Recuperación de Información y Recomendaciones en la Web

Universidad de la República, Uruguay

Curso 2020

Buscador de arte

Recomendación de obras de arte en base a una paleta de colores

Autores:
Cindy Acuña
María Noel Bassagoda
Leandro Dominguez
Agustín Mautone
Guillermo Ripa

Docente:
Libertad Tansini

27 de noviembre de 2020

Índice

1. Introducción

Desde la web hasta el diseño de interiores, los esquemas de color juegan un papel fundamental en la creación de las mejores experiencias de usuario, estableciendo identidades de marca y comunicando estados de ánimo o emociones.

A la hora de elegir una pintura es de vital importancia lograr la perfecta combinación de colores con el entorno que la rodea. No sólo esto, quizás también se busca encontrar la pareja perfecta para un cuadro que ya tenemos. Para esto, es necesario lograr extraer la paleta de colores de una imagen, ya sea de un cuadro o un entorno y luego recomendar obras de arte que compartan estos mismos.

2. Problema y motivación

La motivación de este proyecto surge al analizar distintos sitios de venta de obras de arte donde vimos que en todos se proveían distintos filtros como ser el precio, estilo, artista, etc. Lo que no encontramos fue ningún sitio donde se pudiera filtrar según una determinada paleta de colores. Así surge la idea de desarrollar una herramienta de recomendación de obras de arte en base a sus paletas de colores.

Este problema esta compuesto por varias componentes, a continuación se listan algunos de los problemas que fueron resueltos:

- Conseguir un conjunto de pinturas extenso del cual se pueda extraer las paletas de colores e información relevante de cada obra.
- Indexar de manera eficiente las paletas de colores del conjunto de pintura. Para ello se utilizó también un modelo de IA pre-entrenado con el objetivo de pasar las paletas de colores a una representación vectorial. Facilitando así la búsqueda de las K pinturas más cercanas según una medida de distancia entre la representación de las paletas de colores que se generaron a partir del modelo.
- Implementar una interfaz que permita subir una imagen con la que se realiza la consulta al índice.
- Dentro de la interfaz agregar filtros con los cuales se permite afinar mejor la búsqueda del usuario.

3. Descripción de la solución

Lo primero que se realizó fue la recopilación de información de miles de obras de arte, las cuales se obtuvieron mediante técnicas de web scraping. A partir de esto se generó un CSV con la información recuperada. Luego se utilizó el módulo de Python colorgram [3] para extraer la paleta de colores de cada imagen para luego generar el embedding de cada una de estas paletas de colores. Una vez obtenido todo esto, se procedió a indexar dicha información en un índice de ElasticSearch sobre el cual se generan las consultas necesarias para obtener las obras relevantes.

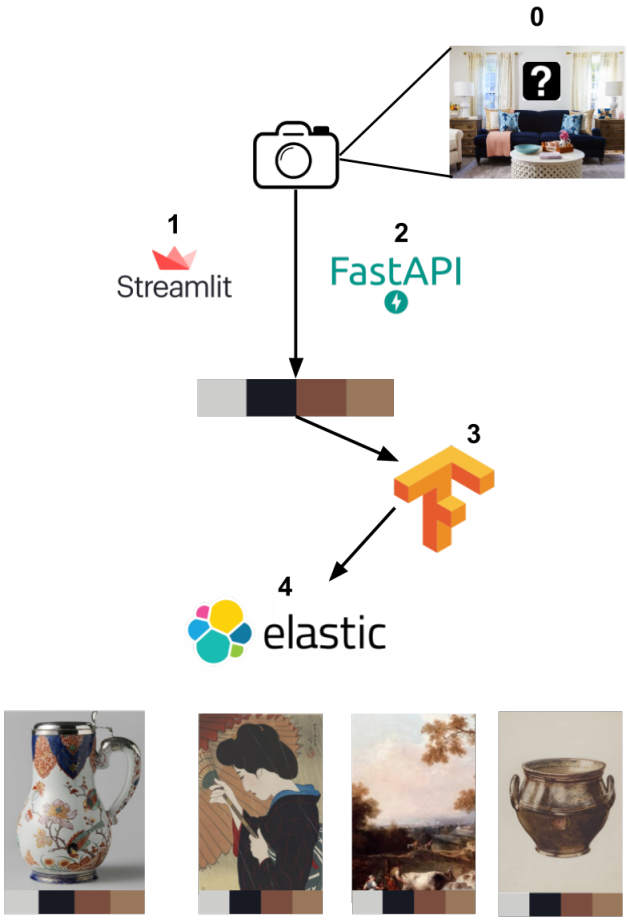
La herramienta consta de un sitio web donde se puede cargar una imagen o ingresar una URL de la misma. A partir de ésta, se procede a generar una paleta de colores que coincida con la imagen y se lo pasa por el modelo de IA para obtener el embedding asociado a dicha paleta. Finalmente se procede a consultar el índice de ElasticSearch por aquellas imágenes cuya paleta de colores se aproximan mejor a la paleta de la imagen, dada según su distancia euclídea.

En nuestro caso, nos limitamos a una paleta de 5 colores, pero esto podría ajustarse de tal forma que podamos afinar mejor la búsqueda, aumentando la cantidad de colores extraídos de las imágenes, siempre y cuando se tenga un modelo de IA lo suficientemente bueno como para generar el embedding asociado.

4. Diseño e Implementación

4.1. Arquitectura

La arquitectura es de tipo cliente-servidor, implementada en Python utilizando herramientas como Streamlit y FastApi. También se utilizo la plataforma TensorFlow para generar el embedding de la paleta de colores y por último Elasticsearch para almacenar los datos de forma indexada. A continuación se muestra un boceto de la arquitectura implementada:



4.2. Herramientas

4.2.1. Streamlit - Interfaz de Usuario

Streamlit [1] es un *framework* que permite el desarrollo de aplicaciones enfocadas en datos y la presentación de los mismos en un formato interactivo.

El sistema cuenta con un frontend desarrollado en Streamlit que permite cargar una imagen y consecuentemente mostrar las obras de arte similares. Se incluye también la información de su precio, título, artista y link para poder efectuar la compra. También permite aplicar distintos filtros como ser: el rangos de precio y el título de la pintura.

4.2.2. ElasticSearch

Es un motor de búsqueda y análisis con la capacidad de lograr búsquedas eficientes y configurables. Tambien se caracteriza por su capacidad de escalar con poco esfuerzo a ambientes distribuidos.

La arquitectura se desarrolla alrededor de un índice de ElasticSearch [2]. Este contiene toda la información necesaria para poder realizar las consultas correspondientes para obtener las pinturas cuyas paletas de colores se aproximan de mejor forma.

4.2.3. Scrapy

Es un framework de extracción de datos fácilmente extensible y de código abierto. Se puede utilizar para una amplia gama de propósitos, desde data-mining hasta monitoreo y pruebas automatizadas.

En este contexto, Scrapy [6] se utilizó para extraer los datos correspondientes a miles de obras de arte del sitio Saatchiart [4]. De las mismas se indexaron, para cada obra los siguientes datos:

- Título.
- Artista.
- Precio.
- Hipervínculo para realizar la compra.
- Hipervínculo a la imagen de la obra.

4.2.4. Colorgram

Es un módulo de Python que te permite extraer una paleta de colores de una imágenes dada. En este contexto utilizamos Colorgram [3] en dos instancias del trabajo. En primer lugar al momento de la recopilación de datos, ya que para generar el índice de ElasticSearch necesitábamos la información de la paleta de colores de cada imagen para generar el embedding asociado. Por otro lado, también se utiliza de igual forma al momento de procesar una imagen ingresada por el usuario, la cual pasa por el mismo proceso de generación de paleta de colores y embedding asociado, para luego consultar el índice de ElasticSearch.

4.2.5. Google Art Palette - Modelo de IA

En este contexto utilizamos el modelo de IA de la herramienta ArtPallete [5] de Google (el cual ya se encuentra preentrenado) para proyectar una paleta de 5 colores en un espacio vectorial de 15 dimensiones. En este nuevo espacio, la distancia euclídea entre los embeddings es una distancia perceptiva entre las paletas de colores originales. Es decir, que las paletas rojas van a estar más cerca entre sí que las verdes por ejemplo. Esto permite que la búsqueda se haga de manera eficiente usando la distancia euclídea para comparar los vectores.

4.2.6. FastAPI

Es un *web framework* desarrollado sobre Python, enfocado en la fácil implementación y altas velocidad de repuesta.

En este contexto, FastAPI [7] se utilizó para construir un servidor que, mediante la exposición de una *API*, permite tres funcionalidades fundamentales.

- La extracción de la paleta de colores de una imagen.
- La conversión de una paleta de colores a la representación obtenida mediante el modelo de IA.
- La búsqueda en el índice de ElasticSearch de las obras con una paleta parecida en el universo de representación.

4.3. Integración

En esta sección se detallará como todas estas herramientas son integradas para el funcionamiento de la aplicación.

El frontend, desarrollado con Streamlit, inicia mostrando dos formas de subir imágenes a la aplicación: mediante un URL a la imagen ó subiendo directamente el archivo de la imagen. Si se opta por utilizar una URL, la aplicación descargará la imagen antes de proceder. Esta imagen luego será enviada al backend para extraer su paleta y conseguir las recomendaciones.

Adicionalmente se agrega la opción de aplicar ciertos filtros, entre los cuales se encuentran titulo, rango de precios y nombre del artista. Estos serán incluidos dentro de la consulta al índice.

Como se mencionó anteriormente, el backend de esta aplicación contiene dos componentes principales.

El primero es un servidor de aplicaciones desarrollado en FastAPI. El mismo expone una API REST bajo el endpoint */recommendations/*.

Este endpoint recibe el archivo binario en relación a la imagen así como los filtros que se pasarán a ElasticSearch. Dentro del proceso encargado de este endpoint se realizan 4 subprocesos:

1. Se consigue la paleta de colores de la imagen.
2. Se realiza un forward-pass por el modelo de representation learning para conseguir el embedding.
3. Se produce la query en base a los filtros recibidos y el embedding.
4. Se realiza la query a ElasticSearch para conseguir los resultados relevantes.

Para realizar la la query a ElasticSearch se forma una query de tipo **script_score**. Este permite efectuar una evaluación de los resultados obtenidos por una query para determinar los scores que se le desean asignar a cada uno en base a una función dada.

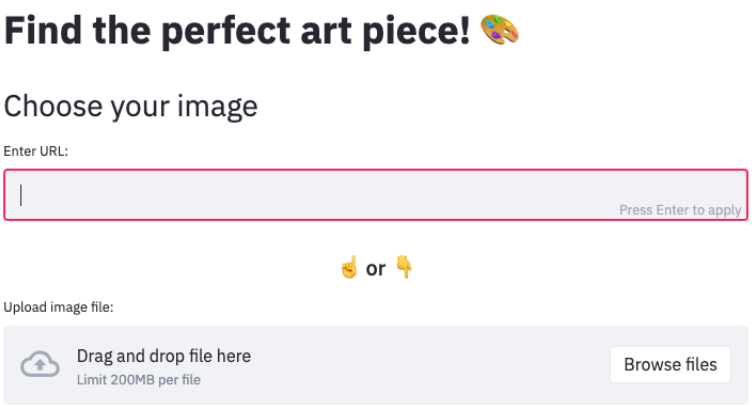
En "Ejemplo query ES" se posible visualizar los campos a cargo de los filtros por texto, así como los filtros de rango de precios. Por último se presenta en la sección *source* el código utilizado para la evaluación de los puntajes de relevancia. Este realiza el inverso de la distancia euclídea. Por lo que los resultados con menor distancia recibirán un puntaje final de relevancia mayor.

```
{
    "size":10
    "query":{
        "script_score":{
            "query":{
                "bool":{
                    "filter":[
                        0:{
                            "range":{
                                "price":{
                                    "gte":390
                                    "relation":"WITHIN"
                                    "lte":10000
                                }
                            }
                        }
                        1:{
                            "match":{
                                "title":{
                                    "query":"Time"
                                }
                            }
                        }
                        2:{
                            "match":{
                                "artist":{
                                    "query":"Alexandra"
                                }
                            }
                        }
                    ]
                }
            }
            "script":{
                "source":"1 / " + \
                "(1 + l2norm(" + \
                "    params.queryVector, doc['palette_embedding'])" + \
                ")"
                "params":{
                    "queryVector":[
                        0:"<class 'numpy.float32'>"
                        1:"<class 'numpy.float32'>"
                        2:"<class 'numpy.float32'>"
                        3:"<class 'numpy.float32'>"
                        4:"<class 'numpy.float32'>"
                        5:"<class 'numpy.float32'>"
                        6:"<class 'numpy.float32'>"
                        7:"<class 'numpy.float32'>"
                        8:"<class 'numpy.float32'>"
                        9:"<class 'numpy.float32'>"
                        10:"<class 'numpy.float32'>"
                        11:"<class 'numpy.float32'>"
                        12:"<class 'numpy.float32'>"
                        13:"<class 'numpy.float32'>"
                        14:"<class 'numpy.float32'>"
                    ]
```

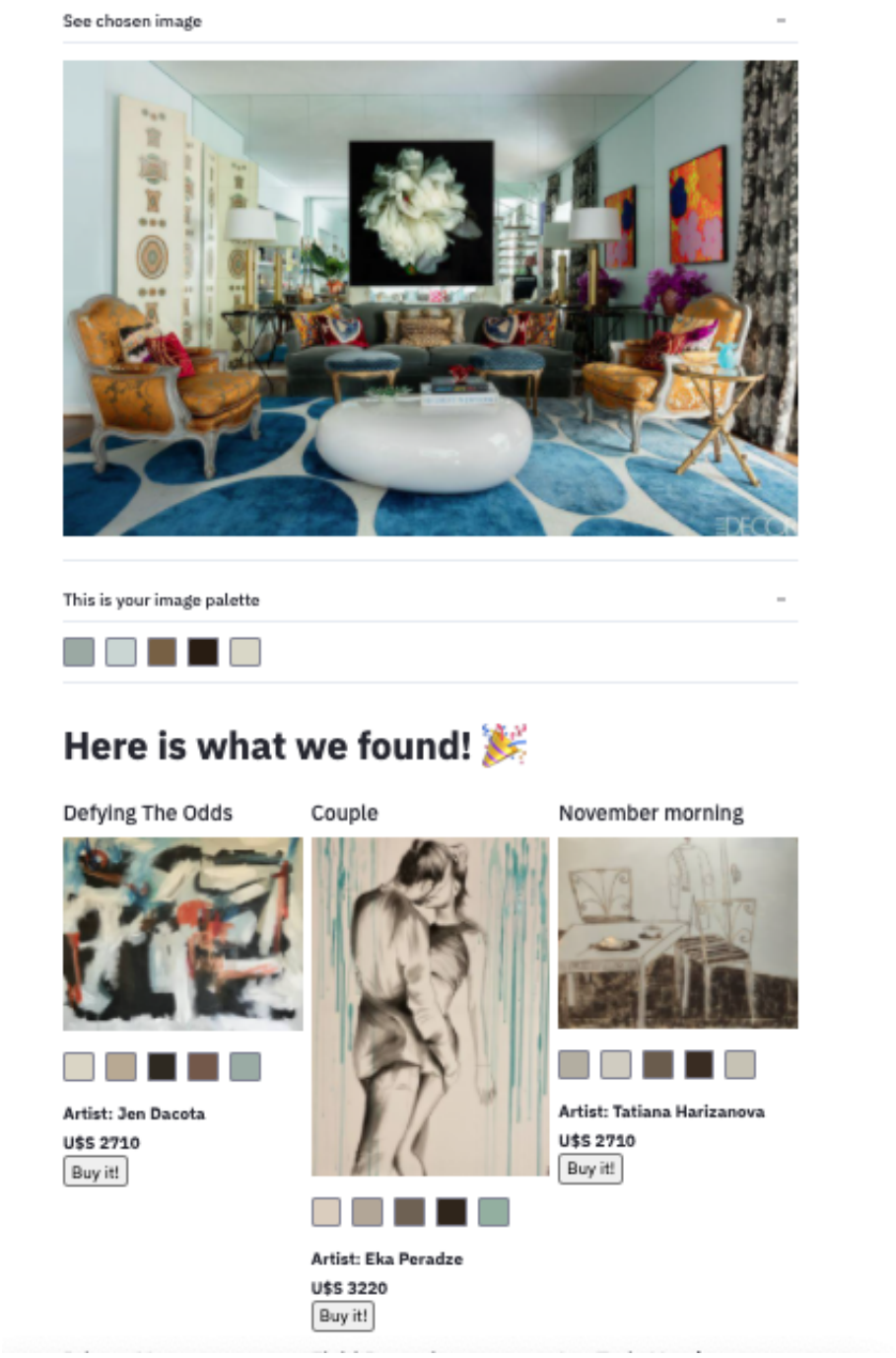
5. Evaluación de los resultados

A continuación se muestran algunos de los resultados obtenidos, mostrando la interfaz realizada y el funcionamiento de la misma.

Lo primero que tenemos es la sección para cargar la imagen que se utiliza para comparar. La misma se puede cargar desde los archivos locales o simplemente indicando su URL.

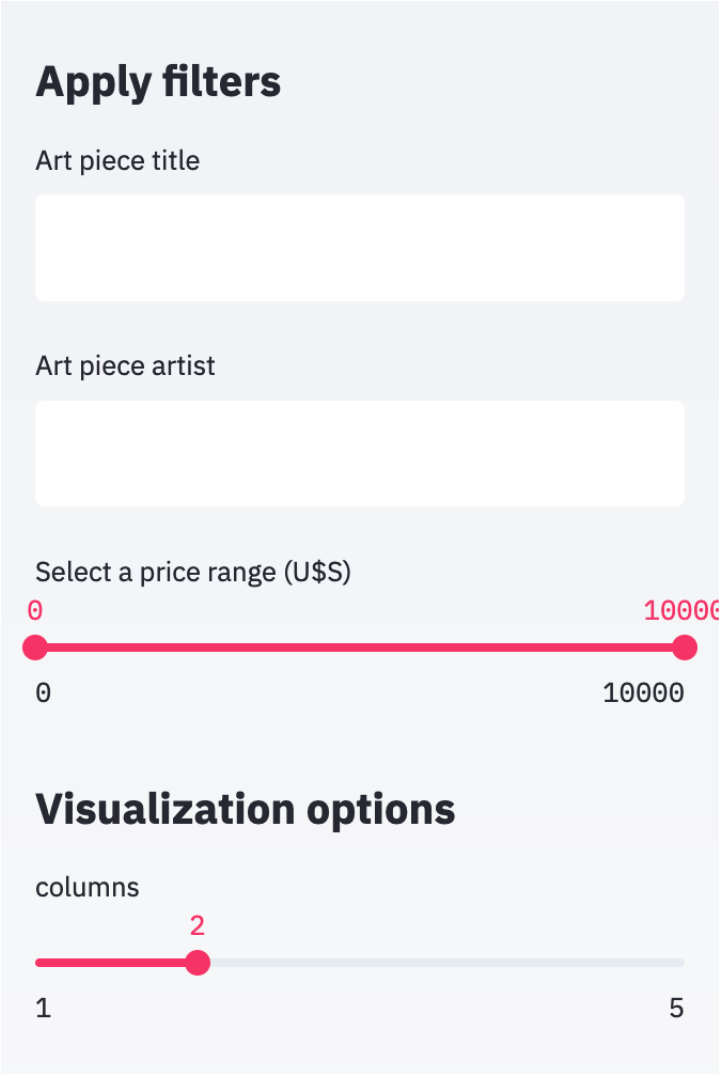


Una vez resuelto esto, la imagen se procesa en el servidor para extraer la paleta de colores que mejor se adecúa a ella, generar el vector que mejor representa dicha paleta dentro del espacio vectorial establecido, para luego consultar en el índice de ElasticSearch por aquellas pinturas que mejor se acercan a dicha paleta de colores. Una vez completado este proceso, se muestran en pantalla los resultados correspondientes a dicha búsqueda como se puede apreciar a continuación:



Aquí se puede apreciar como, a partir de la imagen seleccionada se pudo encontrar pinturas cuyos colores se asemejan a la imagen original. Como se puede ver, cada pintura, al igual que la imagen seleccionada, muestran su paleta de colores asociada y se puede observar lo similares que son entre ellas, permitiéndonos concluir que el proceso de selección funcionó de forma óptima.

Por ultimo, si se desea también se pueden agregar distintos filtros a la búsqueda como se muestra a continuación:



6. Conclusiones y trabajo futuro

Desde el punto de vista práctico, se experimentó la recuperación de información de otros medios a través de técnicas de scraping de datos, además de procesar la información obtenida para poder dar valor a los usuarios. Luego de recuperada y procesada la información se la presentó de forma sencilla e interactiva al usuario para un fácil entendimiento de la misma. Creemos que la solución propuesta a la problemática planteada es adecuada y cumple con las expectativas. Se puede observar cuan acertados son los resultados obtenidos ya que al comparar las paletas generadas para la imagen original y la de las pinturas obtenidas, el parecido entre las mismas es increíble.

Como trabajo futuro se podría extender esta idea para muchos otros ámbitos, no solo el arte. A su vez también se podría extender esta idea para otro tipo de comparaciones, no solo comparación según una paleta de colores. Siempre y cuando se tenga un modelo de IA que permita generar una representación adecuada de los datos a comparar, se podría ajustar esta idea para comparar una infinidad de cosas.

Referencias

[1] Streamlit. The fastest way to build data apps
`https://www.streamlit.io/`

[2] ElasticSearch. The heart of the free and open Elastic Stack
`https://www.elastic.co/elasticsearch/`

[3] Colorgram. A Python library that lets you extract colors from images
`https://github.com/obskyr/colorgram.py`

[4] Saatchiart
`https://www.saatchiart.com/paintings`

[5] Google Art Palette
`https://github.com/googleartsculture/art-palette`

[6] Scrapy. An open source and collaborative framework for extracting the data you need from websites.
`https://scrapy.org/`

[7] FastAPI. A modern, fast (high-performance), web framework for building APIs with Python
`https://fastapi.tiangolo.com/`