Extract dominant colors from an image with ColorWeave

4th Aug, 2015

BY DATAWEAVE

We have taken a special interest in colors in recent times. Some of us can even identify and name a couple of dozen different colors! The genesis for this project was PriceWeave's Color Analytics offering. With Color Analytics, we provide detailed analysis in colors and other attributes related to retailers and brands in Apparel and Lifestyle products space.

The Idea

The initial idea was to simply extract the dominating colors from an image and generate a color palette. Fashion blogs and Pinterest pages are updated regularly by popular fashion brands and often feature their latest offerings for the current season and their newly released products. So, we thought if we can crawl these blogs periodically after every few days/weeks, we can plot the trends in graphs using the extracted colors. This timeline is very helpful for any online/offline merchant to visualize the current trend in the market and plan out their own product offerings.

We expanded this to include Apparel and Lifestyle products from eCommerce websites like Jabong, Myntra, Flipkart, and Yebhi, and stores of popular brands like Nike, Puma, and Reebok. We also used their Pinterest pages.

Color Extraction

The core of this work was to build a robust color extraction algorithm. We developed a couple of algorithms by extending some well known techniques.

One approach we followed was to use standard unsupervised machine learning techniques. We ran k-means clustering against our images data. Here k refers to the number of colors we are trying to extract from the image.

In another algorithm, we extracted all the possible color points from the image and used heuristics to come up with a final set of colors as a palette.

Another of our algorithms was built on top of the Python Image Library (PIL) and the Colorific package to extract and produce the color palette from the image.

Regardless of the approach we used, we soon found out that both speed and accuracy were a problem. Our k-means implementation produced decent results but it took 3–4 seconds to process an entire image! This might not seem much for a small set of images, but the script took 2 days to process 40,000 products from Myntra.

Post this, we did a lot of tweaking in our algorithms and came up with a faster and more accurate model which we are using currently.

ColorWeave API

We have open sourced an early version of our implementation. It is available of github **here**. You can also download the Python package from the Python Package Index **here**. Find below examples to understand its usage.

Retrieve dominant colors from an image URL

```
from colorweave import palette print palette(url="image_url")

Retrive n dominant colors from a local image and print as json:




print palette(url="image_url", n=6, output="json")

Print a dictionary with each dominant color mapped to its CSS3 color name




print palette(url="image_url", n=6, format="css3")

Print the list of dominant colors using k-means clustering algorithm




print palette(url="image_url", n=6, mode="kmeans")
```

Data Storage

The next challenge was to come up with an ideal data model to store the data which will also let us query on it. Initially, all the processed data was indexed by Solr and we used its REST API for all our querying. Soon we realized that we have to come up with better data model to store, index and query the data.

We looked at a few NoSQL databases, especially column oriented stores like Cassandra and HBase and document stores like MongoDB. Since the details of a single product can be represented as a JSON object, and key-value storage can prove to be quite useful in querying, we settled on MongoDB. We imported our entire data (~ 160,000 product details) to MongoDB, where each product represents a single document.

Color Mapping

We still had one major problem we needed to resolve. Our color extraction algorithm produces the color palette in hexadecimal format. But in order to build a useful query interface, we had to translate the hexcodes to human readable color names. We had two options. Either we could use a CSS 2.0 web color names consisting on 16 basic colors (White, Silver, Gray, Black, Red, Maroon, Yellow, Olive, Lime, Green, Aqua, Teal, Blue, Navy, Fuchsia, Purple) or we could use CSS 3.0 web color names consisting of 140 colors. We used both to map colors and stored those colors along with each image.

Color Hierarchy

We mapped the hexcodes to CSS 3.1 which has every possible shades for the basic colors. Then we assigned a parent basic color for every shades and stored them separately. Also, we created two fields—one for the primary colors and the other one for the extended colors which will help us in indexing and querying. At the end, each product had 24 properties associated with it! MongoDB made it easier to query on the data using the aggregation framework.

What next?

A few things. An advanced version of color extraction (with a number of other exciting features) is being integrated into PriceWeave. We are also working on building a small consumer facing product where users will be able to query and find products based on color and other attributes. There are many other possibilities some of which we will discuss when the time is ripe. Signing off for now!

Originally published at **blog.dataweave.in.**

- *DataWeave Marketing*

4th Aug, 2015

BRAND PERCEPTION