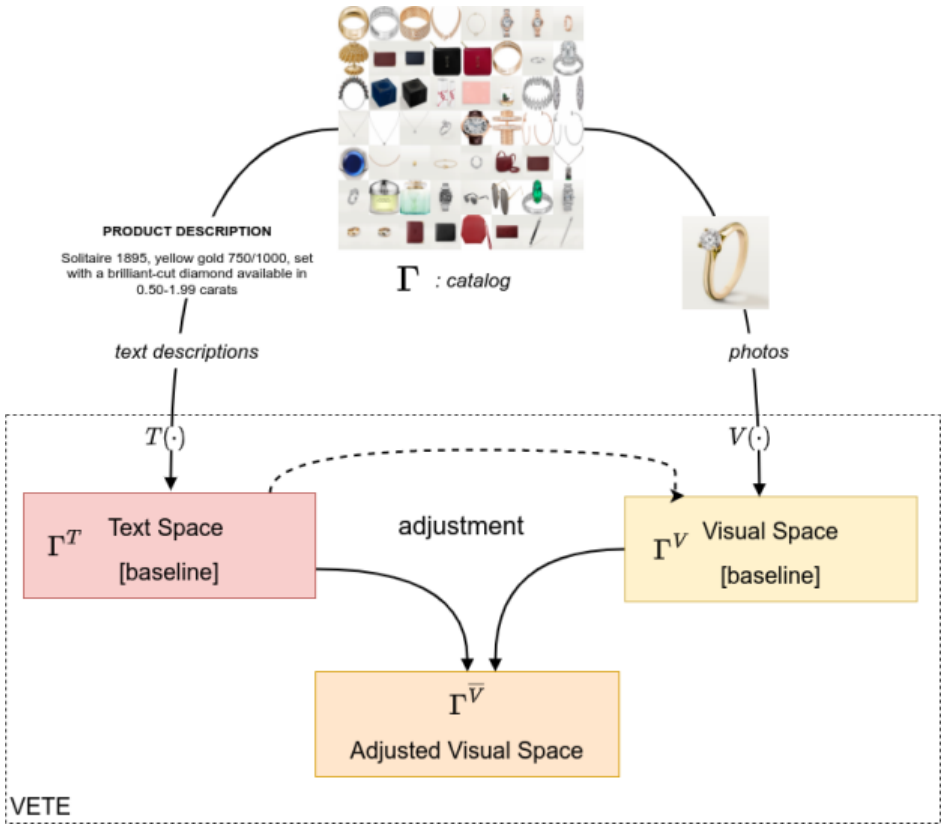


Figura 5: Representación de los espacios vectoriales definidos para VETE

2.5.1. VETE-A: Ajuste dentro del Catálogo

Este ajuste ocurre tras haber computado los *embeddings* visuales y de texto de todos los productos del catálogo. Por cada producto, se buscan sus k vecinos más cercanos de acuerdo a la similitud coseno entre sus *embeddings* de texto. Posteriormente se realiza un promedio entre el *embedding* visual del producto en cuestión y los *embeddings* visuales correspondientes a los k vecinos, obteniéndose así un nuevo vector de *embedding* visual. Este proceso es representado en la Figura 6.

En el estudio de VETE-A se realizaron pruebas con 3 tipos de promedios de *embeddings*: promedio simple con *moving average*, promedio ponderado con similitud coseno, y promedio ponderado con softmax.

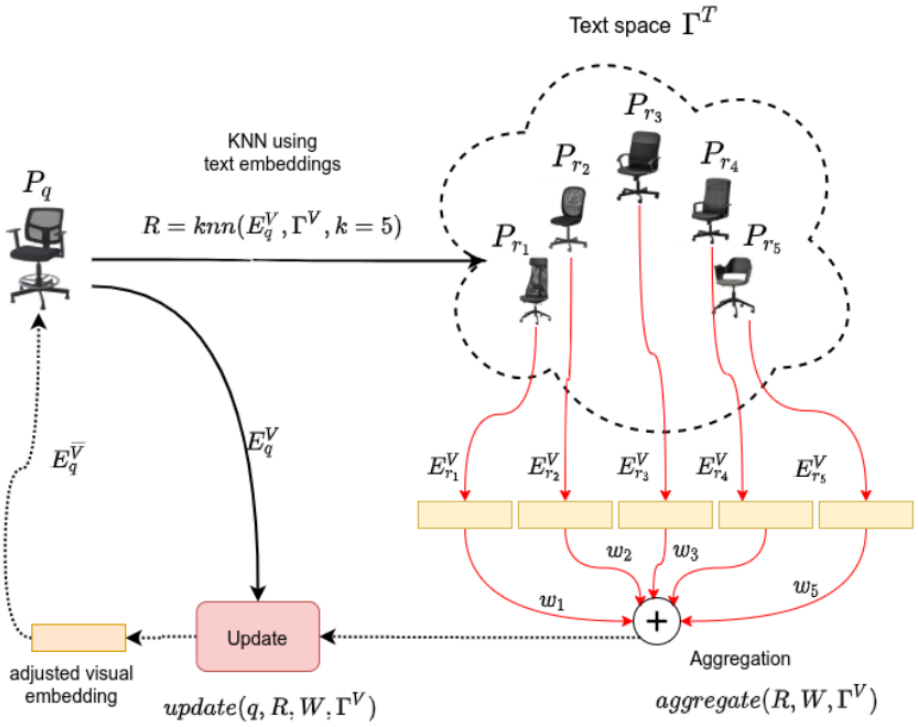


Figura 6: Representación del ajuste de vectores de *embeddings* del catálogo según VETE-A, usando $k = 5$ vecinos.

2.5.2. VETE-B: Ajuste de la *Query*

De forma similar al ajuste definido por VETE-A, para el *embedding* visual computado para cada *query* se seleccionan los 3 productos del catálogo más semejantes, de acuerdo a la similitud coseno con los *embeddings* visuales originales (no ajustados) de estos productos. Entonces, se realiza un promedio simple entre los *embeddings* visuales ajustados correspondientes a los 3 productos seleccionados, obteniéndose así un nuevo vector de *embedding* visual para la *query*.

2.6. Recuperación Multi-modal con GNNs

Se presenta otro modelo [9] que busca representar visual y semánticamente a las imágenes a recuperar. Se generan *embeddings* para texto e imágenes, ambos en un mismo espacio multi-dimensional. Esta vez, la relación entre los elementos visuales y los conceptuales es modelada mediante grafos, en los cuáles se define información valiosa mediante la vecindad entre nodos.

A su vez, se generan *embeddings* para los nodos del grafo mediante aprendizaje multi-modal, mediante el uso de redes neuronales de grafos (GNNs). Por cada nodo, tomamos información de su vecindad, y tras su agregación, actualizamos los embeddings de cada nodo (Figura 7).

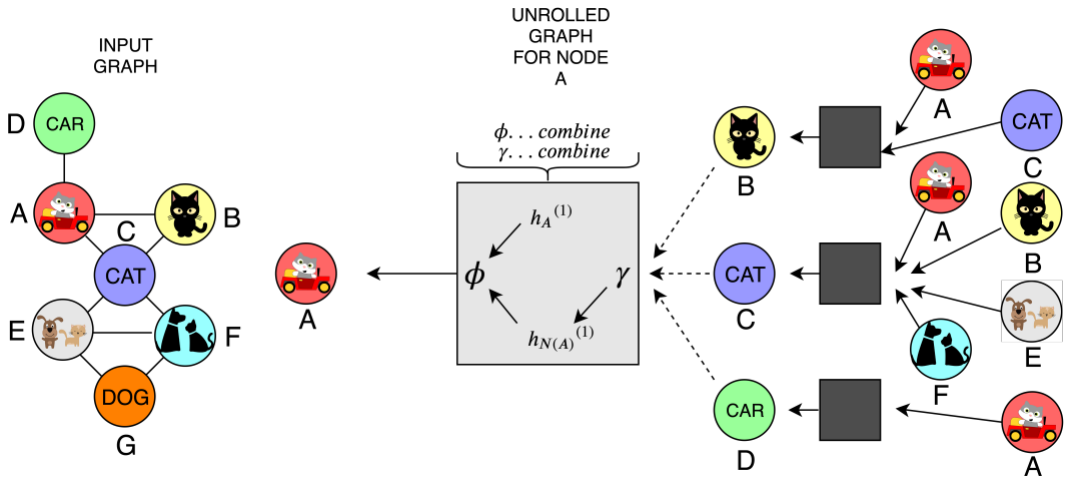


Figura 7: Grafo de entrada representado a la izquierda, y el proceso de agregación de información de la vecindad representado a la derecha

3. Solución

3.1. Descripción General

Se propone un modelo cuyo mecanismo es similar a lo planteado en VETE-A (Subsección 2.5.1), esto es, obtener nuevos *embeddings* visuales como sumas ponderadas de conjuntos –determinados por similitud de texto– de *embeddings* ya existentes.

Para esta propuesta (a diferencia de VETE), el espacio vectorial de *embeddings* a sumar es dado por una red neuronal de grafos (GNN). La obtención de los *embeddings* resultantes es dada por la multiplicación entre la matriz de adyacencia (que representa a las aristas del grafo) con la matriz de los *embeddings* visuales obtenidos por la red. Dado que el grafo es modelado según la similitud de los *embeddings* de texto de los productos, el proceso de ajuste resulta similar al planteado en VETE.

En la Subsección 3.2 se definen las características de los catálogos sobre los cuales se realizan los experimentos, además de las matrices de *embeddings* correspondientes a estos. En múltiples partes del proceso de entrenamiento se hace uso de matrices de similitud para determinar la semejanza entre todos los *embeddings*. Cuatro posibles metodologías de computación de estas matrices son descritas en la Subsección 3.4.

Uno de los usos de la matriz de similitud de *embeddings* de texto es el de describir la adyacencia del grafo a utilizar. En cada ciclo de entrenamiento, la matriz de adyacencia que define al grafo es multiplicada con la matriz de *embeddings* visuales de acuerdo a lo descrito en la Subsección 3.5, obteniéndose así nuevos *embeddings* visuales. Esta multiplicación sería el equivalente de la obtención de promedios simples (para el caso del grafo binario, Subsección 3.6) o de promedios ponderados (para el caso del grafo completo, Subsección 3.7) de *embeddings* visuales planteada en VETE-A (Subsección 2.5.1).

Posteriormente se computa un valor de pérdida para estos nuevos *embeddings* de acuerdo a las funciones descritas en las Subsecciones 3.6 y 3.7, según el tipo de grafo a utilizar.

Al utilizar redes neuronales de grafos, tras cada iteración de entrenamiento el valor del vector asociado a cada vértice es actualizado según los vectores correspondientes a los vértices vecinos. Así, tras la iteración n, los vectores de cada nodo ya habrán sido influenciados por todos aquellos nodos que estén a una distancia n o menor de este.

Lo anterior podría permitir que si en un grafo representáramos a los productos del catálogo como vértices, de tal forma que aquellos que estén en una misma categoría (o que pertenezcan a categorías similares) estén cerca en el grafo, entonces, al momento de actualizar sus valores estos estarán mucho más influenciados por los valores de productos similares que por cualquier otro producto del catálogo.

Tras el proceso, se obtendría un espacio vectorial de *embeddings* visuales ajustados de acuerdo al grafo basado en el espacio textual, como se ejemplifica en la Figura 8.

A diferencia de VETE-A, que obtiene nuevos *embeddings* con la multiplicación de *embeddings* visuales pertenecientes al mismo espacio vectorial que los *embeddings* computados para las *queries*, esta propuesta obtiene sus nuevos *embeddings* al ponderar *embeddings* visuales retornados por una red neuronal que describe un espacio vectorial distinto al de las *queries*. Debido a esto, se hace necesario el uso del ajuste de *queries* de VETE-B (Subsección 2.5.2), para así obtener nuevos *embeddings* visuales que describan a las *queries* en el nuevo espacio visual.

Esta extensión al modelo permitiría al cliente de la tienda poder encontrar resultados en línea de productos más relevantes en su búsqueda, en otras palabras, obtener una mayor precisión de las características más similares al artículo potencial que quiere adquirir.

Para esto, se planea implementar la solución en el lenguaje de programación Python. En específico, para el manejo de las redes neuronales de grafos se hará uso de la librería *Tensorflow* debido a su facilidad de uso, alto rendimiento y escalabilidad.

Será utilizado el mAP (*mean Average Precision*) de categorías globales –map(GC) por sus siglas en inglés– como puntaje para medir el éxito logrado y compararlo con el modelo original. También se graficará la evolución de mAP para árboles de categoría –mAP(CT)– y para subcategorías –mAP(SC).

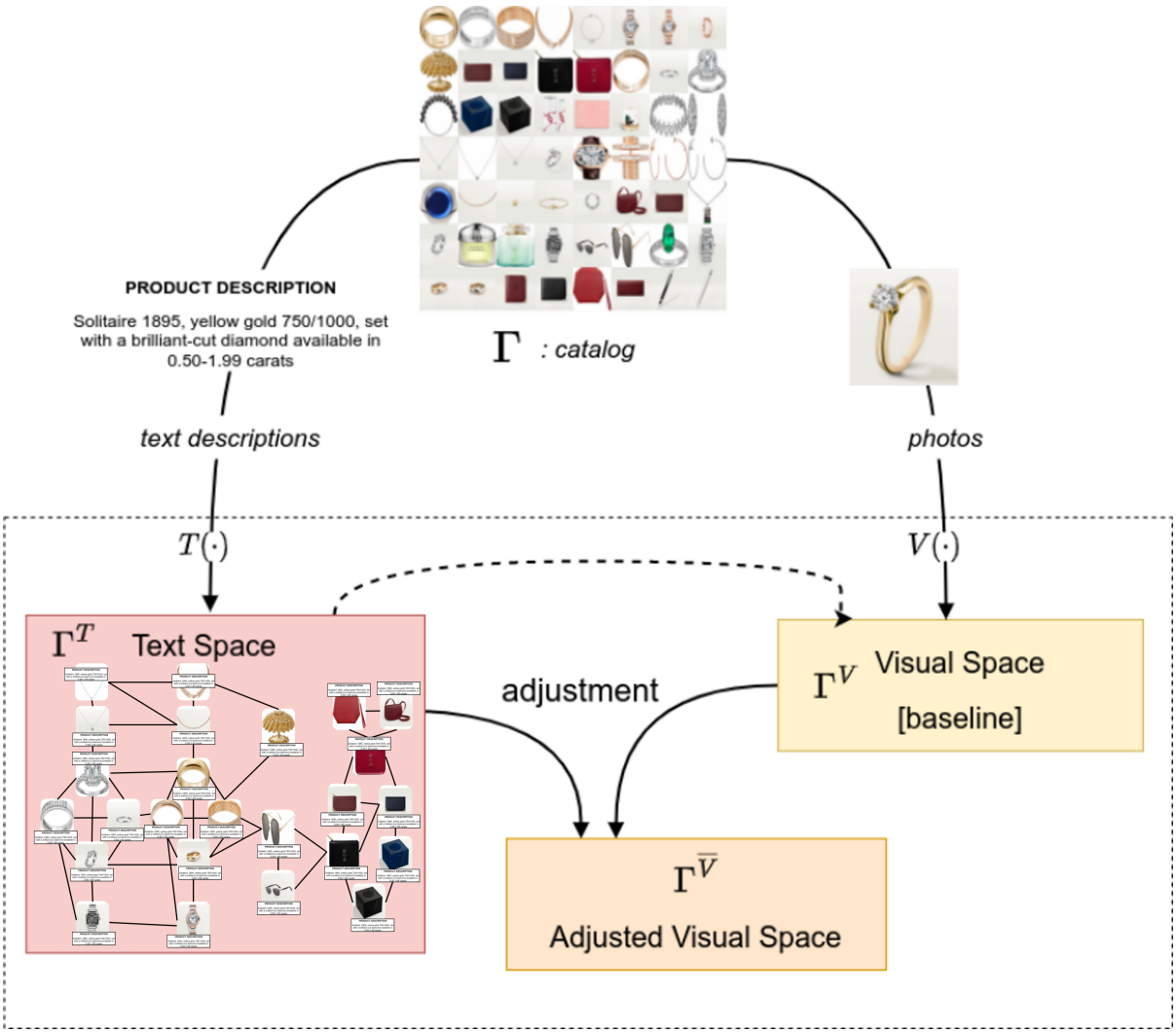


Figura 8: Esquema general de la solución propuesta, con Γ el catálogo de una tienda, Γ^T el espacio en donde se representan las descripciones (texto) de los artículos, y Γ^V el espacio representador de las imágenes. Se propone el modelado del espacio de texto por medio de redes neuronales de grafos, en donde descripciones similares de artículos sean representadas como vértices cercanos entre sí. Esta estrategia sería luego utilizada para generar un espacio visual ajustado $\Gamma^{\overline{V}}$.

3.2. Datasets

Se utilizarán los *datasets* de catálogos de comercios electrónicos que fueron usados para evaluar el desempeño del modelo VETE original (Pepeganga, IKEA y UNIQLO). En estos catálogos podemos encontrar productos de vestuario, juguetes, electrodomésticos, muebles, decoración, entre otros. Los datos fueron obtenidos por el autor de VETE, mediante el uso de las librerías de *Python* de web scraping *BeautifulSoup* y *Selenium*.

Para cada producto de cada dataset se ha rescatado una imagen, su descripción en texto y su árbol de categoría (*Category Tree* o CT). De este último se extrae su categoría global (*Global Category* o GC, el primer elemento en el árbol de categoría) y su subcategoría (*Subcategory* o SC, el tercer elemento en el árbol).

Sea, por ejemplo, un producto con un árbol de categoría "Juguetería/Vehículos/Carros", su categoría global sería "Juguetería", mientras que su subcategoría sería "Carros".

Para cada catálogo se han aislado 100 productos al azar. Posteriormente, se ha computado el *embedding* visual de la imagen correspondiente a cada uno de estos productos. Estos *embeddings* actúan como *queries* de la búsqueda en el espacio visual en ajuste, para permitir evaluar el desempeño del modelo a lo largo del entrenamiento.

3.2.1. Catálogos

UNIQLO

El primer catálogo sobre el cual se realizan pruebas es el de UNIQLO, debido a su reducido tamaño y, por consiguiente, menores tiempos de ejecución del entrenamiento correspondiente. Este catálogo se caracteriza por poseer ropa de mujer, hombre y niño. La cantidad de artículos de estas y otras categorías globales se presentan en la Tabla 1. Posee las siguientes características:

- 1425 productos.
- 5 categorías globales distintas.
- 281 árboles de categoría distintos.
- 51.7 MB de imágenes de productos, comprimidas mediante JPG.
- 1.45 MB de datos de texto sin comprimir.

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

Tabla 1: Cantidad de productos por categoría global (GC) en el catálogo de UNIQLO.

IKEA

Se tiene también el catálogo de IKEA, consistente de productos para el hogar como muebles, colchones y electrodomésticos. La cantidad de artículos de estas y otras categorías globales se presentan en la Tabla 2. Sus características son:

- 5606 productos.
- 18 categorías globales distintas.
- 602 árboles de categoría distintos.
- 187 MB de imágenes de productos, comprimidas mediante JPG.
- 2.48 MB de datos de texto sin comprimir.

Categorías globales	**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 accesories	34
Gardening & plants	30
Home improvement	25

Tabla 2: Cantidad de productos por categoría global (GC) en el catálogo de IKEA.

Pepeganga

Finalmente, se ha trabajado con el catálogo de Pepeganga. Este posee una mayor variedad de productos, incluyendo ropa, juguetes, objetos para el hogar, entre otros. La cantidad de artículos de estas y otras categorías globales se presentan en la Tabla 3.

Por limitaciones de memoria, se ha optado por trabajar con la mitad de la totalidad de los productos. Para no tener problemas por sesgo de selección de la muestra, se ha decidido ordenar el catálogo alfabéticamente según el nombre de los artículos y en base a este orden asignarle un número a cada producto. Posteriormente se seleccionan aquellos cuya enumeración corresponde a un número par. Se tienen entonces:

- 11279 productos a usar.
- 15 categorías globales distintas.
- 440 árboles de categoría distintos.
- 657MB de imágenes de productos, comprimidas mediante JPG.
- 21.1MB de datos de texto sin comprimir.

Categorías globales	**Cantidad**
Toy Store	2781
Clothes and Shoes	2105
Home	1722
Beauty	1200
Babies	1027
School	616
Sports	420
Pets	385
Food and Drinks	306
Furniture	254
Bedroom	158
Technology	120
Videogames and Consoles	94
Home Appliances	63
Christmas Decoration	28

Tabla 3: Cantidad de productos por categoría global (GC) en el catálogo de Pepeganga.

3.2.2. Embeddings

Se han creado representaciones vectoriales basadas en las descripciones textuales de los artículos mediante el modelo *RoBERTa-large* [4]. Este modelo fue pre-entrenado con 1 billón de pares de oraciones, obtenidos a través de diversos datasets que incluyen comentarios de Reddit, WikiAnswers, textos StackExchange, entre otros.

Se ha escogido a RoBERTa-large, debido a que demostró el mejor desempeño promedio (en contraste con el rendimiento obtenido con los modelos *MPNet-base*, *BERT* y *Word2Vec*) en las pruebas realizadas para VETE, en base a los valores de mAP@20(GC) y mAP@20(CT) resultantes con todos los catálogos estudiados [5].

Tomando en consideración que la dimensionalidad de cada vector de *embedding* generado por RoBERTa-large es de 1024, la matriz de *embeddings* de texto de UNIQLO resulta con dimensiones 1425×1024 (de tamaño $1425 \cdot 1024 \cdot 4\text{B} = 5{,}57\text{MB}$), la de IKEA con dimensiones 5606×1024 (de tamaño $5606 \cdot 1024 \cdot 4\text{B} = 21{,}9\text{MB}$) y la de Pepeganga con dimensiones 10960×1024 (de tamaño $10960 \cdot 1024 \cdot 4\text{B} = 42{,}81\text{MB}$).

Similarmente, para las representaciones vectoriales de las imágenes de los productos se ha utilizado el modelo *ResNet-50* pre-entrenado con datos de *ImageNet* [3]. Se ha escogido este modelo puesto que fue usado en el estudio de VETE, lo que permite comparar los resultados obtenidos en ese proyecto y el actual.

Considerando que la dimensionalidad de cada vector de *embedding* generado por ResNet-50 es de 2048, la matriz de *embeddings* visual de UNIQLO resulta con dimensiones 1425×2048 (de tamaño $1425 \cdot 2048 \cdot 4\text{B} = 11{,}13\text{MB}$), la de IKEA con dimensiones 5606×2048 (de tamaño $5606 \cdot 2048 \cdot 4\text{B} = 43{,}8\text{MB}$) y la de Pepeganga con dimensiones 10960×2048 (de tamaño $10960 \cdot 2048 \cdot 4\text{B} = 85{,}63\text{MB}$).

3.3. Trabajo Preliminar

En el semestre de primavera de 2022, se obtuvo el código correspondiente a VETE, y se ha configurado un ambiente de trabajo con los *packages* y *drivers* requeridos. Además, se han ordenado los directorios y archivos de los *datasets* según lo referenciado en el código (e.g., tomar y combinar columnas de archivos de valores separados por comas para obtener listas requeridas por VETE).

Posterior a esto, se ha estudiado la operatividad del código de VETE, y se han ejecutado con distintos argumentos para probar sus funcionalidades (e.g.: computar los *embeddings* de las imágenes, rescatar descriptores del texto en las fichas de los artículos utilizando distintos modelos, entre otras).

Luego, fue necesario elegir una librería para el manejo de las GNNs. El mayor requisito es la compatibilidad con paquetes y kits de desarrollo utilizados con VETE, en particular con:

- Kit de herramientas de CUDA, versión 11.3
- PyTorch, versión 1.12.1
- TensorFlow, versión 2.10.0

Se ha seleccionado a la librería PyG (*PyTorch Geometric*) debido a su alta cantidad de usuarios y, por consiguiente, mayor comunidad en lo que respecta a solución de problemas y guías. Posteriormente, el uso de esta librería sería descartado, y se optaría por utilizar funciones propias de TensorFlow, debido a la facilidad de uso, compatibilidad, y mayor cantidad de guías y ejemplos disponibles.

Se toma inspiración en el modelo mencionado en la Sección 2.6 al momento de definir la estructura de grafo. Cada nodo se asocia con un *embedding* de texto correspondiente a la ficha de un artículo del *dataset* indicado, estableciendo aristas entre este nodo y sus k nodos vecinos más cercanos, de acuerdo a la distancia euclidiana entre sus vectores de *embeddings* de texto correspondientes.

Una diferencia a recalcar entre VETE y la implementación del modelo de la Sección 2.6 es el único espacio vectorial de esta última para representar tanto texto como imágenes, lo que facilita la generación de un grafo que aloja *embeddings* de texto e imágenes. En ese grafo, son los nodos de *embeddings* de imágenes los que son conectados con sus k vecinos más cercanos, mientras que los nodos de *embeddings* de texto son conectados a los de las imágenes que describen.

Para el trabajo preliminar se ha decidido generar un grafo de similitud de *embeddings* de texto con los primeros 2500 de los 22557 artículos del catálogo de *Pepeganga*, estableciendo una arista entre cada nodo y los 6 vecinos con *embeddings* de texto más similares

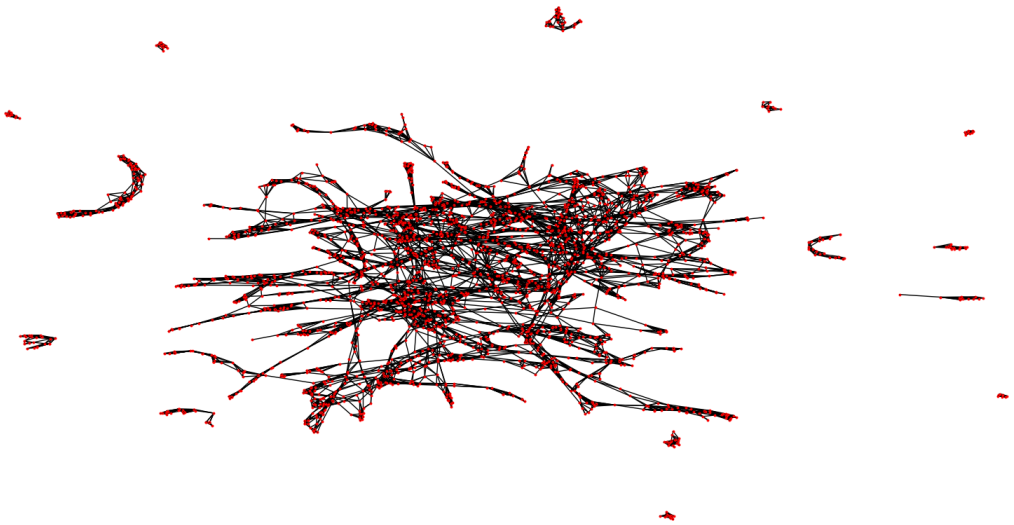


Figura 9: Representación del grafo de similitud de *embeddings* de texto de 2500 artículos del catálogo de *Pepeganga*. Los puntos rojos corresponden a los nodos de *embeddings*, y las líneas negras a las aristas entre vecinos.

En la Figura 9 se puede notar que se obtiene un grafo inconexo. Posteriormente, se optaría por una definición de grafo completo de similitud de *embeddings* de texto, esto es, todos los nodos son vecinos entre sí.

Al fijarse en una de las componentes conexas más pequeñas –con sólo 7 nodos– (Figura 10), se puede apreciar que se han conectado descripciones de productos similares correctamente; en este caso, poleras infantiles.

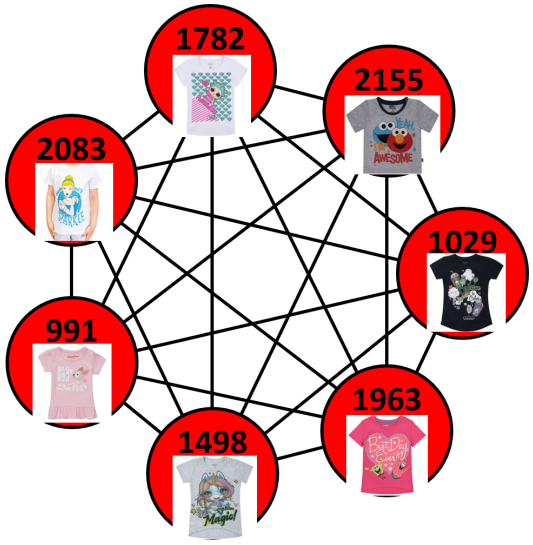


Figura 10: Representación de componente conexa del grafo de similitud de *embeddings* de texto obtenida. Los números indican el índice de cada producto. Imágenes de referencia incluidas.

Como referencia, a continuación se indican las descripciones de los artículos en la componente conexa:

- 991: *This beautiful pink short sleeve t-shirt, with a delicate front print, round neck and boleros will make your little girl feel very comfortable, its modern design is very easy to combine and will make her look wonderful.*
- 1029: *This beautiful short-sleeved and round neck T-shirt, with a delicate front print of one of the Poopsie characters will make your little one feel very comfortable, its modern design is very easy to combine and will make her look fantasy.*
- 1498: *This beautiful short-sleeved and round neck T-shirt, with a delicate front print of one of the Poopsie characters will make your little one feel very comfortable, its modern design is very easy to combine and will make her look fantasy.*
- 1782: *This beautiful short-sleeved T-shirt with A-silhouette, white background with a fun LOL front print and round neckline will make your little girl feel very comfortable, its modern design is very easy to combine with other clothes and will make her look wonderful, besides she will be delighted to take one of her favorite characters with her.*
- 1963: *This beautiful short sleeve t-shirt, with a delicate Sponge Bob print on the front and a round neckline will make your little girl feel very comfortable, its modern design is very easy to combine and will make her look wonderful and beautiful. Made of 100 % cotton.*
- 2083: *This beautiful short sleeve t-shirt, with a delicate front print of one of the girls' favorite Diseny princesses and a round neckline will make your little one feel very comfortable, The design of it is very easy to match and it will make you look wonderful.*
- 2155: *This beautiful short sleeve t-shirt, with a delicate front print and round neck will make your little one feel very comfortable, its modern design is very easy to combine and will make him look wonderful.*

3.4. Matrices de Similitud de *Embeddings*

En el entrenamiento, se hace uso extensivo de matrices de similitud de *embeddings* textuales y visuales, tanto al definir el grafo basado en características de los catálogos de los productos, como al calcular la pérdida de cada iteración de entrenamiento (proceso descrito más adelante). Las matrices de similitud son cuadradas de $n \times n$, con n la cantidad de productos del catálogo (o la cantidad de nodos del grafo), en donde el valor contenido en la posición i, j de la matriz corresponde a un puntaje de similitud entre la representación de vector de *embeddings* del producto i del catálogo, y el vector de *embeddings* del producto j.

Con el fin de estudiar el efecto de potenciar puntajes altos de similitud de manera exponencial, se ha propuesto que algunas de las metodologías de cálculo de puntajes basen sus resultados en la función $softmax$, la cual asigna valores mediante el uso de la función exponencial e^x, y hace que los valores de cada fila de la matriz de similitud sumen 1. Sea la matriz de similitud "sim":

$$softmax(sim_{i,j}) = \frac{e^{sim_{i,j}}}{\sum_{k=1}^{n} e^{sim_{i,k}}} \tag{3}$$

El uso de esta función resulta en matrices no simétricas ($sim_{i,j} \neq sim_{j,i}$). Para efectos de este trabajo, se ha determinado utilizar matrices de similitud simétricas. Lo anterior implica que la imagen de todo producto i sea tan visualmente similar a la del producto j, como la imagen del producto j lo es a la del producto i, y de igual manera entre las descripciones textuales de los artículos de ambos productos. Para establecer esta simetría, se ha definido la siguiente función:

$$simmetrical(sim_{i,j}) = sim_{i,j} + sim_{j,i} - sim_{i,j} \cdot sim_{j,i} \tag{4}$$

En total, se han definido cuatro modelos de matriz de similitud. De estos, dos basan su puntaje en la similitud coseno entre los vectores de *embeddings* de dos productos, y dos basan su puntaje en la distancia euclideana entre estos. Sea *embeddings* una matriz de $n \times d$, con d la dimensión de los *embeddings*, y sea $embeddings_i$ la fila i de esta matriz, correspondiente al vector de *embeddings* de características del producto i del catálogo, tenemos las siguientes propuestas de matriz de similitud:

- Similitud coseno:

$$cos\_sim_{i,j} = \frac{embeddings_i}{\|embeddings_i\|} \cdot \frac{embeddings_j}{\|embeddings_j\|} \tag{5}$$

- Similitud coseno con *softmax*:

$$soft\_cos\_sim_{i,j} = simmetrical(softmax(cos\_sim_{i,j})) \tag{6}$$

- Similitud cuadrada: sea $euclidean(i, j)$ la distancia euclideana entre los vectores de *embeddings* de i y de j:

$$sqrt\_sim_{i,j} = simmetrical\left(softmax\left(\frac{1}{e^{-euclidean(i,j)}}\right)\right) \tag{7}$$

- Similitud cuadrada normalizada según mínimos: sea $min\_euclidean_i$ el menor puntaje de distancia euclideana para la fila i:

$$normmin\_sqrt\_sim_{i,j} = simmetrical\left(softmax\left(\frac{1}{e^{-(euclidean(i,j)-min\_euclidean_i)}}\right)\right) \tag{8}$$

Se ha decidido no utilizar las metodologías planteadas en las Ecuaciones 7 y 8 basadas en distancias euclideanas, debido al alto costo de memoria requerido para su obtención. De aprovechar los operadores matriciales proveídos por la librería *Tensorflow*, sería necesario obtener una matriz de $n \times n \times d$, que represente a las $n \times n$ diferencias o distancias d-dimensionales entre todos los n vectores de *embeddings*, para luego obtener la norma o largo de estos $n \times n$ vectores de diferencia o distancia.

Al tomar en consideración que el catálogo más pequeño a entrenar (UNIQLO) contiene 1425 artículos, que la dimensionalidad de los vectores de *embeddings* visuales de *ResNet-50* es de 2048 y que se trabaja con valores numéricos de 32 *bits* (4 *bytes*), se requiere de un uso de memoria de:

$$1425 \cdot 1425 \cdot 2048 \cdot 4\text{B} = 16634880000\text{B} \approx 15{,}5\text{GB} \tag{9}$$

Alternativamente, es posible obtener las distancias euclideanas de forma iterativa en lugar de operar la totalidad de las matrices simultáneamente, o implementar técnicas de guardado en disco de valores computados. Sin embargo, el tiempo de ejecución de estas alternativas es significativamente elevado en comparación con el cálculo matricial de similitud de *embeddings* basado en similitud coseno.

Para el cálculo de la matriz de similitud coseno se requiere tan solo normalizar los vectores en la matriz *embeddings* de $n \times d$, y luego multiplicar la matriz resultante por su transpuesta. De este modo, cada uno de los n vectores normalizados es multiplicado por todos los demás, obteniendo así $n \times n$ puntajes de similitud. Debido a esto, este estudio se focaliza en metodologías basadas en similitud coseno.

3.5. Ciclo de Entrenamiento

Para cada ciclo de entrenamiento se requieren como *input* las siguientes matrices:

- $vis\_embeddings$, correspondiente a la matriz de *embeddings* visuales de dimensiones $n \times d$, con n la cantidad de productos del catálogo y d la dimensionalidad de cada *embedding* visual (2048 en el caso de *ResNet-50*).
- adj, correspondiente a la matriz –de dimensiones $n \times n$– de adyacencia del grafo. Esta es construída previo al entrenamiento, en base a la matriz de similitud de *embeddings* de texto de descripción del artículo. La matriz adj puede contener pesos tanto binarios como escalares, según el tipo de grafo a utilizar (de acuerdo a lo detallado en las Subsecciones 3.6 y 3.7, respectivamente).

Se busca como *output* una nueva matriz de *embeddings* visuales ajustados. Esta es denominada "$new\_vis\_embeddings$", con dimensiones $n \times u$, siendo u la dimensionalidad de cada uno de los n nuevos *embeddings*.

Al inicio de cada ciclo de entrenamiento, se le entrega como *input* la matriz de *embeddings* visuales a una capa lineal MLP de red neuronal densamente conexa "lyr", con una dimensionalidad de *output* u. Se realiza una transformación lineal a la matriz de *embeddings*, al ponderar sus valores por los pesos de la capa densa y luego sumándole los valores de *bias*, obteniendo así una matriz "$seq\_embeddings$" de dimensiones $n \times u$.

$$seq\_embeddings = lyr(vis\_embeddings) \tag{10}$$

Posterior a esto, considerando a cada fila de $seq\_embeddings$ como un vector asociado a cada nodo del grafo, y con el objetivo de agregar información de sus vecinos a cada fila de $seq\_embeddings$, multiplicamos esta matriz por la matriz de adyacencia adj.

$$ret\_embeddings = adj \cdot seq\_embeddings \tag{11}$$

De esta manera, cada fila i de la matriz resultante "$ret\_embeddings$" (de dimensiones $n \times u$), posee información propagada de distintas filas de la matriz $seq\_embeddings$, según los pesos definidos en la fila i de la matriz de adyacencia adj.

Después, se hace uso de la función de activación de unidad lineal rectificada ($ReLU$ por sus siglas en inglés) para introducir no-linealidad a los *embeddings* obtenidos en $ret\_embeddings$, y así finalmente conseguir nuevas representaciones u-dimensionales de los *embeddings* visuales de los nodos.

$$new\_vis\_embeddings = ReLU(ret\_embeddings) \tag{12}$$

A continuación, se computa la matriz de similitud de los nuevos *embeddings* visuales, utilizando la misma metodología empleada para el cálculo de la matriz de similitud de *embeddings* de texto usada para la obtención de la matriz de adyacencia.

Finalmente, se realiza el cálculo de la función de pérdida para esta iteración, se obtienen las variables observadas, se calculan los gradientes de acuerdo a estas variables y al valor de pérdida resultante, y se aplican estos a los valores de *kernel* y *bias* de la capa densa de red neuronal.

A continuación, se entrará en detalle sobre el cálculo de la función de pérdida, y más adelante se discutirá de pasos adicionales en el ciclo de entrenamiento, agregados posteriormente al cambiar la metodología de cálculo de pérdida.

3.6. Grafo Binario

Como se menciona en la Sección 3.3, inicialmente se planteó una matriz de adyacencia en donde, para cada nodo i, se consideraban vecinos los k nodos con menor distancia euclideana entre su *embedding* de texto asociado y el correspondiente a i. Por los motivos explicados en la Sección 3.4, se ha optado por realizar los cálculos de similitud de *embeddings* en base a similitud coseno en lugar de distancias euclideanas.

Con la implementación descrita anteriormente, se pueden establecer relaciones de vecindad unidireccionales: si el *embedding* de texto asociado a un nodo j es considerado uno de los k más cercanos al correspondiente a un nodo i, no necesariamente el *embedding* de i es uno de los más cercanos al de j, por lo que j es vecino de i pero no i de j.

Se optó por darle un carácter no dirigido al grafo, esto mediante la definición de nuevas aristas desde todo nodo j hasta todo nodo i tal que exista una arista desde i hasta j (es decir, si j es vecino de i, i también lo será de j).

Para esta primera versión de definición del grafo, se ha decidido calificar a cada par de nodos como "vecinos" y "no vecinos". Entonces, las aristas del grafo son representadas por una matriz de adyacencia de $n \times n$ con valores binarios. Al ser vecinos los nodos i y j en el grafo, se guardará un 1 en las posiciones (i, j) y (j, i) de la matriz de adyacencia. En caso contrario, se guardará un 0 en estas posiciones.

Sean:

- new_vis_sim la matriz de similitud visual computada para una iteración.
- $neighbours_i$ el conjunto de todos los j tal que $adj_{i,j} = 1$
- $sum\_esimilarity\_non\_neighbours_i = \sum_{j \in ([1,n] - neighbours_i)} e^{new\_vis\_sim_{i,j}}$

Se ha planteado la siguiente función de pérdida contrastiva:

$$loss\_A = \sum_{i=1}^{n} \frac{\sum_{j \in neighbours_i} \left(-log \frac{e^{new\_vis\_sim_{i,j}}}{sum\_esimilarity\_non\_neighbours_i} \right)}{|neighbours_i|} \tag{13}$$

Podemos asumir la influencia que cada par i, j de *embeddings* tendrá sobre la fracción anidada en el numerador según sus puntajes de similitud:

- Si con los vecinos j de i se tienen $new\_vis\_sim_{i,j}$ bajos, disminuirá el valor de la fracción anidada, aumentará el valor de $-log(\cdot\cdot\cdot/...)$, y aumentará el valor de pérdida final.
- Si con los vecinos j de i se tienen $new\_vis\_sim_{i,j}$ altos, aumentará el valor de la fracción anidada, disminuirá el valor de $-log(\cdot\cdot\cdot/...)$, y disminuirá el valor de pérdida final.
- Si con los no vecinos j de i se tienen $new\_vis\_sim_{i,j}$ altos, disminuirá el valor de la fracción anidada, aumentará el valor de $-log(\cdot\cdot\cdot/...)$, y aumentará el valor de pérdida final.
- Si con los no vecinos j de i se tienen $new\_vis\_sim_{i,j}$ bajos, aumentará el valor de la fracción anidada, disminuirá el valor de $-log(\cdot\cdot\cdot/...)$, y disminuirá el valor de pérdida final.

Lo anterior puede interpretarse como un aumento en el valor final de pérdida por cada:

- Par de nodos vecinos, correspondidos con *embeddings* visuales en entrenamiento que resultaron distintas.
- Par de nodos no vecinos, correspondidos con *embeddings* visuales en entrenamiento que resultaron similares.

Recordando que los nodos vecinos corresponden a aquellos productos con *embeddings* de texto similares, para disminuir el valor final de pérdida se tendrán que "acercar" pares de *embeddings* visuales correspondientes a productos con descripciones textuales similares, y "alejar" pares de *embeddings* visuales correspondientes a productos con descripciones textuales distintas.

3.7. Grafo Completo

Para observar el comportamiento de la evolución del campo visual al existir una influencia más global entre los elementos del campo de texto, y para evitar componentes aisladas en el grafo, se ha decidido definir un grafo completo. Esto es, que existan aristas entre todos los pares de nodos.

Se ha decidido que el peso asociado a cada una de las aristas sea directamente proporcional al puntaje de similitud de texto asociado al par de *embeddings* correspondientes a los nodos unidos por la arista. Inicialmente, esto se abordó al utilizar los mismos valores de la matriz de similitud de texto "$text\_sim$":

$$adj_{i,j} = text\_sim_{i,j} \tag{14}$$

De acuerdo a la Ecuación 11, se propagarían los valores opuestos a las filas de la matriz $seq\_embeddings$ correspondientes a los pares de nodos cuya arista en adj posea un peso negativo, lo cuál se puede dar al usar una matriz de similitud coseno, cuyo potencial rango es [-1,1]. Con lo anterior, efectivamente se estarían "alejando" los *embeddings* resultantes de productos con descripciones textuales distintas.

Tomando en consideración que las funciones de pérdida propuestas ya se encargan de "alejar" en el campo visual a los productos textualmente distintos, la propagación de valores opuestos descrita anteriormente podría ser interpretada como una "corrección excesiva".

Para que la información propagada desde los nodos "menos similares" sea ignorada o minimizada, los valores de $text\_sim$ son ajustados linealmente al rango $[0, 1]$. Sean $text\_sim_{a,b}$ y $text\_sim_{c,d}$ los valores mínimo y máximo de la matriz de similitud de texto, respectivamente, entonces para todo par i, j:

$$adj_{i,j} = \frac{text\_sim_{i,j} - text\_sim_{a,b}}{text\_sim_{c,d} - text\_sim_{a,b}} \tag{15}$$

Construyendo la matriz adj según lo descrito en la Ecuación 15 conseguiremos valores en el rango $[0, 1]$, con $adj_{a,b} = 0$ y $adj_{c,d} = 1$. Esto de forma independiente al rango de los valores de la matriz $text\_sim$.

3.7.1. Función de Pérdida para el Grafo Completo

Al cambiar la forma en que se construye el grafo, se busca también cambiar la perspectiva discreta de pares "vecinos" y "no vecinos" por la de pares "similares" y "distintos" al momento de calcular el valor de pérdida. Bajo esta nueva definición, todos los pares tienen el potencial de ser elegidos aleatoriamente bajo uno de estos dos conjuntos. En la Sección 3.7.2 se ahondará sobre la selección aleatoria de pares para cada iteración de entrenamiento.

Sean:

- s_{text} y s_{visual} funciones que otorgan puntajes en función de un puntaje de similitud de *embeddings*.
- r_{pairs} el conjunto de puntajes de similitud de *embeddings* de todos los pares de nodos elegidos aleatoriamente (tanto "similares" como "distintos").

$$loss\_B = \frac{\sum_{i \in r_{pairs}} max\left(0, s_{text}(i) \cdot log\frac{s_{text}(i)}{s_{visual}(i)}\right) + max\left(0, (1 - s_{text}(i)) \cdot log\frac{1-s_{text}(i)}{1-s_{visual}(i)}\right)}{|r_{pairs}|} \quad (16)$$

Dos distintas versiones de las funciones s son presentadas en las Secciones 3.7.3 y 3.7.4.

Podemos asumir la influencia que cada par i de *embeddings* tendrá sobre los sumandos del numerador de la fracción según sus puntajes de similitud:

- $s_{text}(i)$ y $s_{visual}(i)$ similarmente altos: el resultado de $log\frac{s_{text}(i)}{s_{visual}(i)}$ tenderá a 0, minimizando el sumando izquierdo, mientras que $1 - s_{text}(i)$ tenderá a 0, minimizando el sumando derecho.
- $s_{text}(i)$ alto y $s_{visual}(i)$ bajo: el resultado de $log\frac{s_{text}(i)}{s_{visual}(i)}$ será alto, aumentando el sumando izquierdo, mientras que $1 - s_{text}(i)$ tenderá a 0, minimizando el sumando derecho.
- $s_{text}(i)$ y $s_{visual}(i)$ similarmente bajos: $s_{text}(i)$ tenderá a 0, minimizando el sumando izquierdo, mientras que el resultado de $log\frac{1-s_{text}(i)}{1-s_{visual}(i)}$ tenderá a 0, minimizando el sumando derecho.
- $s_{text}(i)$ bajo y $s_{visual}(i)$ alto: $s_{text}(i)$ tenderá a 0, minimizando el sumando izquierdo, mientras que el resultado de $log\frac{1-s_{text}(i)}{1-s_{visual}(i)}$ será alto, aumentando el sumando derecho.

Igualmente que con la primera función de pérdida (Ecuación 13), lo anterior puede interpretarse como un aumento en el valor final de pérdida por cada:

- Par de *embeddings* de texto similares, correspondidos con *embeddings* visuales en entrenamiento que resultaron distintas.
- Par de *embeddings* de texto distintos, correspondidos con *embeddings* visuales en entrenamiento que resultaron similares.

Para disminuir el valor final de pérdida, se tendrán que "acercar" pares de *embeddings* visuales correspondientes a productos con descripciones textuales similares, y "alejar" pares de *embeddings* visuales correspondientes a productos con descripciones textuales distintas.

3.7.2. Selección de *Batches* Aleatorios para el Cálculo de Pérdida

En primer lugar, para seleccionar aleatoriamente el *batch* de pares para el cálculo de pérdida en la iteración, a cada par se le debe asignar 2 valores de probabilidad: una probabilidad de ser escogido dentro del subgrupo de los pares "similares", y otra probabilidad de ser escogido como par "distinto".

Estos valores de probabilidad estarán basados en el puntaje de similitud de texto, puesto que los puntajes de similitud visual son valores en entrenamiento cuya confiabilidad varía a lo largo de las iteraciones. Es importante recordar que para cada par de nodos distintos $i, j, i \neq j$, su puntaje de similitud se halla tanto en $text\_sim_{i,j}$ como en $text\_sim_{j,i}$, por lo que nos preocuparemos de omitir valores de similitud para la misma arista.

Sean:

- $half\_text\_sim$ un vector con los valores del triángulo superior de $text\_sim$ incluyendo su diagonal (esto es, $text\_sim_{i,j}$ en donde $i \leq j$).
- $half\_text\_sim_a$ el valor mínimo de $half\_text\_sim$.
- $half\_text\_sim_b$ el valor máximo de $half\_text\_sim$.

Se propone una versión inicial para el cálculo de probabilidades de pares similares ("$prob\_sim\_text$") y distintos ("$prob\_dis\_text$") en donde se ajustan linealmente los valores de $half\_text\_sim$:

$$prob\_sim\_text_i = \frac{half\_text\_sim_i - half\_text\_sim_a}{\sum_{j \in |half\_text\_sim|} (half\_text\_sim_j - half\_text\_sim_a)} \tag{17}$$

$$prob\_dis\_text_i = \frac{-half\_text\_sim_i + half\_text\_sim_b}{\sum_{j \in |half\_text\_sim|} (-half\_text\_sim_j + half\_text\_sim_b)} \tag{18}$$

Como se puede apreciar en los gráficos de distribución de pares aleatorios en la Sección 4.2.1, el abordar este problema con un ajuste lineal de puntajes de similitud resulta en distribuciones similares de pares seleccionados para ambos subgrupos, independientemente del valor de similitud de texto correspondiente al par.

Como buscamos distribuciones con mayores tendencias a los pares con los puntajes de similitud más altos y bajos, resultó necesario replantear el cálculo de probabilidades. En esta segunda versión, se define un parámetro τ con tal de realizar un ajuste lineal inicial de los valores de similitud al rango $[-\tau, \tau]$:

$$pre\_prob\_sim\_text_i = 2\tau \cdot \frac{half\_text\_sim_i - half\_text\_sim_a}{half\_text\_sim_b - half\_text\_sim_a} - \tau \tag{19}$$

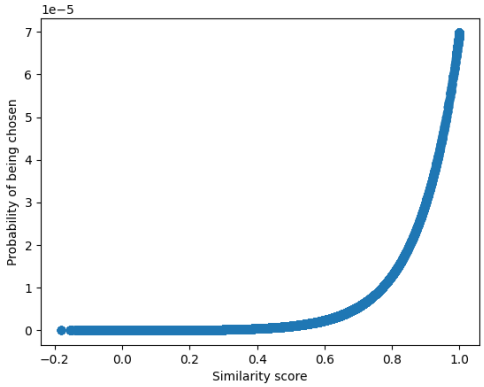
$$pre\_prob\_dis\_text_i = 2\tau \cdot \frac{-half\_text\_sim_i + half\_text\_sim_b}{half\_text\_sim_b - half\_text\_sim_a} - \tau \tag{20}$$

Tras el ajuste lineal al rango mencionado, se obtienen las probabilidades de acuerdo a la función exponencial:

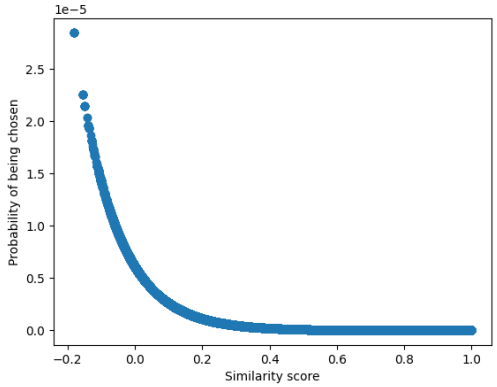
$$prob\_sim\_text_i = \frac{e^{pre\_prob\_sim\_text_i} - e^{-\tau}}{\sum_{j \in |half\_text\_sim|} (e^{pre\_prob\_sim\_text_j} - e^{-\tau})} \tag{21}$$

$$prob\_dis\_text_i = \frac{e^{pre\_prob\_dis\_text_i} - e^{-\tau}}{\sum_{j \in |half\_text\_sim|} (e^{pre\_prob\_dis\_text_j} - e^{-\tau})} \tag{22}$$

Al hacer uso de de estas funciones de probabilidad, con $\tau = 5$ y en función de los puntajes de similitud de texto, se obtienen los valores de probabilidad descritos por la Figura 11.



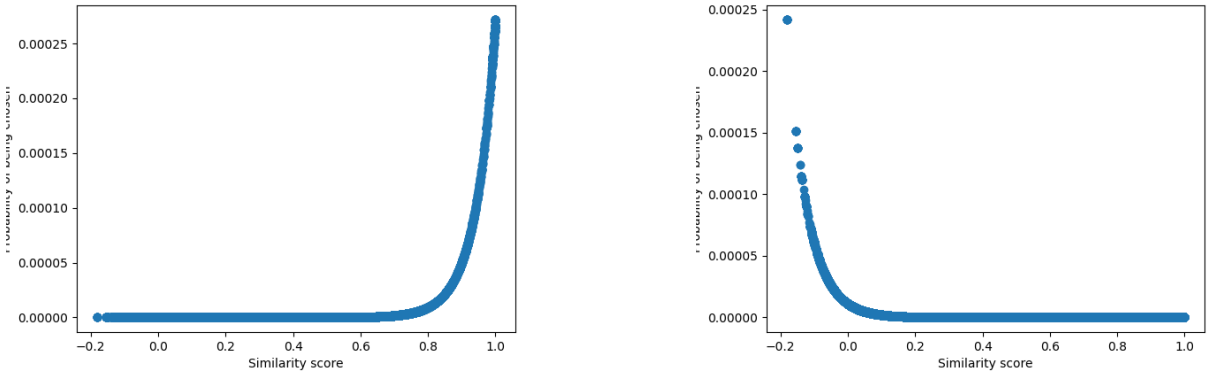
(a) Probabilidad de ser seleccionado dentro de los pares "similares"



(b) Probabilidad de ser seleccionado dentro de los pares "distintos"

Figura 11: Gráficos de probabilidad de selección de un par de nodos en función de su valor de similitud coseno (sin *softmax*) de texto correspondiente, con $\tau = 5$. Al usar similitud coseno con *softmax* se obtiene la misma curva, para un rango más reducido de puntajes de similitud (eje horizontal).

Alternativamente, se decide experimentar también con $\tau = 10$. Los respectivos valores de probabilidad se describen en la Figura 12.



(a) Probabilidad de ser seleccionado dentro de los pares "similares"

(b) Probabilidad de ser seleccionado dentro de los pares "distintos"

Figura 12: Gráficos de probabilidad de selección de un par de nodos en función de su valor de similitud coseno sin *softmax* de texto correspondiente, con $\tau = 10$.

De acuerdo a la estrategia descrita, con un valor τ dado, se selecciona aleatoriamente 100 pares de nodos "similares", y 100 pares de nodos "distintos", para un total de 200 pares a ser considerados al calcular la pérdida en la iteración.

3.7.3. Función s como Ajuste Lineal de Valores de Similitud

Como se puede intuir al observar el numerador de $loss\_B$ (Ecuación 16), en particular los elementos del tipo $s(i)$ y $1 - s(i)$, se espera que los valores retornados por la función s estén dentro del rango $[0, 1]$. Más precisamente, se requiere que estén dentro de un rango $[\alpha, 1-\alpha]$, con α tendiente a 0, para evitar indefinir las fracciones de la ecuación.

Se proponen entonces funciones s basadas en ajustes lineales de $text\_sim$ y new_vis_sim al rango $[\alpha, 1-\alpha]$, usando valores comunes de mínimos y máximos de similitud, para conservar la relación existente entre los *embeddings* de texto y visuales.

Sean:

- $min\_overall$ el mínimo entre todos los valores de $text\_sim$ y new_vis_sim.
- $max\_overall$ el máximo entre todos los valores de $text\_sim$ y new_vis_sim.
- $rand\_text\_sim$ un vector con los valores de $text\_sim$ correspondientes a los nodos escogidos aleatoriamente r_{pairs}, de acuerdo a lo descrito en la Sección 3.7.2.
- $rand\_vis\_sim$ un vector con los valores de new_vis_sim correspondientes a los nodos en r_{pairs}.

Entonces, los puntajes s para cada i en r_{pairs} se obtienen según las siguientes propuestas de funciones "s_linear":

$$s\_linear_{text}(i) = 2\alpha \cdot \frac{rand\_text\_sim_i - min\_overall}{max\_overall - min\_overall} - \alpha \tag{23}$$

$$s\_linear_{visual}(i) = 2\alpha \cdot \frac{rand\_vis\_sim_i - min\_overall}{max\_overall - min\_overall} - \alpha \tag{24}$$

3.7.4. Función s como Probabilidad de Selección de Par

De forma alternativa a las funciones s_linear, se propone el usar valores similares a los correspondientes a la probabilidad de selección de un par dentro de aquellos determinados como más "similares" (Ecuación 21).

Sean:

- $rand\_text\_sim_a$ el valor mínimo de $rand\_text\_sim$.
- $rand\_text\_sim_b$ el valor máximo de $rand\_text\_sim$.
- $rand\_vis\_sim_a$ el valor mínimo de $rand\_vis\_sim$.
- $rand\_vis\_sim_b$ el valor máximo de $rand\_vis\_sim$.

Entonces, los puntajes s para cada i en r_{pairs}, al sumarles $\alpha \to 0$ para evitar indefinir las fracciones de la Ecuación 16, son dados por las funciones s_prob, de acuerdo a las siguientes ecuaciones:

$$pre\_s_{text}(i) = 2\tau \cdot \frac{rand\_text\_sim_i - rand\_text\_sim_a}{rand\_text\_sim_b - rand\_text\_sim_a} - \tau \tag{25}$$

$$pre\_s_{visual}(i) = 2\tau \cdot \frac{rand\_vis\_sim_i - rand\_vis\_sim_a}{rand\_vis\_sim_b - rand\_vis\_sim_a} - \tau \tag{26}$$

$$s\_prob_{text}(i) = \frac{e^{pre\_s_{text}(i)} - e^{-\tau}}{\sum_{j \in |rand\_text\_sim|} (e^{pre\_s_{text}(i)} - e^{-\tau})} + \alpha \tag{27}$$

$$s\_prob_{visual}(i) = \frac{e^{pre\_s_{visual}(i)} - e^{-\tau}}{\sum_{j \in |rand\_text\_sim|} (e^{pre\_s_{visual}(i)} - e^{-\tau})} + \alpha \tag{28}$$

4. Evaluación

Para facilitar la comparación de resultados respecto al modelo reajustado de recuperación de imágenes original, se utilizarán las mismas métricas utilizadas en su evaluación de resultados correspondiente.

El rendimiento de la estrategia propuesta será evaluado en términos de *mean Average Precision* (mAP), métrica que indica la relevancia de los resultados entregados por un modelo. Por cada una de las 100 *queries* del set de evaluación de cada catálogo, se evaluará la mAP de acuerdo a los 20 resultados más relevantes (mAP@20). Se presentan gráficos con la evolución de la mAP para categoría global (GC), árbol de categorías (CT) y subcategoría (SC).

Al ser los vectores de *embeddings* visuales contenidos en $new\_vis\_embeddings$ (Ecuación 12) de carácter distinto a aquellos obtenidos directamente de *ResNet-50*, como lo son las *queries* de los sets de evaluación, se hace necesario el uso de la técnica de *ajuste de query* propuesta en VETE-B [5]. De este modo, se obtienen nuevos vectores de *embeddings* visuales capaces de representar a las *queries* y a la vez ser comparables con aquellos en $new\_vis\_embeddings$.

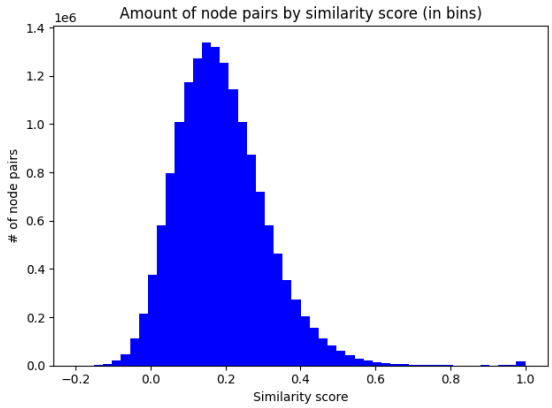
Se ha decidido evaluar el modelo usando la red ResNet-50 para la recuperación de características visuales del catálogo. Con el fin de facilitar el estudio del espacio visual de los *embeddings* en entrenamiento, se utiliza $u = 2048$ como la dimensionalidad del *output* de lyr (Ecuación 10), la misma dimensionalidad de los vectores de *embeddings* retornados por *ResNet-50*.

4.1. Distribución de Pares según Matriz de Similitud a usar

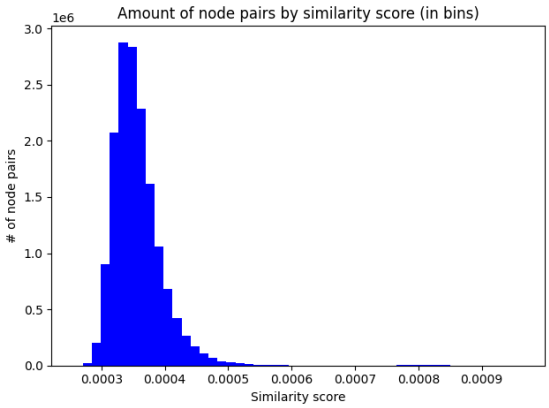
A continuación se presentan comparaciones entre las distribuciones de los pares de nodos de acuerdo a sus puntajes de similitud de texto coseno y coseno con *softmax*, para cada uno de los 3 catálogos a estudiar.

En las Figuras 13.b, 14.b y 15.b se puede apreciar que para los tres catálogos la distribución de pares tiende a valores de similitud bajos en relación al rango total de los valores. Esto permite explicar la distribución de *batches* aleatorios resultantes (que se muestran en las Subsecciones 4.2.1 y 4.2.2) con el uso de matrices de similitud coseno con *softmax*, en donde para ambos tipos de *batches* (de pares "similares" y "distintos") se obtienen pares con puntajes de similitud mayormente bajos.

IKEA



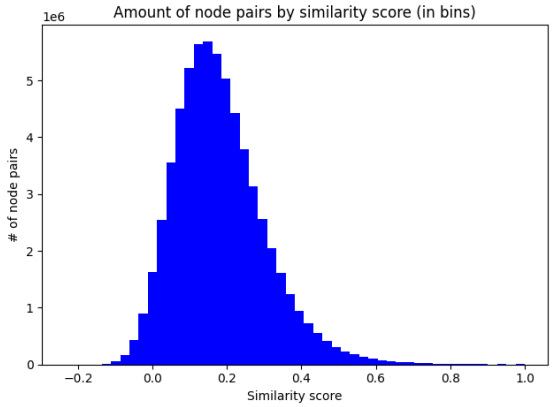
(a) Cantidad según similitud coseno.



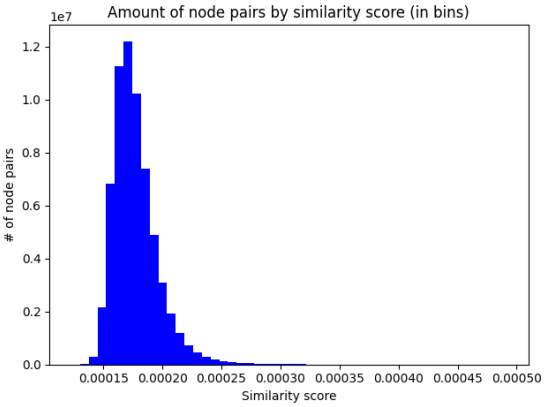
(b) Cantidad según similitud coseno con *softmax*.

Figura 13: Cantidad de pares de nodos.

Pepeganga



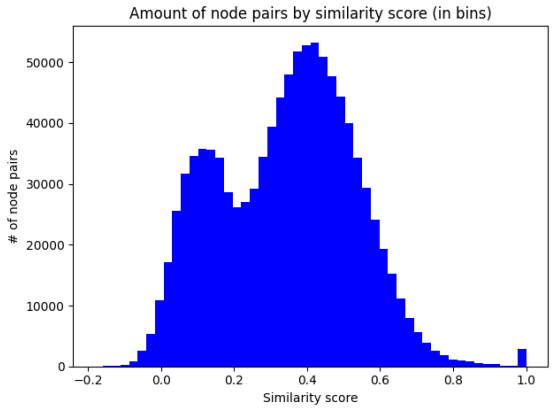
(a) Cantidad según similitud coseno.



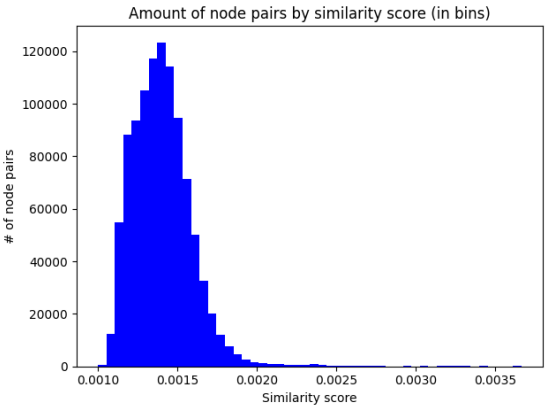
(b) Cantidad según similitud coseno con *softmax*.

Figura 14: Cantidad de pares de nodos.

UNIQLO



(a) Cantidad según similitud coseno.



(b) Cantidad según similitud coseno con *softmax*.

Figura 15: Cantidad de pares de nodos.

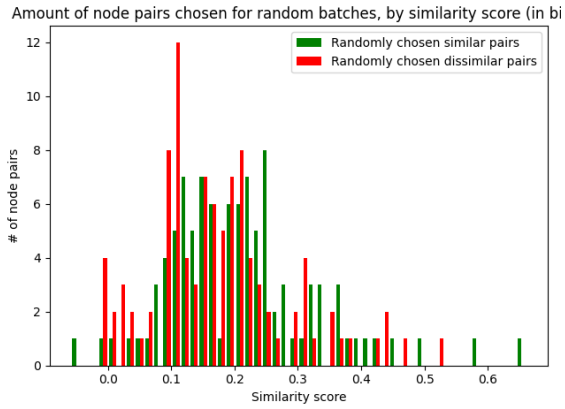
Como se puede apreciar en las Figuras 13, 14 y 15, al hacer uso de matrices de similitud coseno con *softmax* los pares tienden a obtener puntajes bajos de similitud en relación al rango completo de puntajes obtenidos.

4.2. Selección de *Batches* Aleatorios para *loss_B*

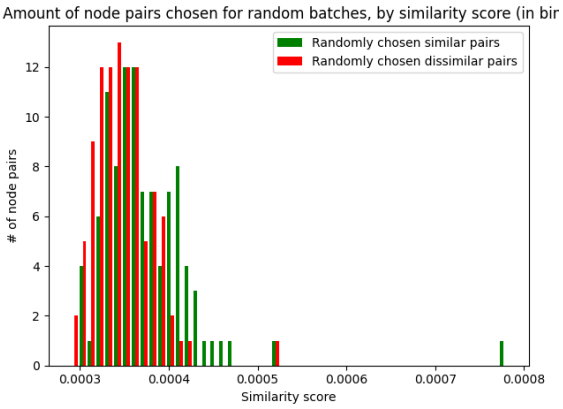
A continuación se presentan ejemplos de distribuciones de cantidades de pares de nodos por puntaje de similitud de texto, esto según la función de probabilidad utilizada (lineal o exponencial), tipo de matriz de similitud coseno (con y sin uso de *softmax*), para cada uno de los 3 catálogos a estudiar.

4.2.1. Con Selección Lineal

IKEA



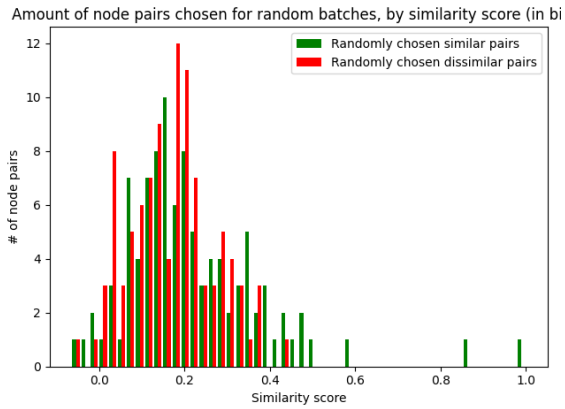
(a) Seleccionados según similitud coseno.



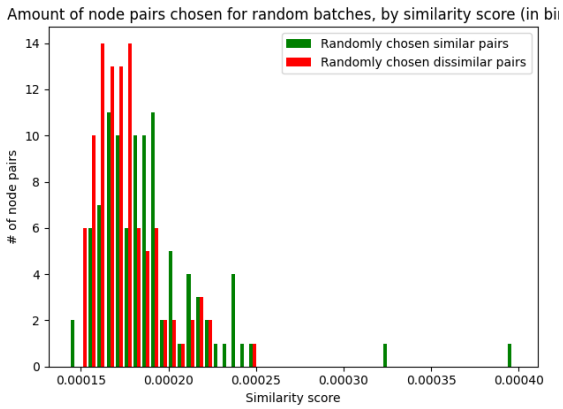
(b) Seleccionados según similitud coseno con *softmax*.

Figura 16: Ejemplo de cantidad de pares escogidos aleatoriamente según puntajes de similitud.

Pepeganga



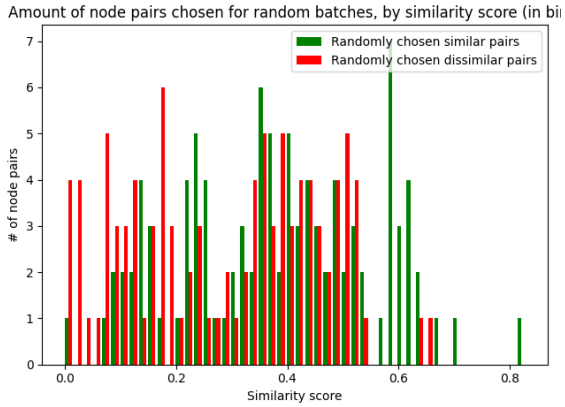
(a) Seleccionados según similitud coseno.



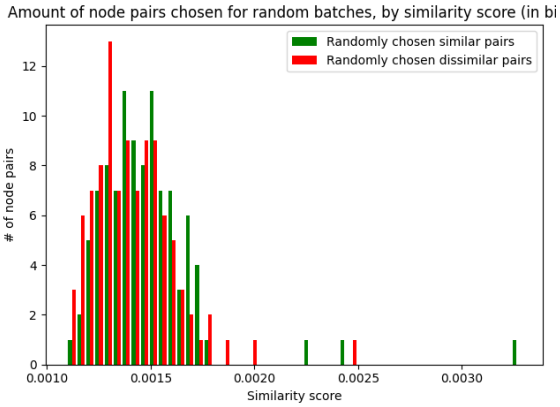
(b) Seleccionados según similitud coseno con *softmax*.

Figura 17: Ejemplo de cantidad de pares escogidos aleatoriamente según puntajes de similitud.

UNIQLO



(a) Seleccionados según similitud coseno.



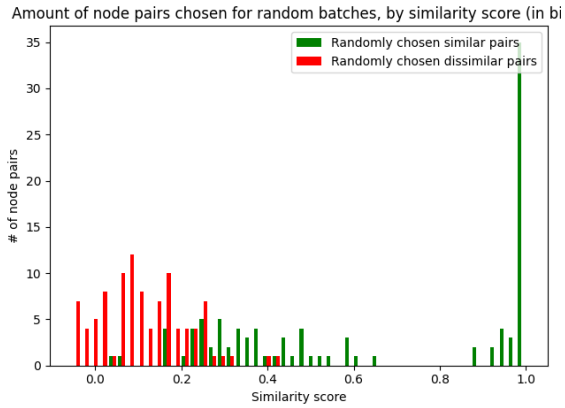
(b) Seleccionados según similitud coseno con *softmax*.

Figura 18: Ejemplo de cantidad de pares escogidos aleatoriamente según puntajes de similitud.

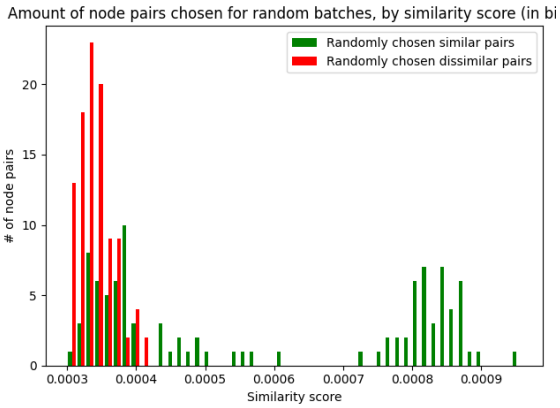
En las Figuras 16, 17 y 18 se puede apreciar que, al utilizar las funciones de probabilidad de las Ecuaciones 17 y 18 que ajustan linealmente los puntajes de similitud de texto se obtienen pares casi homogéneamente repartidos de acuerdo a las distribuciones de todos los pares del catálogo (ver Subsección 4.1), en lugar de conseguir pares característicos de los puntajes de similitud más altos y bajos.

4.2.2. Con Selección Exponencial, $\tau = 5$

IKEA



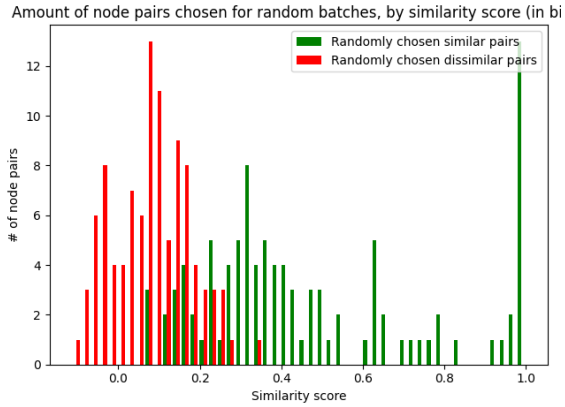
(a) Seleccionados según similitud coseno.



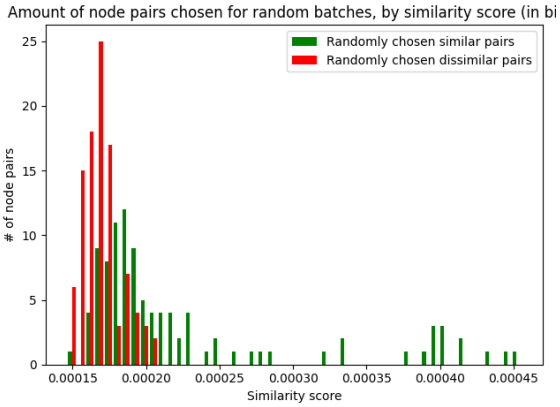
(b) Seleccionados según similitud coseno con *softmax*.

Figura 19: Ejemplo de cantidad de pares escogidos aleatoriamente según puntajes de similitud.

Pepeganga



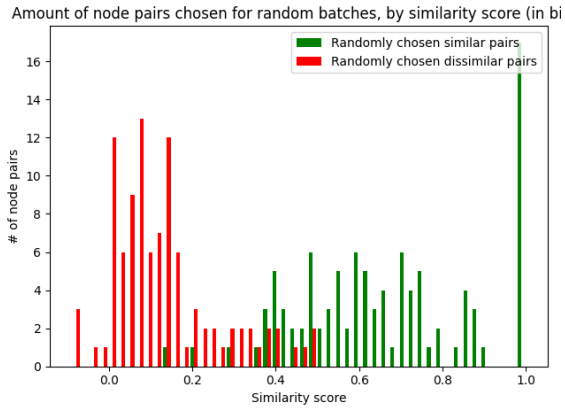
(a) Seleccionados según similitud coseno.



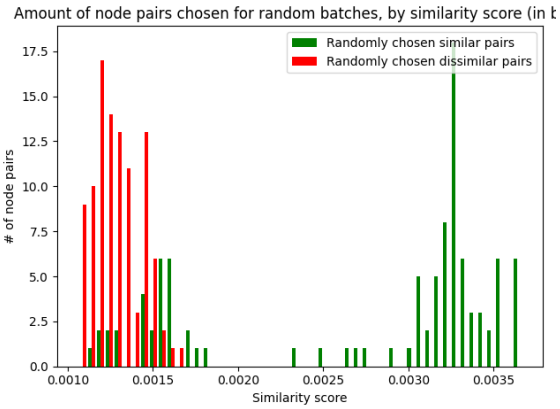
(b) Seleccionados según similitud coseno con *softmax*.

Figura 20: Ejemplo de cantidad de pares escogidos aleatoriamente según puntajes de similitud.

UNIQLO



(a) Seleccionados según similitud coseno.



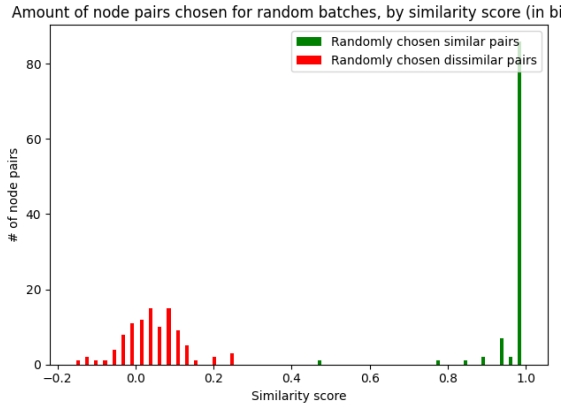
(b) Seleccionados según similitud coseno con *softmax*.

Figura 21: Ejemplo de cantidad de pares escogidos aleatoriamente según puntajes de similitud.

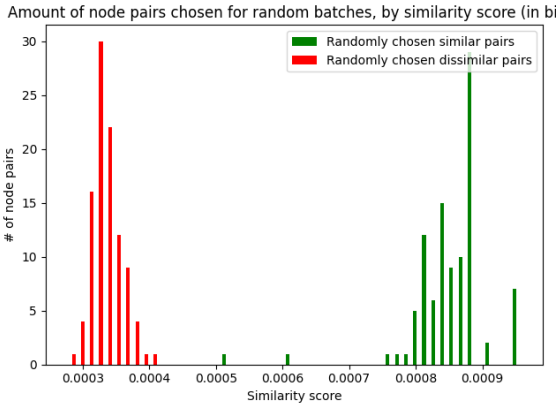
En las Figuras 19, 20 y 21 se puede apreciar que, al utilizar las funciones de probabilidad de las Ecuaciones 21 y 22 que ajustan exponencialmente los puntajes de similitud de texto con $\tau = 5$, se obtienen pares nos encontramos con problemáticas distintas de acuerdo al tipo de matriz de similitud utilizada. De usar matrices de similitud coseno (sin *softmax*) se obtienen aleatoriamente una cantidad considerable de pares de mediana similitud, por lo que no se cumple el propósito de entrenar en base a los pares más similares y más distintos. De usar matrices de similitud coseno con *softmax*, muchos de los pares de baja similitud son seleccionados dentro del subgrupo de los pares "similares" (según probabilidades otorgadas por la Ecuación 21), creado un desbalance entre los pares de bajo y alto puntaje a considerar al momento de calcular la pérdida en la iteración.

4.2.3. Con Selección Exponencial, $\tau = 10$

IKEA



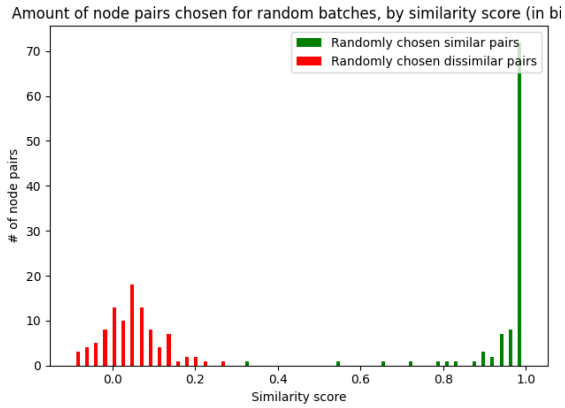
(a) Seleccionados según similitud coseno.



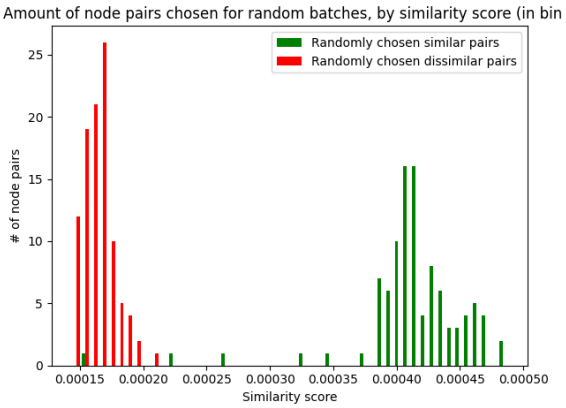
(b) Seleccionados según similitud coseno con *softmax*.

Figura 22: Ejemplo de cantidad de pares escogidos aleatoriamente según puntajes de similitud.

Pepeganga



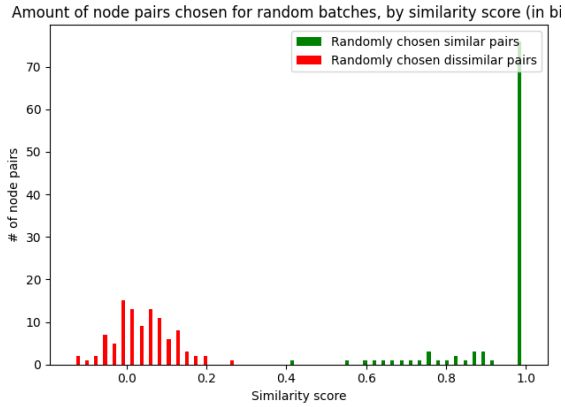
(a) Seleccionados según similitud coseno.



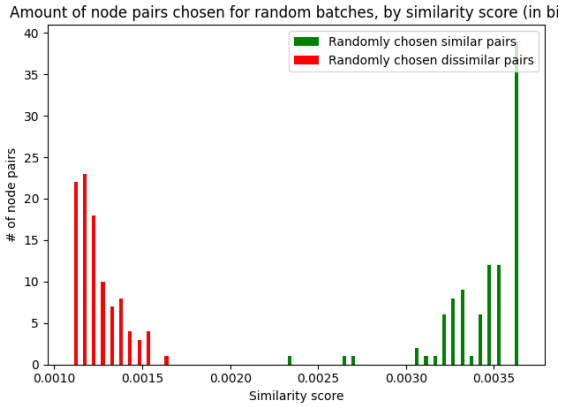
(b) Seleccionados según similitud coseno con *softmax*.

Figura 23: Ejemplo de cantidad de pares escogidos aleatoriamente según puntajes de similitud.

UNIQLO



(a) Seleccionados según similitud coseno.



(b) Seleccionados según similitud coseno con *softmax*.

Figura 24: Ejemplo de cantidad de pares escogidos aleatoriamente según puntajes de similitud.

En las Figuras 22, 23 y 24 se puede apreciar que, con $\tau = 10$, se obtienen *batches* más característicos de productos similares y distintos, por lo que se decide utilizar las funciones de probabilidad con ajuste exponencial descritas por las Ecuaciones 21 y 22, con $\tau = 10$, para la selección de *batches* aleatorios para cada iteración del ciclo de entrenamiento.

4.3. Marco de Experimentación

4.3.1. *Baselines* y Experimentos Planteados

Para la búsqueda de productos en los 3 catálogos propuestos y posterior evaluación de precisión se han planteado 2 *baselines*:

- Búsqueda sobre el espacio vectorial definido por los *embeddings* visuales entregados por *ResNet-50*, pre-entrenado con *ImageNet*, sin un ajuste adicional dependiente de espacios vectoriales de *embeddings* de texto.
- Búsqueda sobre el espacio vectorial definido por los *embeddings* visuales entregados por ResNet-50, ajustado según los vectores de *embeddings* de texto retornados por RoBERTa-large. Este ajuste corresponde al de mejor desempeño en cada escenario descrito en *VETE* [5], esto es, computando para cada producto el promedio simple de los *embeddings* visuales entre sus k productos (7 en el caso de UNIQLO, 5 en el caso de IKEA) más similares textualmente. Además, se ha hecho uso del ajuste de *queries* de VETE-B. No se incluye este *baseline* para el estudio de Pepeganga, debido a diferencias en el tamaño de los dataset usados por VETE y por este estudio.

Para el estudio de precisión del modelo descrito en este informe se plantean 4 escenarios. Todos ajustan el espacio de vectores de *embeddings* visuales entregados por ResNet-50 según los vectores de *embeddings* de texto retornados por RoBERTa-large por medio del uso de GNNs. Además, se ha hecho uso del ajuste de *queries* de VETE-B.

Se decide hacer uso de los grafos completos descritos en la Sección 3.7. Para la selección de *batches* aleatorios con propósito de cálculo de pérdida se ha decidido utilizar la estrategia de selección exponencial con valor de parámetro $\tau = 10$, de acuerdo a lo mencionado en la Subsección 4.2.3

Los 4 escenarios planteados son:

- Con uso de matrices de similitud coseno sin *softmax*, y función de pérdida calculada con el uso de la función $s\_linear(i)$, descrita en la Subsección 3.7.3.
- Con uso de matrices de similitud coseno sin *softmax*, y función de pérdida calculada con el uso de la función $s\_prob(i)$, descrita en la Subsección 3.7.4.
- Con uso de matrices de similitud coseno con *softmax*, y función de pérdida calculada con el uso de la función $s\_linear(i)$.
- Con uso de matrices de similitud coseno con *softmax*, y función de pérdida calculada con el uso de la función $s\_prob(i)$.

4.3.2. Hardware Utilizado

Se realiza la computación de la red neuronal y de tensores en una *GPU NVIDIA RTX A4000*, con 16GB de memoria de video. La conexión remota a este equipo fue proporcionada gracias al profesor Nils Murrugarra Llerena.

4.3.3. Métrica de Precisión

Para cada una de las 100 *queries* del set de evaluación para cada catálogo se obtienen los 20 productos más similares (según similitud coseno) de acuerdo al espacio de *embeddings* visuales correspondientes a cada escenario de experimentación, y se obtiene el valor de precisión promedio (AP@20, por las siglas de *average precision*) de acuerdo a la Ecuación 29.

$$AP@20 = \frac{\sum_{i=1}^{20} \left(relevant(i) \cdot \frac{\sum_{j=1}^{i} relevant(j)}{i} \right)}{\sum_{i=1}^{20} relevant(i)} \tag{29}$$

En donde $relevant(i) = 1$ si la categoría del i-ésimo producto retornado por la búsqueda coincide con la categoría de la *query*, y $relevant(i) = 0$ en el caso contrario.

De este modo, el mAP@20(GC) corresponde a un promedio simple entre los valores de AP@20 obtenidos para cada una de las 100 *queries* del set de evaluación, usando coincidencias de categorías globales para determinar los valores de $relevant(i)$. Similarmente, se usan coincidencias de árboles de categorías y de subcategorías para los cálculos de mAP@20(CT) y mAP@20(SC), respectivamente.

Se obtendría un 100 % de mAP@20(GC) si para cada uno de los 100 productos *query* del set de evaluación se rescatasen 20 productos pertenecientes a su misma categoría global.

Se considera un modelo exitoso aquel que logre superar el porcentaje de mAP@20(GC) obtenido con la búsqueda en el espacio vectorial de *embeddings* visuales de ResNet-50 sin ajuste. Idealmente se busca superar el mAP@20(GC) máximo obtenido por VETE para cada catálogo (puesto que este ya logra superar el *baseline* de ResNet-50 sin ajustes).

Los puntajes recopilados en las siguentes subsecciones (Tablas 4, 5 y 6) corresponden a los valores máximos de mAP@20(GC) alcanzados a lo largo de las 100 iteraciones de entrenamiento. Para todos los escenarios con los 3 catálogos, los valores máximos fueron alcanzados en la primera iteración.

4.4. Búsqueda en el Catálogo de UNIQLO

Configuración de ajuste	**mAP@20(GC)**
Baseline (ResNet-50)	70.66 %
Baseline (mejor resultado con VETE)	74.78 %
Similitud coseno, $s\_linear(i)$	60.60 %
Similitud coseno, $s\_prob(i)$	60.10 %
Similitud coseno+softmax, $s\_linear(i)$	59.37 %
Similitud coseno+softmax, $s\_prob(i)$	60.05 %

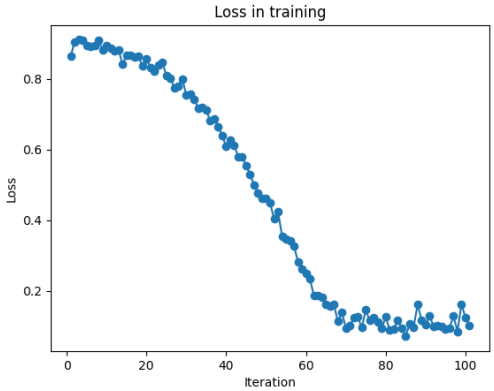
Tabla 4: Evaluación de la recuperación de productos del catálogo de UNIQLO, usando los dos *baselines* planteados junto con cuatro escenarios para el modelo propuesto

En la Tabla 4 se puede apreciar que en ninguno de los 4 escenarios propuestos el modelo fue capaz de superar la coincidencia de categorías globales obtenidas por los modelos *baselines*. Para el catálogo de UNIQLO, se obtuvo una mayor precisión al utilizar matrices de similitud coseno sin *softmax*, además de la función $s\_linear(i)$ dentro del cálculo de pérdida.

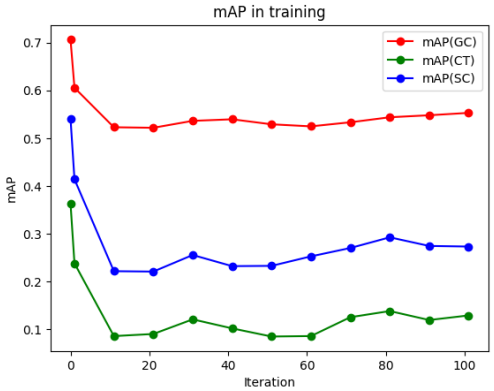
4.4.1. Evolución de mAP y Pérdida

Las Figuras 25, 26, 27 y 28 detallan la evolución de los valores de mAP@20 y del valor de pérdida a lo largo de las 100 iteraciones de entrenamiento.

Similitud coseno, *s_ linear(i)*



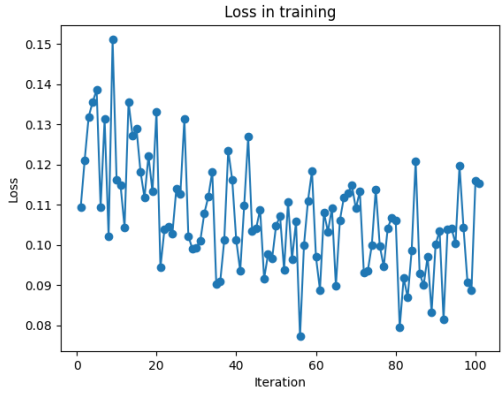
(a) Evolución del valor de pérdida a lo largo del entrenamiento.



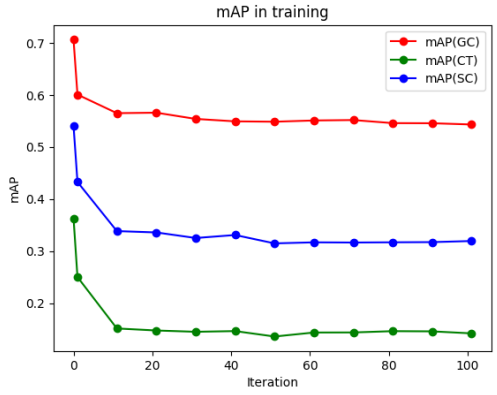
(b) Evolución de mAP@20, comenzando con el puntaje baseline de ResNet-50.

Figura 25: Evolución de la búsqueda en el catálogo de UNIQLO a lo largo de 100 iteraciones de entrenamiento usando similitud coseno y $s\_linear(i)$

Similitud coseno, *s_prob(i)*



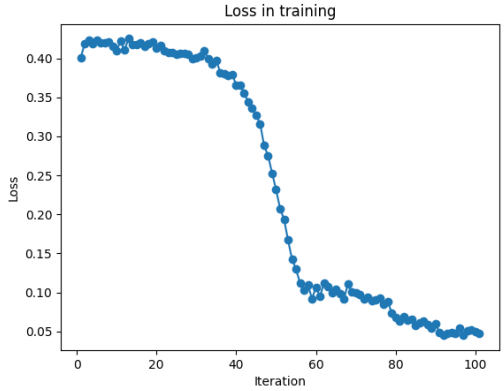
(a) Evolución del valor de pérdida a lo largo del entrenamiento.



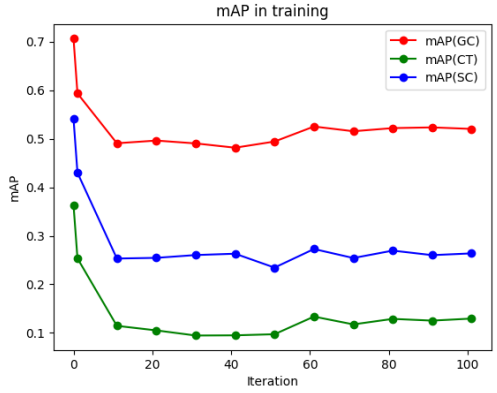
(b) Evolución de mAP@20, comenzando con el puntaje baseline de ResNet-50.

Figura 26: Evolución de la búsqueda en el catálogo de UNIQLO a lo largo de 100 iteraciones de entrenamiento usando similitud coseno y $s\_prob(i)$

Similitud coseno+softmax, *s_linear(i)*



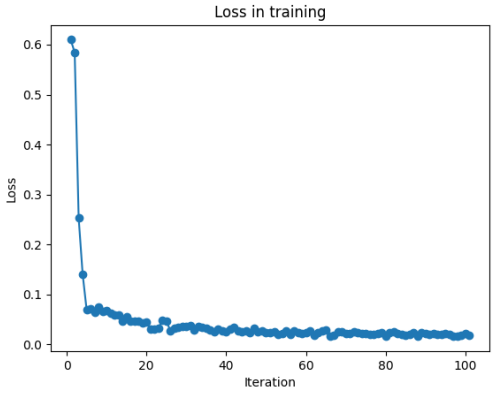
(a) Evolución del valor de pérdida a lo largo del entrenamiento.



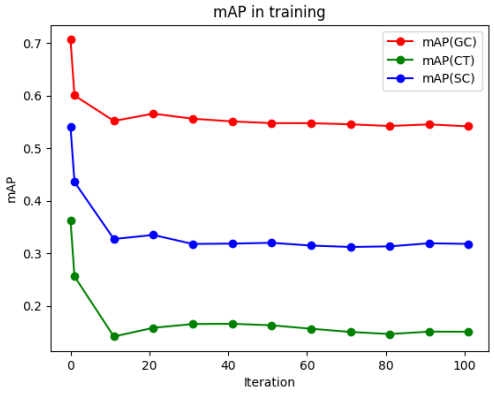
(b) Evolución de mAP@20, comenzando con el puntaje baseline de ResNet-50.

Figura 27: Evolución de la búsqueda en el catálogo de UNIQLO a lo largo de 100 iteraciones de entrenamiento usando similitud coseno con softmax y $s\_linear(i)$

Similitud coseno+softmax, *s_prob(i)*



(a) Evolución del valor de pérdida a lo largo del entrenamiento.



(b) Evolución de mAP@20, comenzando con el puntaje baseline de ResNet-50.

Figura 28: Evolución de la búsqueda en el catálogo de UNIQLO a lo largo de 100 iteraciones de entrenamiento usando similitud coseno con softmax y $s\_prob(i)$

Es posible notar la disminución instantánea de todos los valores de mAP. De acuerdo a la manera en que se ha construído la función de pérdida, se esperaría que a menor valor de pérdida se obtuviesen mayores valores de mAP@20, sin embargo, este no es el caso para ninguno de los 4 escenarios propuestos. A pesar de esto, en la Subsección 4.4.2 se presentan algunos ejemplos de los resultados en la práctica, y se realizan observaciones sobre su potencial relevancia con las *queries*.

Es de interés también la variación errática del valor de pérdida al usar matrices de similitud coseno sin *softmax* junto a la función s_prob (Figura 26.a), además de la rápida disminución de esta al usar matrices de similitud coseno con *softmax* junto a la función s_prob (Figura 28.a).

4.4.2. Ejemplos de Recuperación

Se presentan ejemplos de productos recuperados de acuerdo a los espacios vectoriales de *embeddings* visuales del *baseline* de ResNet-50 y de los 4 escenarios propuestos para este modelo, para 3 *queries* del set de evaluación.

a) Nombre del producto: MEN COLOR SOCKS
Categoría global: Men

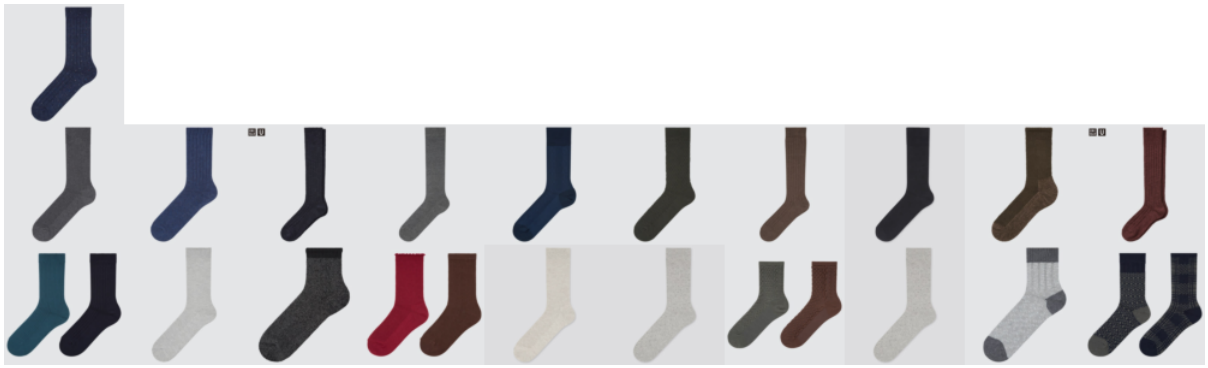


Figura 29: Recuperación de los 20 productos más similares a la *query* según el *baseline* ResNet-50 (sin entrenamiento). Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 75 % corresponde a la categoría global "Men", el 20 % corresponde a "Women" y el 5 % corresponde a "Kids".

Figura 30: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 25 % corresponde a la categoría global "Kids", el 55 % corresponde a "Men" y el 20 % corresponde a "Women".

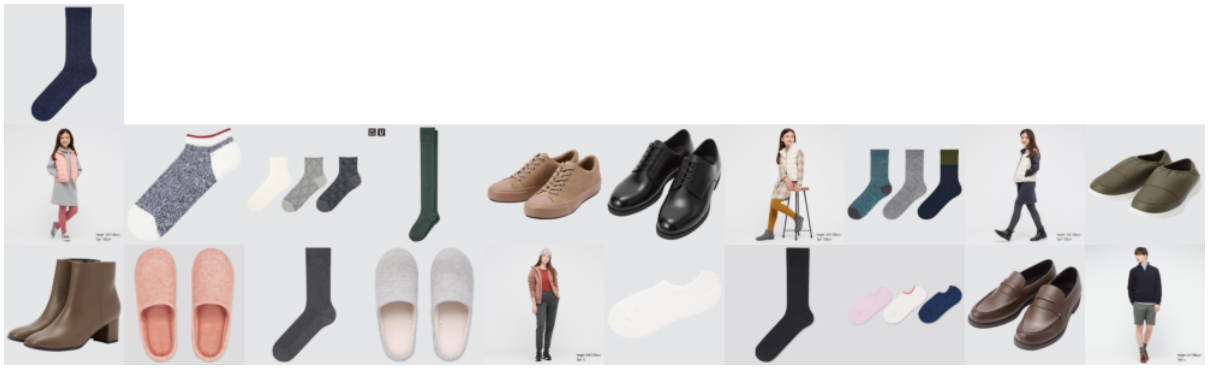


Figura 31: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 25 % corresponde a la categoría global "Kids", el 55 % corresponde a "Men" y el 20 % corresponde a "Women".

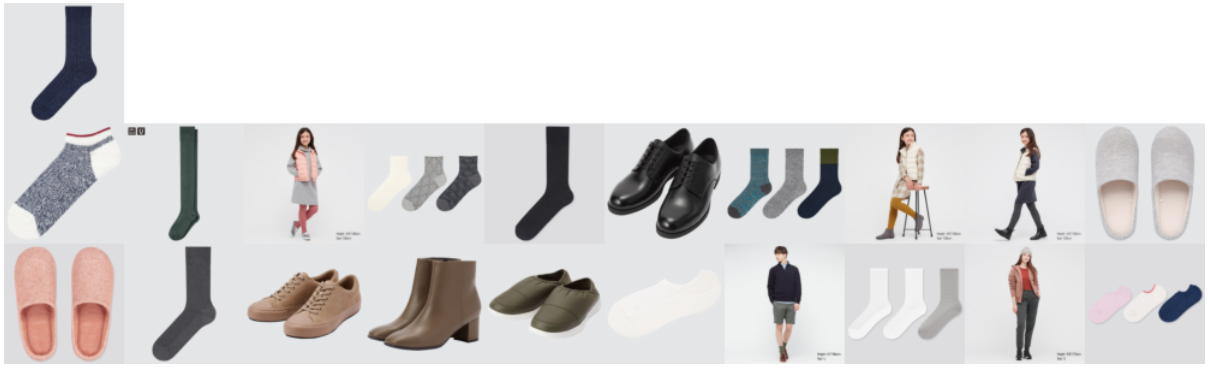


Figura 32: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 50 % corresponde a la categoría global "Men", el 25 % corresponde a "Women" y el 25 % corresponde a "Kids".

Figura 33: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 50 % corresponde a la categoría global "Men", el 20 % corresponde a "Women" y el 30 % corresponde a "Kids".

En los ejemplos de las Figuras 29, 30, 31, 32 y 33, se consigue el mayor porcentaje de acierto (55 %) al usar matrices de similitud coseno sin *softmax* (Figuras 30 y 31), pero ninguno de los 4 escenarios del modelo propuesto logra superar el 75 % de acierto de categoría global que se obtuvo con el *baseline* (Figura 29).

Si bien el modelo entrega productos relacionados al caso de uso de la *query* (en este caso, calzado), también hace que se deje de entregar productos mucho más pertinentes, como lo son los demás calcetines. En su lugar, son rescatados productos como chaquetas, *shorts* o *leggings* (correspondientes a las imágenes de personas de cuerpo completo).

Se puede argumentar que, para este producto, la similitud visual entre los productos originalmente retornados era una "ventaja" por la relevancia de estos, la cual es minimizada al agregar influencia de las descripciones textuales de los artículos.

b) **Nombre del producto:** HEATTECH-LINED FUNCTION GLOVES
Categoría global: Men

Figura 34: Recuperación de los 20 productos más similares a la *query* según el *baseline* ResNet-50 (sin entrenamiento). Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 15 % corresponde a la categoría global "Women", el 10 % corresponde a "Kids", el 35 % corresponde a "Men" y el 40 % corresponde a "Baby".

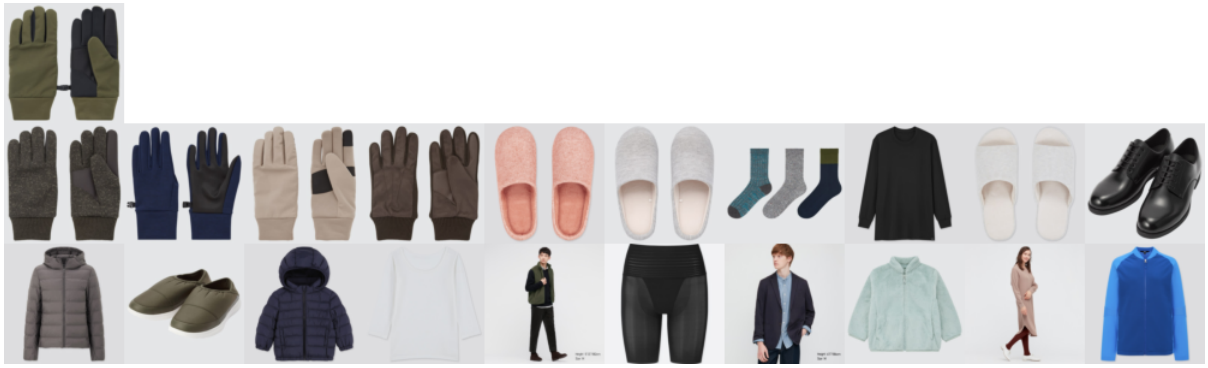


Figura 35: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 55 % corresponde a la categoría global "Men", el 25 % corresponde a "Women", el 5 % corresponde a "Kids" y el 15 % corresponde a "Baby".

Figura 36: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 55 % corresponde a la categoría global "Men", el 10 % corresponde a "Women", el 10 % corresponde a "Kids" y el 25 % corresponde a "Baby".

Figura 37: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 55 % corresponde a la categoría global "Men", el 25 % corresponde a "Women", el 10 % corresponde a "Kids" y el 10 % corresponde a "Baby".

Figura 38: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 60 % corresponde a la categoría global "Men", el 15 % corresponde a "Women", el 10 % corresponde a "Kids" y el 15 % corresponde a "Baby".

En los ejemplos de las Figuras 34, 35, 36, 37 y 38, la influencia de las descripciones textuales ha permitido que sean retornados productos óptimos para enfrentar el frío, tales como chaquetas y capas interiores. Además, potencialmente debido al uso de materiales similares, también se muestran productos de calzado, almohadas, entre otros.

Si bien es con el *baseline* que se obtiene una mayor cantidad de guantes (Figura 34), su porcentaje de acierto de categoría global (35 %) es superado por el modelo planteado en los 4 escenarios estudiados, obteniéndose una mayor precisión (60 %) al usar matrices de similitud coseno con *softmax* junto a la función *s_prob* (Figura 38).

Respecto a lo anterior, es importante aclarar que debido a la reducida cantidad de categorías globales en el catálogo de UNIQLO y al hecho de que la categoría "Men" es la más frecuente en este (Tabla 1), el haber obtenido mayores coincidencias de esta categoría potencialmente ha sido causa del azar y no de las propiedades del modelo entrenado.

c) Nombre del producto: WOMEN ULTRA STRETCH HIGH-RISE LEGGINGS PANTS

Categoría global: Women

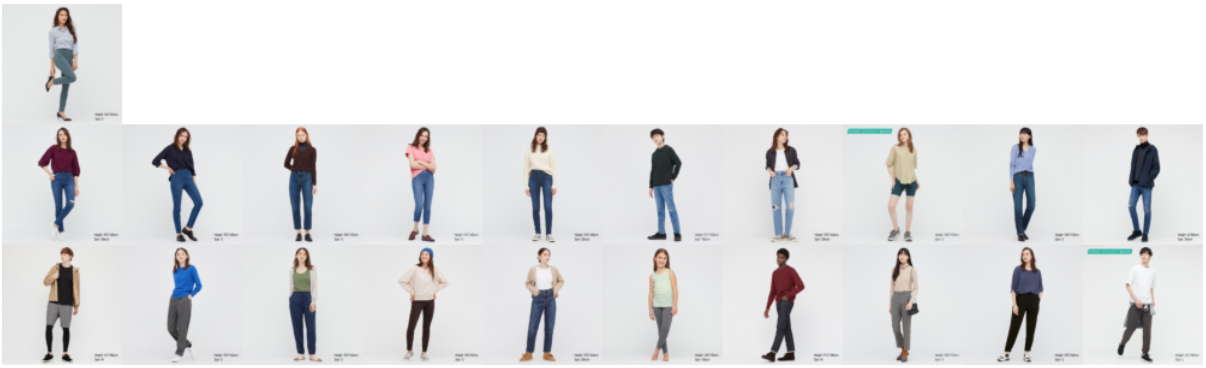


Figura 39: Recuperación de los 20 productos más similares a la *query* según el *baseline* ResNet-50 (sin entrenamiento). Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 70 % corresponde a la categoría global "Women", el 10 % corresponde a "Kids" y el 20 % corresponde a "Men".

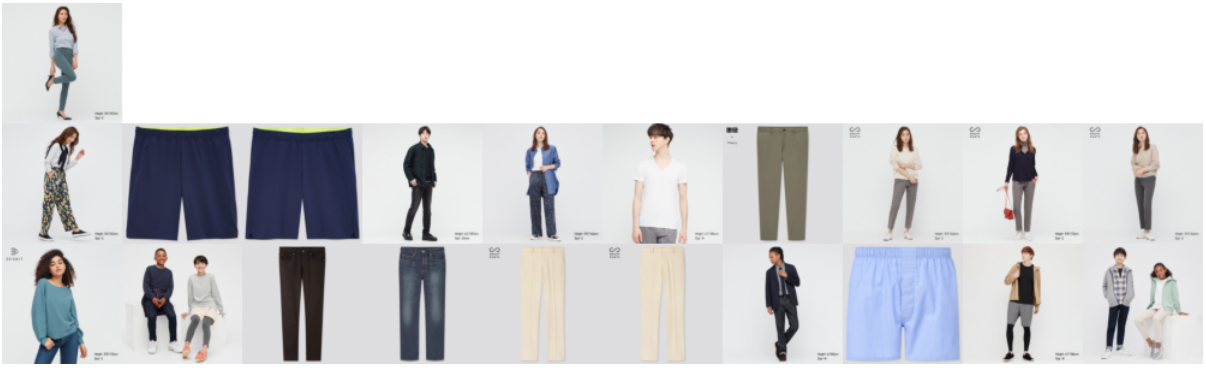


Figura 40: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 45 % corresponde a la categoría global "Women", el 45 % corresponde a "Men" y el 10 % corresponde a "Kids".

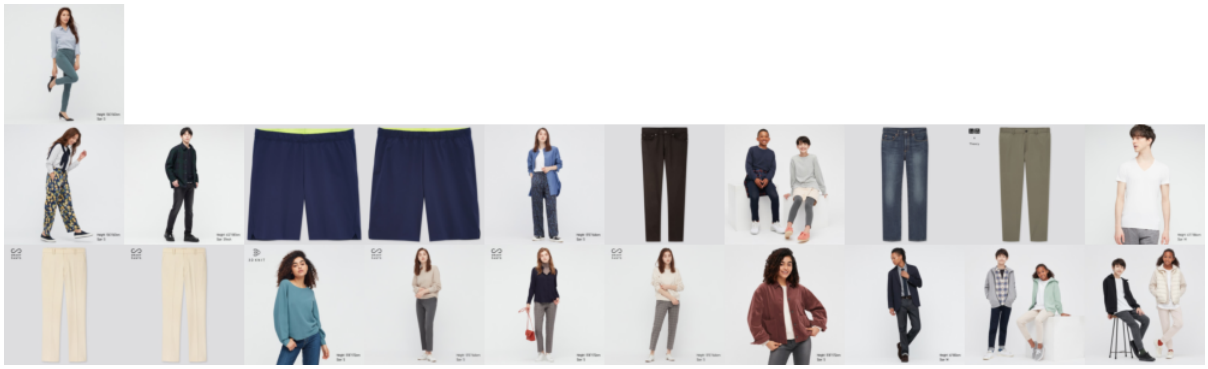


Figura 41: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 50 % corresponde a la categoría global "Women", el 35 % corresponde a "Men" y el 15 % corresponde a "Kids".

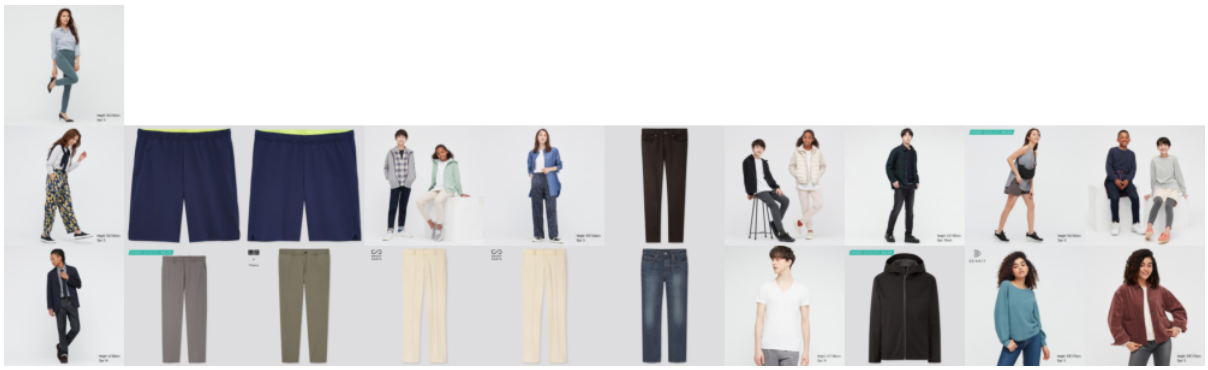


Figura 42: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 40 % corresponde a la categoría global "Women", el 45 % corresponde a "Men" y el 15 % corresponde a "Kids".

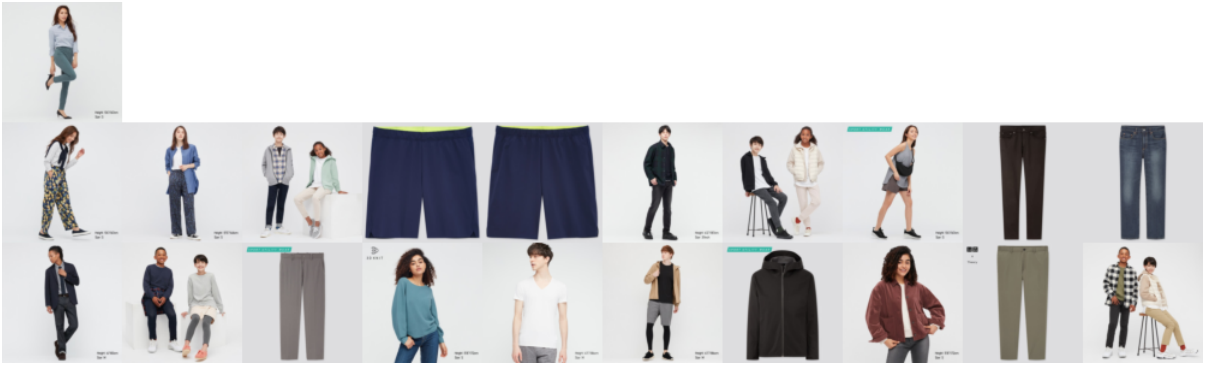


Figura 43: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 30 % corresponde a la categoría global "Women", el 20 % corresponde a "Kids" y el 50 % corresponde a "Men".

En los ejemplos de las Figuras 39, 40, 41, 42 y 43, el *baseline* entrega productos de distinta índole (pantalones, camisas, entre otros) debido a la similitud visual entre las imágenes provistas (modelos posando con dichos artículos).

Al usar el modelo propuesto, algunos de los artículos entregados por el *baseline* son reemplazadas por resultados más relevantes como pantalones largos y *shorts* (puesto que un usuario que busca *leggings* estará más interesado en distintos tipos de pantalones que en las camisas sugeridas por el *baseline*), mas también el modelo entrega nuevas sugerencias no pertinentes (chaquetas y chalecos).

Ninguno de los 4 escenarios del modelo propuesto logra superar el 70 % de acierto de categoría global que se obtuvo con el *baseline* (Figura 39), consiguiéndose el mayor porcentaje de acierto (50 %) al usar matrices de similitud coseno sin *softmax* junto a la función *s_prob* (Figura 41).

4.5. Búsqueda en el Catálogo de IKEA

Configuración de ajuste	**mAP@20(GC)**
Baseline (ResNet-50)	71.36 %
Baseline (mejor resultado con VETE)	75.13 %
Similitud coseno, $s\_linear(i)$	64.06 %
Similitud coseno, $s\_prob(i)$	64.72 %
Similitud coseno+softmax, $s\_linear(i)$	63.72 %
Similitud coseno+softmax, $s\_prob(i)$	63.10 %

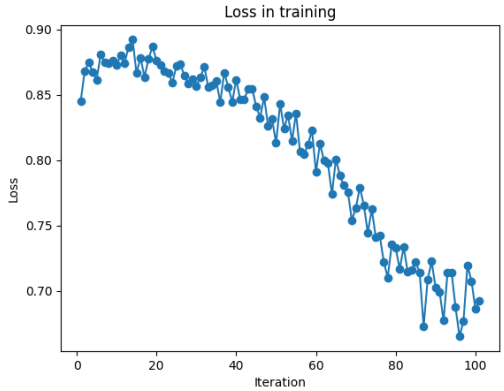
Tabla 5: Evaluación de la recuperación de productos del catálogo de IKEA, usando los dos *baselines* planteados junto con cuatro escenarios para el modelo propuesto

En la Tabla 5 se puede apreciar que, nuevamente, en ninguno de los 4 escenarios propuestos el modelo fue capaz de superar la coincidencia de categorías globales obtenidas por los modelos *baselines*. Para el catálogo de IKEA, se obtuvo una mayor precisión al utilizar matrices de similitud coseno sin *softmax*, además de la función $s\_prob(i)$ dentro del cálculo de pérdida.

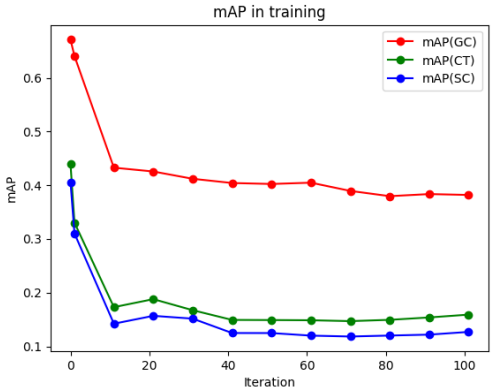
4.5.1. Evolución de mAP y Pérdida

Las Figuras 44, 45, 46 y 47 detallan la evolución de los valores de mAP@20 y del valor de pérdida a lo largo de las 100 iteraciones de entrenamiento.

Similitud coseno, *s_ linear(i)*



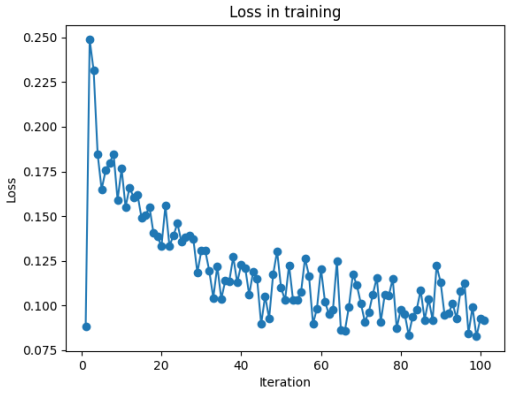
(a) Evolución del valor de pérdida a lo largo del entrenamiento.



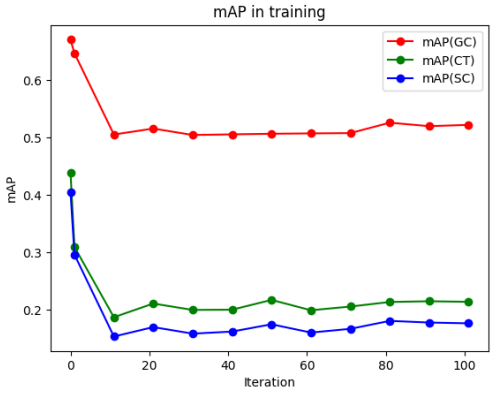
(b) Evolución de mAP@20, comenzando con el puntaje baseline de ResNet-50.

Figura 44: Evolución de la búsqueda en el catálogo de IKEA a lo largo de 100 iteraciones de entrenamiento usando similitud coseno y $s\_linear(i)$

Similitud coseno, *s_prob(i)*



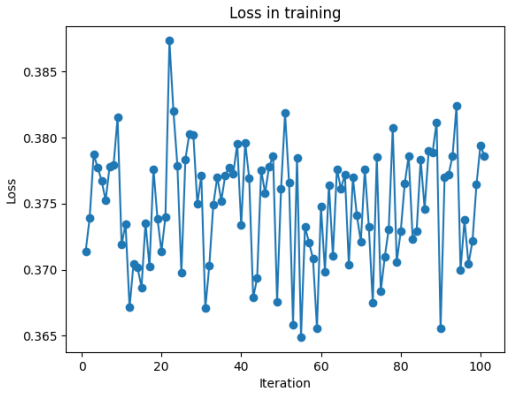
(a) Evolución del valor de pérdida a lo largo del entrenamiento.



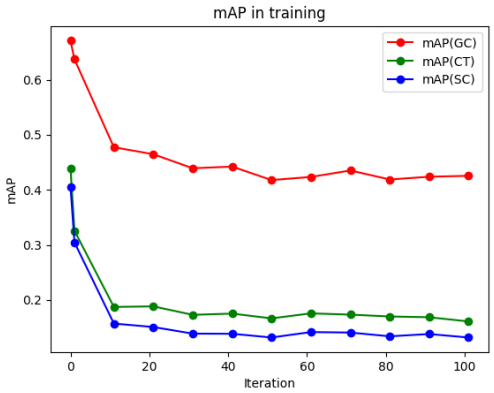
(b) Evolución de mAP@20, comenzando con el puntaje baseline de ResNet-50.

Figura 45: Evolución de la búsqueda en el catálogo de IKEA a lo largo de 100 iteraciones de entrenamiento usando similitud coseno y $s\_prob(i)$

Similitud coseno+softmax, *s_linear(i)*



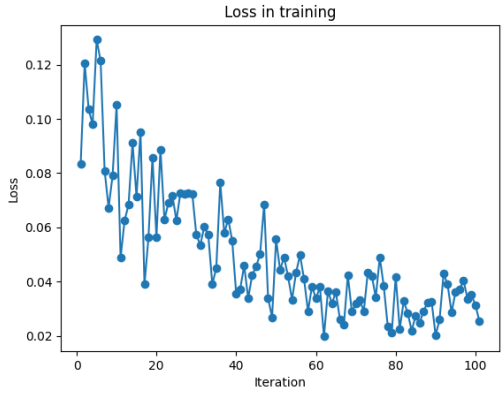
(a) Evolución del valor de pérdida a lo largo del entrenamiento.



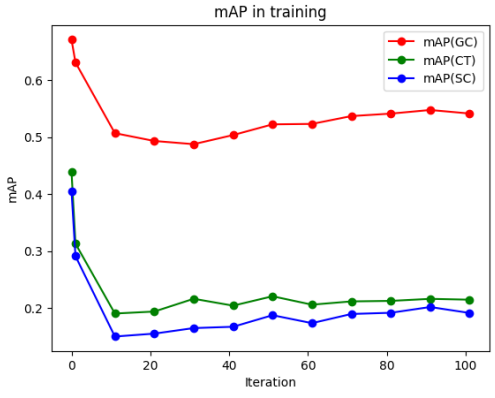
(b) Evolución de mAP@20, comenzando con el puntaje baseline de ResNet-50.

Figura 46: Evolución de la búsqueda en el catálogo de IKEA a lo largo de 100 iteraciones de entrenamiento usando similitud coseno con softmax y $s\_linear(i)$

Similitud coseno+softmax, *s_prob(i)*



(a) Evolución del valor de pérdida a lo largo del entrenamiento.



(b) Evolución de mAP@20, comenzando con el puntaje baseline de ResNet-50.

Figura 47: Evolución de la búsqueda en el catálogo de IKEA a lo largo de 100 iteraciones de entrenamiento usando similitud coseno con softmax y $s\_prob(i)$

Nuevamente se observa un rápido descenso de los valores de mAP, además de un comportamiento más errático del valor de pérdida que con el catálogo de UNIQLO, en particular al usar matrices de similitud coseno con *softmax* (Figuras 46.a y 47.a).

4.5.2. Ejemplos de Recuperación

Se presentan ejemplos de productos recuperados de acuerdo a los espacios vectoriales de *embeddings* visuales del *baseline* de ResNet-50 y de los 4 escenarios propuestos para este modelo, para 3 *queries* del set de evaluación.

a) Nombre del producto: BORRBY Lantern for block candle
Categoría global: Home Décor

Figura 48: Recuperación de los 20 productos más similares a la *query* según el *baseline* ResNet-50 (sin entrenamiento). Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 20 % corresponde a la categoría global "Home Décor", el 50 % corresponde a "Furniture", el 5 % corresponde a "Baby & kids", el 20 % corresponde a "Storage & organization" y el 5 % corresponde a "Kitchen & appliances".

Figura 49: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 95 % corresponde a la categoría global "Home Décor" y el 5 % corresponde a "Lighting".

Figura 50: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 90 % corresponde a la categoría global "Home Décor" y el 10 % corresponde a "Lighting".

Figura 51: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 100 % corresponde a la categoría global "Home Décor".

Figura 52: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 95 % corresponde a la categoría global "Home Décor" y el 5 % corresponde a "Lighting".

En los ejemplos de las Figuras 48, 49, 50, 51 y 52, se ha logrado eliminar de los productos recuperados a aquellos visualmente similares pero prácticamente irrelevantes, como las sillas. En su lugar, los modelos ahora entregan productos relacionados con la iluminación, tales como ampolletas (bombillas) y velas.

El 20 % de acierto de categoría global del *baseline* (Figura 48) es considerablemente superado por el modelo planteado en los 4 escenarios estudiados, obteniéndose una mayor precisión (100 %) al usar matrices de similitud coseno con *softmax* junto a la función *s_linear* (Figura 51). Para los otros 3 escenarios, todos los productos rescatados que no coinciden con la categoría "Home Décor" pertenecen a la categoría "Lighting" (Figuras 49, 50 y 52), que posee una aparente relación con el producto *query*.

b) Nombre del producto: KARISMATISK Shopping bag
Categoría global: Storage & organization

Figura 53: Recuperación de los 20 productos más similares a la *query* según el *baseline* ResNet-50 (sin entrenamiento). Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 35 % corresponde a la categoría global "Storage & organization", el 25 % corresponde a "Bathroom", el 10 % corresponde a "Cookware & tableware", el 10 % corresponde a "Baby & kids", el 10 % corresponde a "Lighting", el 5 % corresponde a "Laundry & cleaning" y el 5 % corresponde a "Hoonde a "Laundry & cleaning" y el 5 % corresponde a "Home Décor".

Figura 54: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 40 % corresponde a la categoría global "Storage & organization", el 35 % corresponde a "Home Décor", el 15 % corresponde a "Beds & mattresses", el 5 % corresponde a "Home improvement" y el 5 % corresponde a "Baby & kids".

Figura 55: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 35 % corresponde a la categoría global "Storage & organization", el 40 % corresponde a "Home Décor", el 10 % corresponde a "Beds & mattresses", el 5 % corresponde a "Home improvement", el 5 % corresponde a "Baby & kids" y el 5 % corresponde a "Bathroom".

Figura 56: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 50 % corresponde a la categoría global "Storage & organization", el 25 % corresponde a "Home Décor", el 15 % corresponde a "Beds & mattresses", el 5 % corresponde a "Home improvement" y el 5 % corresponde a "Baby & kids".

Figura 57: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 55 % corresponde a la categoría global "Storage & organization", el 20 % corresponde a "Home Décor", el 15 % corresponde a "Beds & mattresses", el 5 % corresponde a "Home improvement" y el 5 % corresponde a "Baby & kids".

En los ejemplos de las Figuras 53, 54, 55, 56 y 57, los modelos logran entregar nuevos productos más pertinentes a la *query*, como otros tipos de contenedores de tela. El 35 % de acierto de categoría global del *baseline* (Figura 53) es superado o igualado por el modelo planteado en los 4 escenarios estudiados, obteniéndose una mayor precisión (55 %) al usar matrices de similitud coseno con *softmax* junto a la función *s_prob* (Figura 57).

Sin embargo, también son rescatados productos irrelevantes, como lo son las toallas y la ropa de cama.

c) **Nombre del producto:** VARDAGEN Serving bowl
Categoría global: Cookware & tableware

Figura 58: Recuperación de los 20 productos más similares a la *query* según el *baseline* ResNet-50 (sin entrenamiento). Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 75 % corresponde a la categoría global "Cookware & tableware", el 10 % corresponde a "Home Décor", el 5 % corresponde a "Storage & organization" y el 10 % corresponde a "Lighting".

Figura 59: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 90 % corresponde a la categoría global "Cookware & tableware" y el 10 % corresponde a "Baby & kids".

Figura 60: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 90 % corresponde a la categoría global "Cookware & tableware" y el 10 % corresponde a "Baby & kids".

Figura 61: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 85 % corresponde a la categoría global "Cookware & tableware" y el 15 % corresponde a "Baby & kids".

Figura 62: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 90 % corresponde a la categoría global "Cookware & tableware" y el 10 % corresponde a "Baby & kids".

En los ejemplos de las Figuras 58, 59, 60, 61 y 62, si bien los modelos son capaces de entregar boles o tazones no retornados por el *baseline*, este es un producto que se beneficia de sus rasgos visuales, pues al poseer el vidrio un aspecto característico permite de por sí que productos relevantes sean entregados sin necesidad de recurrir a información textual. Los tazones entregados por los modelos son menos relevantes que los retornados por el *baseline* debido a sus materiales.

El 75 % de acierto de categoría global del *baseline* (Figura 58) es superado por el modelo planteado en los 4 escenarios estudiados, obteniéndose una precisión de 85 % al usar matrices de similitud coseno con *softmax* junto a la función *s_linear* (Figura 61), y una precisión de 90 % en los otros 3 escenarios (Figuras 59, 60 y 62).

4.6. Búsqueda en el Catálogo de Pepeganga

Por motivos del alto uso de memoria por las matrices de similitud, no ha sido posible trabajar con la misma extensión del catálogo de Pepeganga estudiada para VETE [5], por lo que se han computado modelos para la mitad de los productos disponibles (como fue mencionado en la Subsección 3.2.1).

Configuración de ajuste	**mAP@20(GC)**
Baseline (ResNet-50)	80.32 %
Similitud coseno, $s\_linear(i)$	76.52 %
Similitud coseno, $s\_prob(i)$	76.38 %
Similitud coseno+softmax, $s\_linear(i)$	76.88 %
Similitud coseno+softmax, $s\_prob(i)$	76.46 %

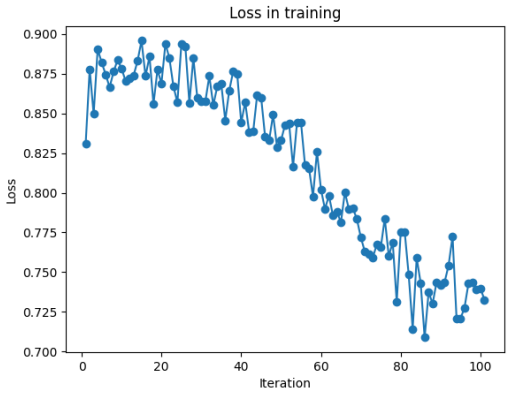
Tabla 6: Evaluación de la recuperación de productos del catálogo de Pepeganga, usando los dos *baselines* planteados junto con cuatro escenarios para el modelo propuesto

En la Tabla 6 se puede apreciar que, nuevamente, en ninguno de los 4 escenarios propuestos el modelo fue capaz de superar la coincidencia de categorías globales obtenidas por el modelo *baseline*. Para el catálogo de Pepeganga, se obtuvo una mayor precisión al utilizar matrices de similitud coseno con *softmax*, además de la función $s\_linear(i)$ dentro del cálculo de pérdida.

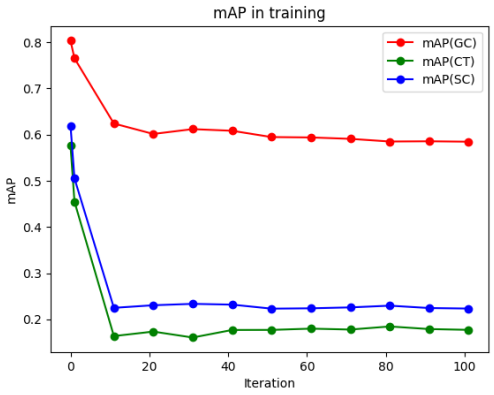
4.6.1. Evolución de mAP y Pérdida

Las Figuras 63, 64, 65 y 66 detallan la evolución de los valores de mAP@20 y del valor de pérdida a lo largo de las 100 iteraciones de entrenamiento.

Similitud coseno, *s_linear(i)*



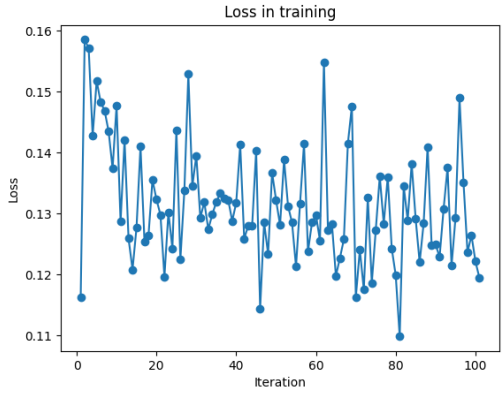
(a) Evolución del valor de pérdida a lo largo del entrenamiento.



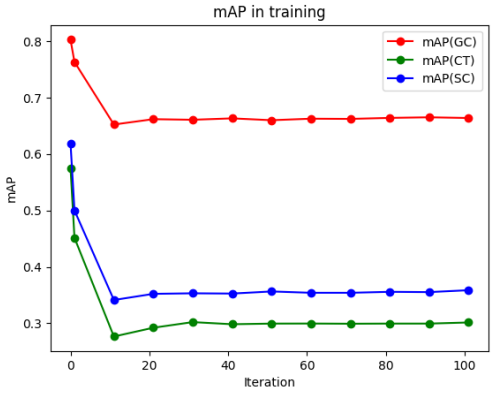
(b) Evolución de mAP@20, comenzando con el puntaje baseline de ResNet-50.

Figura 63: Evolución de la búsqueda en el catálogo de Pepeganga a lo largo de 100 iteraciones de entrenamiento usando similitud coseno y $s\_linear(i)$

Similitud coseno, *s_prob(i)*



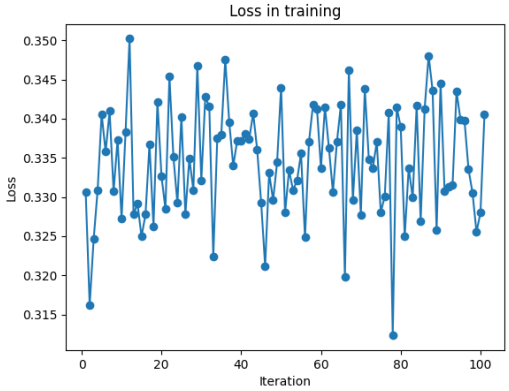
(a) Evolución del valor de pérdida a lo largo del entrenamiento.



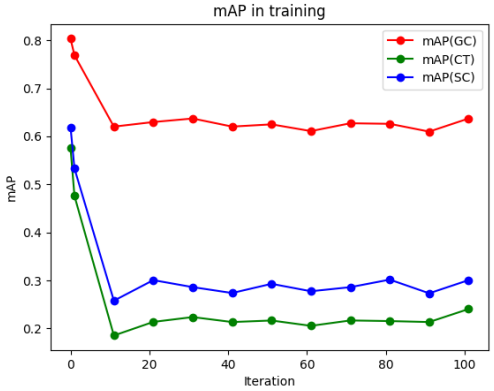
(b) Evolución de mAP@20, comenzando con el puntaje baseline de ResNet-50.

Figura 64: Evolución de la búsqueda en el catálogo de Pepeganga a lo largo de 100 iteraciones de entrenamiento usando similitud coseno y $s\_prob(i)$

Similitud coseno+softmax, *s_linear(i)*



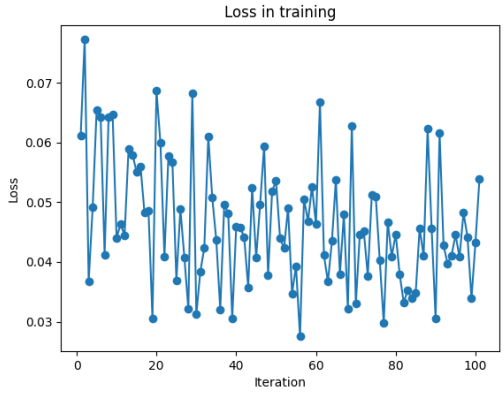
(a) Evolución del valor de pérdida a lo largo del entrenamiento.



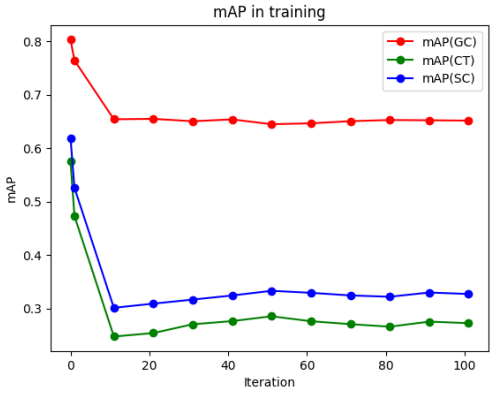
(b) Evolución de mAP@20, comenzando con el puntaje baseline de ResNet-50.

Figura 65: Evolución de la búsqueda en el catálogo de Pepeganga a lo largo de 100 iteraciones de entrenamiento usando similitud coseno con softmax y $s\_linear(i)$

Similitud coseno+softmax, *s_prob(i)*



(a) Evolución del valor de pérdida a lo largo del entrenamiento.



(b) Evolución de mAP@20, comenzando con el puntaje baseline de ResNet-50.

Figura 66: Evolución de la búsqueda en el catálogo de Pepeganga a lo largo de 100 iteraciones de entrenamiento usando similitud coseno con softmax y $s\_prob(i)$

En esta ocasión, la variación del valor de pérdida es mucho más errática que en los catálogos previos, siendo más predecible al usar matrices de similitud coseno sin *softmax* y funciones *s_linear* (Figura 63.a). Una potencial causa puede ser el aumento de artículos y pares de nodos a trabajar, por lo que un posible ajuste para un potencial trabajo futuro sería el de aumentar el tamaño del *batch* de los pares de nodos escogidos aleatoriamente al momento de calcular la pérdida de cada iteración.

4.6.2. Ejemplos de Recuperación

Se presentan ejemplos de productos recuperados de acuerdo a los espacios vectoriales de *embeddings* visuales del *baseline* de ResNet-50 y de los 4 escenarios propuestos para este modelo, para 3 *queries* del set de evaluación.

a) Nombre del producto: Auriculares Graphite Metal - Rosado
Categoría global: Technology

Figura 67: Recuperación de los 20 productos más similares a la *query* según el *baseline* ResNet-50 (sin entrenamiento). Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 30 % corresponde a la categoría global "Technology", el 15 % corresponde a "Beauty", el 15 % corresponde a "Home", el 5 % corresponde a "Sports", el 15 % corresponde a "Babies", el 5 % corresponde a "Toy Store", el 10 % corresponde a "School" y el 5 % corresponde a "Pets".

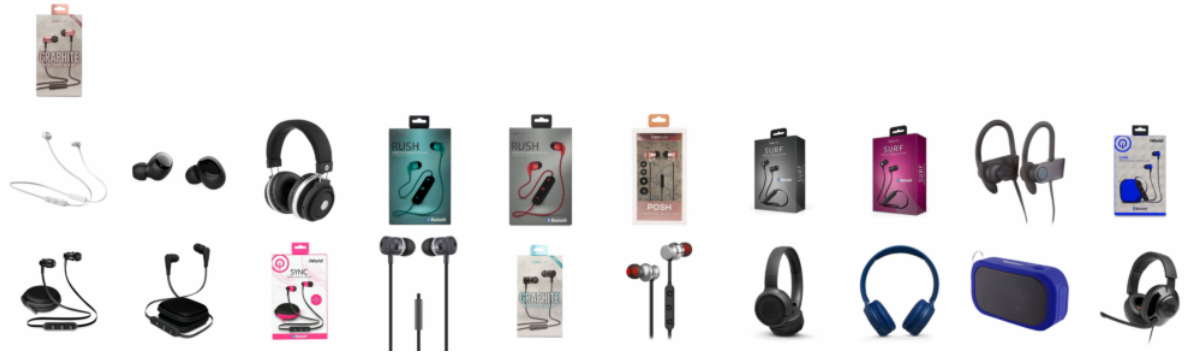


Figura 68: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 100 % corresponde a la categoría global "Technology".

Figura 69: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 90 % corresponde a la categoría global "Technology" y el 10 % corresponde a "Sports".

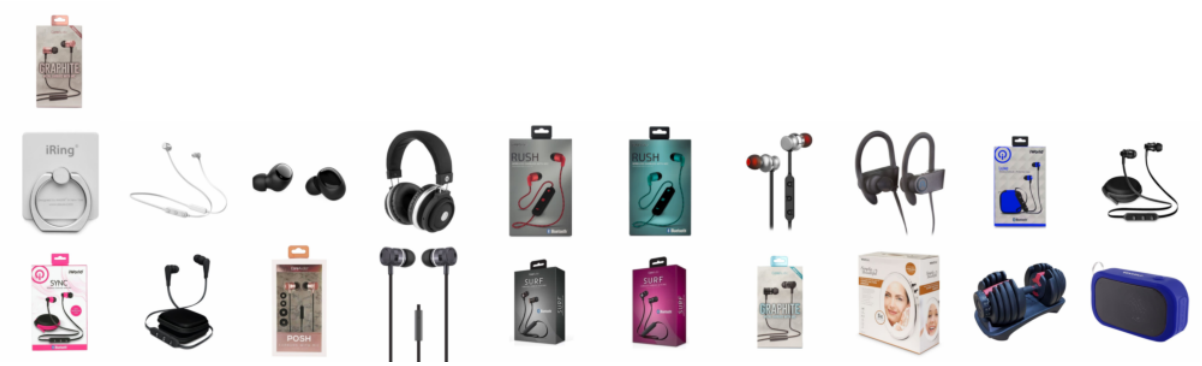


Figura 70: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 90 % corresponde a la categoría global "Technology", el 5 % corresponde a "Home" y el 5 % corresponde a "Sports".

Figura 71: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 100 % corresponde a la categoría global "Technology".

En los ejemplos de las Figuras 67, 68, 69, 70 y 71, se retornan más audífonos de formatos varios, de manera independiente de si son representados en su empaque o no. Estos productos reemplazan a otros productos irrelevantes a la consulta que fueron entregados por el *baseline* debido al empaque similar.

El 30 % de acierto de categoría global del *baseline* (Figura 67) es considerablemente superado por el modelo planteado en los 4 escenarios estudiados, obteniéndose una mayor precisión (100 %) al usar matrices de similitud coseno sin *softmax* junto a la función *s_linear* (Figura 68) y usando matrices de similitud coseno con *softmax* junto a la función *s_prob* (Figura 71).

b) **Nombre del producto:** Set Club Chelsea Columpio - Barbie
Categoría global: Toy Store

Figura 72: Recuperación de los 20 productos más similares a la *query* según el *baseline* ResNet-50 (sin entrenamiento). Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 100 % corresponde a la categoría global "Toy Store".

Figura 73: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 100 % corresponde a la categoría global "Toy Store".

Figura 74: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 100 % corresponde a la categoría global "Toy Store".

Figura 75: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 95 % corresponde a la categoría global "Toy Store" y el 5 % corresponde a "School".

Figura 76: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 100 % corresponde a la categoría global "Toy Store".

En los ejemplos de las Figuras 72, 73, 74, 75 y 76, los nuevos modelos reemplazan varios de los artículos tipo *play-set* por otros juguetes y productos enfocados al mismo público objetivo que, si bien son pertinentes, no son más similares a la consulta que los artículo retornados por el *baseline*.

El escenario en donde se utilizan matrices de similitud coseno con *softmax* junto con la función *s_linear* (Figura 75) es el único en donde no se alcanza el 100 % de acierto de categoría global obtenido por el *baseline* (Figura 72)

c) **Nombre del producto:** Tendedero Vertical Acero Inoxidable 17M Tramontina
Categoría global: Home

Figura 77: Recuperación de los 20 productos más similares a la *query* según el *baseline* ResNet-50 (sin entrenamiento). Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 75 % corresponde a la categoría global "Home", el 5 % corresponde a "Furniture", el 5 % corresponde a "Sports", el 10 % corresponde a "Technology" y el 5 % corresponde a "Babies".

Figura 78: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 80 % corresponde a la categoría global "Home", el 10 % corresponde a "Sports", el 5 % corresponde a "Home Appliances" y el 5 % corresponde a "Pets".

Figura 79: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 85 % corresponde a la categoría global "Home", el 10 % corresponde a "Sports" y el 5 % corresponde a "Home Appliances".

Figura 80: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_linear*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 85 % corresponde a la categoría global "Home", el 10 % corresponde a "Sports" y el 5 % corresponde a "Home Appliances".

Figura 81: Recuperación de los 20 productos más similares a la *query* tras 1 iteración de entrenamiento con *matriz de similitud coseno con softmax y función s_prob*. Arriba a la izquierda se presenta la imagen de consulta, y el orden de los resultados recuperados es de izquierda a derecha, y de arriba hacia abajo. De estos, el 85 % corresponde a la categoría global "Home", el 10 % corresponde a "Sports" y el 5 % corresponde a "Home Appliances".

En los ejemplos de las Figuras 77, 78, 79, 80 y 81, tanto el *baseline* como los modelos propuestos entregan productos irrelevantes. Al consultar con *baseline*, se retornan productos visualmente similares, mas la mayoría de estos no son tendederos. Los modelos propuestos retornan algunos productos de acero inoxidable, pocos de ellos siendo tendederos.

El 75 % de acierto de categoría global del *baseline* (Figura 77) es superado por el modelo planteado en los 4 escenarios estudiados, obteniéndose una precisión de 80 % al usar matrices de similitud coseno sin *softmax* junto a la función *s_linear* (Figura 78), y una precisión de 85 % en los otros 3 escenarios (Figuras 79, 80 y 81).

5. Conclusiones

A lo largo del presente informe, se presentó la propuesta de un modelo de recuperación de productos de catálogos de eCommerce basado en redes neuronales de grafos como alternativa a la metodología VETE de generación de *embeddings* visuales basada en texto, fundamentada en promedios de los *embeddings* retornados por modelos como ResNet-50.

La metodología planteada se sustenta en la construcción de un grafo completo con un nodo por cada producto. Cada nodo contiene un *embedding* en entrenamiento, calculado con ResNet-50, al cual se le propaga información de sus vecinos de acuerdo a pesos definidos por la similitud entre el texto de la descripción de los artículos. De este modo, al realizar una consulta a modo de imagen es posible recuperar productos no solo visualmente similares, sino también que posean características potencialmente deseables y/o relevantes no apreciables en una imagen.

Para que esto fuese llevado a cabo, se planteó una función de pérdida que optimizara, a lo largo del entrenamiento, el "acercamiento" de *embeddings* visuales correspondientes a pares de productos cuyos *embeddings* de texto fuesen similares, logrando así la disminución del valor de pérdida.

Fueron diseñadas funciones de probabilidad –ajustables exponencialmente– para la selección aleatoria de *batches* de pares de productos en función del rango de los puntajes de similitud de *embeddings* de texto y, posteriormente, ser estos pares considerados para la evaluación de pérdida. La función de probabilidad propuesta permite priorizar la selección de los pares más similares textualmente y los más distintos, para así proveer a la función de pérdida de valores más característicos para el aprendizaje.

Teniendo esto último en cuenta, el diseño y entrenamiento del modelo —como una extensión de VETE— resulta cumplirse con éxito, no obstante, la precisión de la búsqueda visual (medida mediante mAP@20) no logró superar a lo obtenido usando las metodologías ya conocidas.

Los resultados usando la metodología basada en GNNs reportaron valores de mAP@20 entre 59.37 % y 60.6 % para UNIQLO (Tabla 4), 63.1 % y 64.72 % para IKEA (Tabla 5), y 76.38 % y 76.88 % para Pepeganga (Tabla 6) –usando las 4 distintas especificaciones planteadas–, mientras que en el escenario base de la búsqueda en el espacio visual definido por ResNet-50 sin ajustar se obtuvieron los valores 70.66 %, 71.36 % y 80.32 %, correspondientemente. Resultados aún más altos se obtienen tras el ajuste de *embeddings* realizado por VETE.

A pesar de esto, para ciertas consultas ejemplificadas en las Secciones 4.5.2 y 4.6.2 se obtienen resultados más relevantes al compararlos directamente con aquellos retornados por la búsqueda en el espacio visual de ResNet-50 sin ajuste. Para estos casos, se ha comprobado que es posible aumentar el porcentaje de coincidencias entre las categorías globales de la *query* y de los productos sugeridos. Esto es, el modelo sí ha sido capaz de retornar productos con características más pertinentes, mas de vez en cuando junto a otros productos ni visual ni textualmente relevantes.

El uso de las alternativas para las distintas componentes del modelo provocan alteraciones en aspectos del entrenamiento –como la evolución de la pérdida. Sin embargo, el rendimiento neto no tiene variaciones considerables frente a estos cambios, tanto en el análisis cuantitativo (en base a los puntajes de mAP) como cualitativo.

Es posible que haya existido una "sobrecorrección" al momento de propagar información en el campo visual, lo que permitiría explicar los mayores puntajes de mAP obtenidos al inicio del entrenamiento. Un posible trabajo futuro sería el de confirmar o negar esto, y en caso de confirmarlo, trabajar en estrategias para mitigarlo.

Si bien en su estado actual el trabajo realizado no significa una mejora frente a estrategias como VETE [5] en términos de mAP@20, la capacidad de propagar características de un espacio vectorial (en este caso, texto) a lo largo de vectores de otro espacio (en este caso, visual) por medio de GNNs ha sido demostrada, en cierta medida, al realizar un análisis cualitativo y cuantitativo de algunos resultados ejemplificados. Por lo anterior, se cree que esta estrategia podría ser explorada aún más para potencialmente obtener mejores resultados cuantitativos y una mejor evolución de estos a lo largo del entrenamiento.

Un posible trabajo futuro, contemplado en la planificación original de este estudio, es el uso de vectores de características generados por CLIP, los cuales ya dieron resultados favorables en VETE.

Bibliografía

[1] Devlin, Jacob, Ming Wei Chang, Kenton Lee y Kristina Toutanova: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 2018. https://arxiv.org/abs/1810.04805.

[2] Fajardo, Daniel: *Los cambios en el comercio electrónico que se aceleraron con la pandemia*. La Tercera, Noviembre 2021. https://www.latercera.com/pulso/noticia/los-cambios-en-el-comercio-electronico-que-se-aceleraron-con-la-pandemia/VQR6XKWSBRAQXIGQVUQDCFMRIE/.

[3] He, Kaiming, Xiangyu Zhang, Shaoqing Ren y Jian Sun: *Deep Residual Learning for Image Recognition*, 2015. https://arxiv.org/abs/1512.03385.

[4] Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer y Veselin Stoyanov: *RoBERTa: A Robustly Optimized BERT Pretraining Approach*, 2019. https://arxiv.org/abs/1907.11692.

[5] Martínez, Guillermo, Jose M. Saavedra y Nils Murrugara-Llerena: *VETE: improving visual embeddings through text descriptions for eCommerce search engines*, 2023. https://link.springer.com/article/10.1007/s11042-023-14595-8.

[6] McConnell, R K: *Method of and apparatus for pattern recognition*. Enero 1986. https://www.osti.gov/biblio/6007283.

[7] McInnes, Leland, John Healy y James Melville: *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*, 2018. https://arxiv.org/abs/1802.03426.

[8] Mikolov, Tomas, Kai Chen, Greg Corrado y Jeffrey Dean: *Efficient Estimation of Word Representations in Vector Space*, 2013. https://arxiv.org/abs/1301.3781.

[9] Misraa, Aashish Kumar, Ajinkya Kale, Pranav Aggarwal y Ali Aminian: *Multi-Modal Retrieval using Graph Neural Networks*, 2020. https://arxiv.org/abs/2010.01666.

[10] Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger y Ilya Sutskever: *Learning Transferable Visual Models From Natural Language Supervision*, 2021. https://arxiv.org/abs/2103.00020.