**カラーベースフィルタリングのコード構成**

**TeamHTTP**

**2018.02.07**

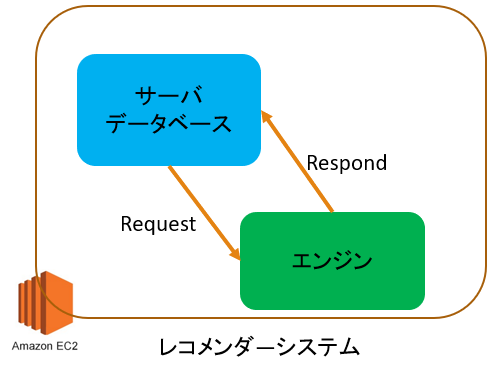
1. コードの目的

色聴共感覚を基づいたカラーベースフィルタリングを実現するために、画像の色を分析して、その色の調性格を推定する。

推定された色の調性格結果をans\_color.csvに書き込んで、サーバ側が伝えられる。

1. レコメンドの流れ

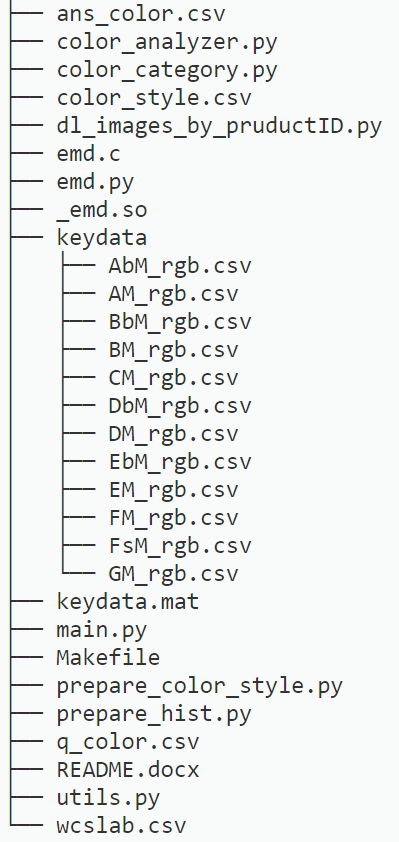
下図の関係図を示すように、レコメンダ―システムにはエンジンとサーバ、二つ部分に分けている。サーバ側がエンジンを呼び出してから、調性格結果をもらえる。



　調性格結果を取得してから、サーバ側がデータベースに同じ調性格商品をランダムに抽出して、レコメンド結果として推薦する。

1. コードの構成

下図はコードの構成、順番に説明する。ちなみに、pycファイル、GAデータなどを省略した。



├── ans\_color.csv：調性格結果をここに書き込む

├── color\_analyzer.py：画像のRGB色を抽出する

├── color\_category.py：色のMunsell上分布を引数として、色の調性格を推定

├── color\_style.csv：すべての商品の調性格、エンジンが使わないけど、調性格結果をDBに保存すべき

├── dl\_images\_by\_pruductID.py：商品IDを用いて商品の画像をダウンロード

├── emd.c：EMD距離計算のコア、直接に使わない

├── emd.py：EMD距離計算のインターフェース、Pythonで呼び出せる

├── \_emd.so：emd.cの動的リンク、直接に使わない

├── keydata：共感覚者の色聴RGBデータ(これを用いて各調性格がMunsell上の分布を求める)

│   ├── AbM\_rgb.csv：AbM調に対応するRGB色、以下略

│   ├── AM\_rgb.csv

│   ├── BbM\_rgb.csv

│   ├── BM\_rgb.csv

│   ├── CM\_rgb.csv

│   ├── DbM\_rgb.csv

│   ├── DM\_rgb.csv

│   ├── EbM\_rgb.csv

│   ├── EM\_rgb.csv

│   ├── FM\_rgb.csv

│   ├── FsM\_rgb.csv

│   └── GM\_rgb.csv

├── keydata.mat：共感覚者の色聴データ、元々丁さんのmatlabデータ

├── main.py：サーバがmian.pyを呼び出して、調性格結果をans\_color.pyに書き込む

├── Makefile：emd.cから\_emd.soを作成する、\_emd.so削除されてしまったら使う

├── prepare\_color\_style.py：初期準備段階で、color\_style.csvを作成する

├── prepare\_hist.py：初期準備段階で、./keydata/に各調性格のMunsell分布を計算

├── q\_color.csv：調性格を求めたい商品のID(複数ID可能)

├── README.docx：本ファイル

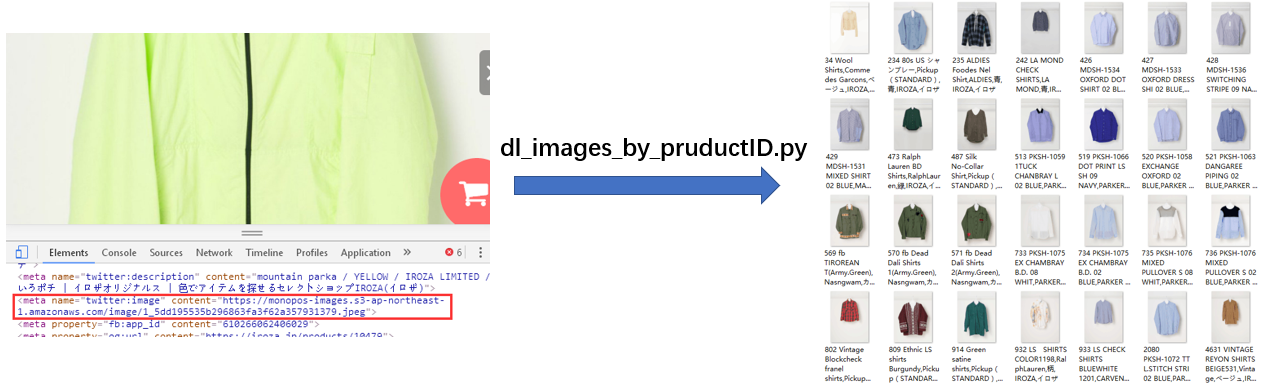
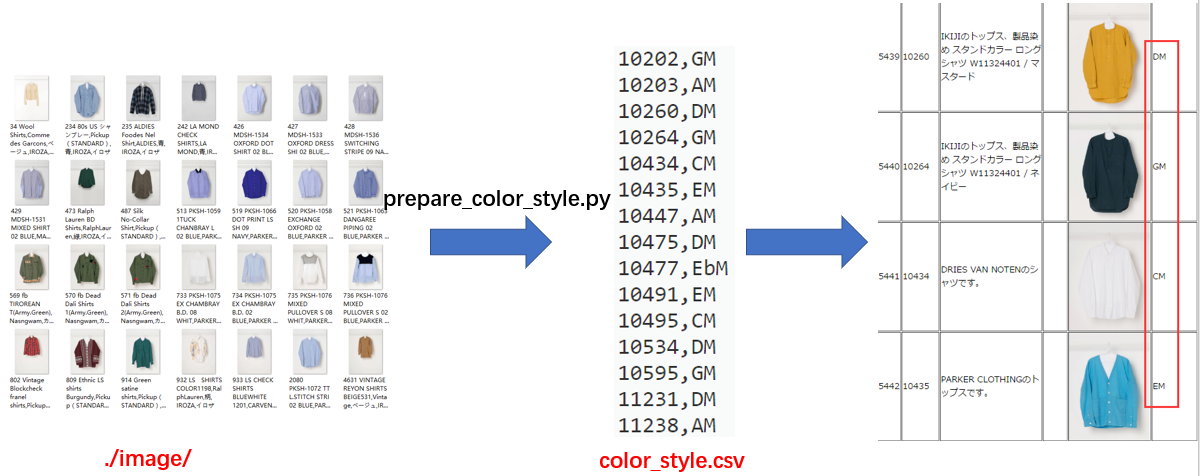
├── utils.py：いろいろ計算関数をここに整理された

└── wcslab.csv：Lab色とMunsell上座標の対応関係

1. 導入プロセス
   1. prepare\_hist.pyを使って、./ keydata/に各調性格のRGB色のMunsell上分布を計算する。

計算結果がすでにcolor\_category.pyに書かれたから、このプロセスを略してもよい。

* 1. すべての商品IDを用意して、id.jpgという形式でall\_id.csvに保存する。
  2. dl\_images\_by\_pruductID.pyを使って、all\_id.csvの商品画像をダウンロードして、./image/に保存する。
  3. prepare\_color\_style.pyを使って、./image/の画像の調性格を推定して、結果をデータベースに保存する。



1. 呼び出す

q\_color.csvに複数の商品IDを書き込んでから(id.jpg形式)、main.pyを呼び出す。

呼び出す方法は：python main.py q\_color.csv ans\_color.csv ./all\_image/ wcslab.csv