Building an Image-Based Shoe Recommendation System

Amrit Saxena
Stanford University
amrit@cs.stanford.edu

Neal Khosla
Stanford University
nealk@cs.stanford.edu

Vignesh Venkataraman
Stanford University
viggy@stanford.edu

Abstract

We have developed an image-based shoe recommendation system to help modernize the retail industry. This system assesses a dataset of over 13,000 men's shoes and extracts the shape, color, and texture associated with each of these shoes. It then recommends shoes that are similar to a user's input image on the basis of these features. On the whole, our algorithm takes seconds to compute and has a recommendation accuracy of over 84% for each of shape, color, and texture across the top 10 recommendations for each image.

1 Introduction

In response to the relatively slow rate of technological innovation in the retail industry, we set out to utilize machine learning techniques to modernize the industry. We identified real-time recommendation systems for retail goods as an area with significant opportunities for advancement. While e-commerce platforms utilize techniques like collaborative filtering to power recommendations for users, recommendations in brick-and-mortar stores are still very low-tech and are generally driven by word-of-mouth from friends and family and pitches by salesmen. The problem lies in that it is difficult to build a useful user profile vector and recommend things to people in real time as they shop physically. Even if it were possible to track this data, it is fairly difficult to compute recommendations for items in the brick-and-mortar setting due to poorly-defined taxonomies and a general lack of data properly and consistently characterizing products in retail database systems. We wanted to develop a mechanism that could provide extremely accurate and efficient recommendations without having to depend on a large amount of data points. As a result, we decided to leverage machine learning to build an algorithm that could recommend retail goods on the basis of image similarity as such a process would require no data outside of images of the goods.

Our belief is that successfully computing recommendations on the basis of image similarity would be a stepping-stone towards building more robust recommendation systems that could significantly improve the way in which people shop as it would allow for more customized, flexible, and accurate guidance in identifying desired goods. As this provides significant value to commercial entities and consumers alike, this is a problem that many people are trying to tackle in the real world right now, albeit with varying levels of success.

Acknowledging that the problem that we were seeking to solve was a difficult one, we decided to initially attempt to solve a simpler subproblem. We proceeded by assessing shoes, a retail product that is not overly variable or complex, in order to maximize our chances of substantive results. Beyond the relative lack of complexity in shoe styles, footwear is also a significant industry within the retail space and represents transactions of over $54 billion per year [4].

2 Dataset

For our analysis, we wanted to compile a dataset of images with consistent lighting and positioning so that we could focus on developing a robust algorithm without having to worry about making sense of the noise in "dirty" images. As a result, we created a dataset by scraping the data for all of the 13,035 different men's shoes that were listed on Zappos.com at the time of scraping. We chose to use images from Zappos for our project as the site contains a fairly exhaustive set of shoes among its product listings and has images with exceptionally consistent lighting, positioning, and orientation. For each shoe, we scraped a side-view image of a left-shoe facing left. Along with each of these images, we scraped data about each of the products including the targeted gender, the name of the shoe, the brand of the shoe, the shoe price, color, ratings per reviews on the website, and Zappos' provided categorical taxonomy of the shoe.

3 Process

In our efforts to create a robust recommendation system, we began by analyzing and identifying what features might define a shoe and its style. While we had initially hoped to use our data (specifically the Zappos' taxonomy of each shoe) as a metric for the success of our classification of style, we quickly concluded that Zappos provided extremely poor taxonomies. Nearly 50% of shoes

in the dataset were classified in the "Sneakers & Athletic Shoes" category. Because this provided taxonomy lacked any real granularity, we pivoted to approach this problem as an unsupervised learning problem. We then shifted our focus to determining a feature set that would enable us to accurately group similar shoes. Through a drawn out brainstorming process that involved both numerous discussions and cursory explorations of different approaches, our group concluded that the three main features that defined shoes were shape, color, and texture.

1. Shape - We identified that shoes with similar shapes tend to belong to the same style categories and therefore concluded that the shape of a shoe would give us significant fundamental insight into the style of a shoe.

2. Color - We also noted that from a visual perspective, the color of a shoe is one of the first things that the brain registers, and thus, that color might provide a good feature to base recommendations on.

3. Texture - We decided that ascertaining the texture of a shoe would make our recommendations far more accurate as shoes with similar shapes and colors can often be of vastly different styles. For example, a low-cut brown running shoe and a low-cut brown casual-wear skate shoe will have extremely similar shapes and colors, but the running shoe would have a mesh-like texture, while the skate shoe would have a much smoother texture.

Additionally, since this is an unsupervised learning problem, for each one of our accuracy calculations in this paper, we measured accuracy for a randomly selected sample of 400 shoes as this represents a statistically significant sample size at the 95% confidence level with a 5% confidence interval.

3.1 Shape

To ascertain shoe shapes, we first tried a fairly rudimentary clustering approach to determine cluster centers corresponding to different styles of shoes. By using the category taxonomies from Zappos' website, we computed prototypes for cluster centers for each one of the major categories. To do this, we converted each image corresponding to a given category to a grayscale array and computed the average grayscale array corresponding to each category. Once we had these initialized cluster centers, we ran k-means on our entire dataset of images. This technique resulted in a style classification accuracy of about 55% on a randomly-selected, statistically significant sample. As previously mentioned, we realized that one of the primary reasons for this was that Zappos' taxonomies were highly generalized and thus did not do a good job of differentiating between shoes with vastly different shapes and styles and causing this method to significantly underfit our dataset. Thus, for our next approach, we went through all of our images and found nine images that we determined were good representations of most of the distinct shapes and styles of men's shoes (i.e., low-tops, sneakers, high-tops, boots, slippers, clogs, boat shoes, sandals, and flip flops). We then clustered on the basis of these initialized cluster centers and achieved a shape classification accuracy of about 75%. Subsequently, since the only relevant factor for this step of our algorithm was shape, we decided to mask all of the images as illustrated in Figure 1. To compute these masks, we utilized OpenCV's thresholding capabilities [1]. Specifically, we applied the inverse binary thresholding function on grayscale images in order to turn the background of the image black while whiting out the contents of the shoe itself. The inverse binary thresholding function is defined by:

$$\texttt{dst(x,y)} = \begin{cases} 0, & \text{if } \texttt{src(x,y) > thresh} \\ \texttt{maxVal}, & \text{otherwise} \end{cases} \tag{1}$$

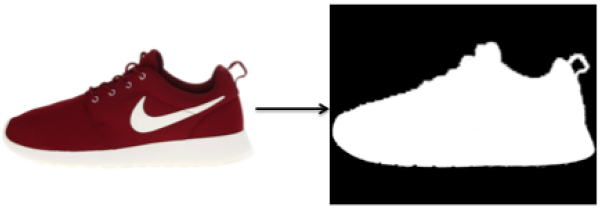

Figure 1: A mask computed for one of the shoes in our dataset.

Thus, by passing in 254 for our value of `thresh` and 255 for our `maxVal`, the background, which is entirely whitespace, is transformed to black. Similarly, any space that isn't pure white (i.e., the shoe itself) is transformed to white. Now, instead of running k-means clustering on the raw images themselves, we decided to utilize the grayscale array representations of the masks of the images that we had selected as cluster centers before and simply assigned each image in our dataset to the cluster center that had the smallest matrix difference from it as follows:

$$\arg\min_{cluster} \text{non\_zero\_count(cluster - image)} \tag{2}$$

This approach worked very well, and we achieved a shape-based style classification accuracy of about 90%.

3.2 Color

Our first attempt to create relevant color features for each shoe involved computing color histograms over the entire image for each pixel's RGB values. We then compared the histograms of all of the shoes using a correlation metric in the OpenCV compareHist function library (for an elaboration of this see Section 3.3) [5]. However, this yielded mixed results: although we were able to correctly match the primary color of the shoe, the histogram comparison was weighted too heavily by the frequency with which each color appeared. The primary color of the image was dominating the histogram's structure, hindering our ability to differentiate based on secondary colors. This suggested that our initial approach would not work, and we started to explore other methodologies.

Our second approach sought to classify the colors of the shoes by utilizing a modified form of k-means clustering to identify the top four most prevalent color cluster centers in a given image. More specifically, we ran the scipy python library's vq.kmeans clustering function for 30 iterations with randomly selected initial cluster centers until convergence, where convergence is defined to be:

$$\sum_i (R_i - \bar{R}_i)^2 + (G_i - \bar{G}_i)^2 + (B_i - \bar{B}_i)^2 \leq 0.00005 \quad (3)$$

where $(\bar{R}_i, \bar{G}_i, \bar{B}_i)$ is the cluster center that the current pixel (R_i, G_i, B_i) has been mapped to [6]. Both the cluster centers and the mappings between pixels and their closest cluster centers are returned for the iteration with the lowest overall error. Figure 2 illustrates an example output of this clustering algorithm run on a shoe in our dataset where the four most prevalent colors are shades of light grey, dark grey, blue, and white.

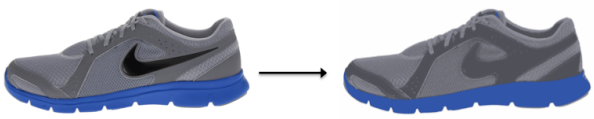

Figure 2: An example of the output from the k-means based color clustering with four cluster centers that we employed to match colors. In this image, the predominant colors are discovered to be blue, light grey, dark grey, and white.

We then mapped the RGB cluster centers into the CIEL*a*b* (L*a*b*) color space, which is a more visual color metric according to its definition. The L*a*b* color space is designed such that the mathematical distance between two points in the color space approximates perceived visual distance between two colors (i.e., how the human eye would classify their closeness) [2]. Moreover, when compared to the RGB color space, there is much greater uniformity of color spacing, meaning that the relative distances between colors become meaningful measures of ranking color similarity. The conversion from RGB into this space allowed us to calculate similarity between colors using a simple Euclidean distance function and accurately approximate similarity in color as a human would perceive it. These cluster centers were then precomputed for our entire dataset in order to optimize the speed of the algorithm for future color-based comparisons. Finally, we sorted the four color values of the cluster centers by ascending luminosity, allowing us to make an ordered comparison of the color centroids. The color similarity score of any two images in our data set was calculated to be the sum of the square of the Euclidean distance between each of the color centroids in the L*a*b* color space. Therefore, a smaller similarity score represents a smaller difference between the color distributions of two images and thus a smaller difference in their visual color similarity because of the properties of the L*a*b* color space. A randomly-selected, statistically significant sample was fed into our color similarity engine and the top-10 results for each shoe were manually verified as color matches or not. The similarity measure achieved an accuracy rate of 95%, indicating the success of our color feature construction and similarity matching.

3.3 Texture

To compute textural similarities between shoes, we first employed a texture descriptor called Local Binary Pattern (LBP). Figure 3 depicts the basic concept behind the LBP algorithm. LBP features are represented by first calculating the difference in grayscale value between each pixel in the image and its 8 direct neighbors. Then, the signs of those calculated differences are noted. Since there are 8 neighbors, each with the possibility of having a positive or negative difference, there are 256 possible LBPs. Because we applied the VLFeat library's LBP implementation to our dataset, we adhered to their standard of reducing those 256 possible LBPs to 58 quantized LBP possibilities, which depend on the transitions between negative and positive differences when the neighbors are scanned in anti-clockwise order. Reducing the number of quantized LBP possibilities to 58 allows the computation of a more reliable histogram that is ultimately the final output of this algorithm [3].

After calculating the LBP features for each of our images, we compared their texture-based similarity by utilizing OpenCV's compareHist function to calculate the correlation between two histograms [5]. This function

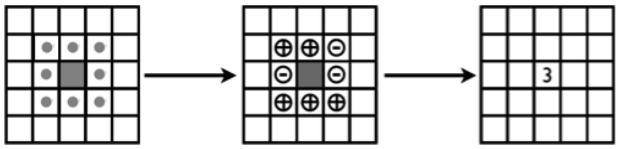

Figure 3: A demonstration of the basic algorithm behind Local Binary Patterns (LBPs) [7].

denotes the distance between any two histograms H_1 and H_2 as:

$$d(H_1,H_2) = \frac{\sum_i (H_1(i)-\bar{H_1})(H_2(i)-\bar{H_2})}{\sqrt{\sum_i (H_1(i)-\bar{H_1})^2 \sum_i (H_2(i)-\bar{H_2})^2}} \quad (4)$$

where $\bar{H_k}$ is the mean of the values in H_k or $\frac{1}{n}\sum_i H_k(i)$.

After implementing this basic version of LBPs and applying it to our dataset, it became evident that our method of comparing textures was not properly functioning. First, we observed that the comparison of texture as an additional factor in our similarity engine (along with color and shape) was having no effect on the outputted set of recommended shoes, revealing that our engine was not weighting texture correctly. However, in trying to determine how to reweight the algorithm, we discovered that almost all texture histograms for all images were being categorized as 99% or more identical. This led us to believe that our texture algorithm was not giving us any insight into differences in texture between images and thus needed to be refitted. Our first attempt to solve this problem was to switch from comparing the histograms using the compareHist function to measuring the Kullback-Leibler Divergence (KL Divergence) between the two distributions. However, our issues persisted, prompting us to reconsider our original LBP feature determinations. In order to investigate this, we temporarily edited the recommendation algorithm to recommend shoes solely on the basis of texture and were achieving less than 25% accuracy. After analyzing possible causes extensively, we concluded that our algorithm was not weighting the more granular textures in the shoes heavily enough and was just matching the significant white region of the images, which did not contain the shoe as similar. To address this, we decided to implement a modified version of Spatial Pyramid Matching to get more granular features[8]. Spatial Pyramid Matching works by creating histograms of features for progressively smaller sized subdivisions of each image and weighting the more granular histograms more when comparing two images histogram features. Figure 4 depicts the idea behind Spatial Pyramid Matching for a trivial example with 3 quantized features.

For the purposes of our texture-matching algorithm, we leveraged this concept by creating LBP histograms

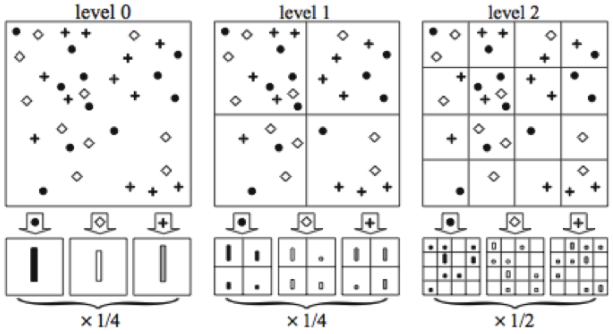

Figure 4: A depiction of the Pyramid Matching algorithm for a trivial example with 3 quantized features. The image is divided into 3 different levels of resolution for which feature histograms are computed and weighted more heavily for higher resolution subdivisions [8].

for progressively smaller subdivisions of our images for use in calculating the similarity between two images. We utilized VLFeat's functionality that allows the user to specify different resolution divisions of the image, enabling us to use the same LBP library by calculating extra histograms for finer resolutions than exclusively the entire image. Through iterative consideration of different grid sizes, we were able to settle on dividing our images into 1, 4, and 12 subregions and improved our isolated texture recommendation accuracy to approximately 85%.

3.4 Final Algorithm

After identifying these features and optimizing their extractions, we worked to develop our algorithm to compare shoes based on the similarity of these features. Initially, we hypothesized that our data and the problem we were attempting to solve would be conducive to a clustering algorithm based on the features of each shoe because it would allow us to find the intrinsic groupings or structure in the data if it existed. Unfortunately, we quickly discovered that such a structure did not seem to exist. Our attempts to run k-means on our feature set yielded very poor results as the clusters determined did not result in recommendations that humans would label as similar shoes and instead resembled what seemed to be an almost random assortment of shoes. As a result, we abandoned the more traditional machine learning clustering-based approaches and developed our own filtering-based algorithm for determining similarity, as shown in Algorithm 1.

4 Results

Because of the unstructured nature of this problem, determining a system for measuring the accuracy of our

ALGORITHM 1: Shoe Recommendation Algorithm

load *image*
load *image_mask*
for *each shoe*! = *image in dataset* **do**
    $shape\_scores[shoe] \leftarrow$
    $nonzero(image\_mask - shoe\_mask)$
end
for *each shoe in top fourth of shape_scores by num of shoes in the same cluster center* **do**
    $texture\_scores \leftarrow$
    $compareHist(image\_LBPs, shoe\_LBPs)$
end
for *each shoe in top third of texture_scores* **do**
    $overall\_scores \leftarrow \sum(color\_distance)^2$
end
print top 10 *shoes* from *overall_scores*

recommendation engine was not possible in a traditional sense. Thus, in order to appropriately evaluate our system, we calculated accuracy by simulating the way a user might judge the accuracy of our system. To do this, we ran through a randomly selected, statistically significant sample of images in our dataset and examined the 10 best matches for each of these images. We then judged each of the recommendations on whether it closely matched the original shoe in each category of shape, color and texture, and computed a score from 0-10 for each of these categories based on the number of shoes in the top-10 that closely matched the original shoe in the respective category. In other words, for each category of each image, we had a score that was $\frac{x}{10}$ where x was the number of recommended shoes that closely resembled the original shoe's traits in that category. We then averaged the ratings for all the shoes in order to receive an accuracy score for our algorithm in suggesting similar textures, colors, and shapes. Thus, in total, our accuracy for each attribute was

$$\frac{\sum \text{recommended shoes that matched category}}{\sum \text{total recommendations}} \quad (5)$$

Using this method, we were able to determine our recommendation engine achieved 90% accuracy in suggesting similarly shaped shoes, 84.03% accuracy in suggesting similarly colored shoes, and 85.97% accuracy in suggesting similarly textured shoes. Figure 5 demonstrates a standard output from our recommendation system.

5 Conclusion

As briefly alluded to in our results section, our initial results are encouraging in that they suggest that we have developed a fairly successful first-cut for attacking this problem. However, we also have room for improvement. Beyond improving the 10-15% recommendation error rate in each of our categories, our algorithms are not ready for use with images that are not as clean as the ones in our dataset. We believe that this problem may be significantly more difficult when dealing with data that is less homogenous and we could face serious challenges in dealing with images involving inconsistent lighting, positioning, or orientation. In particular, our algorithm for shape is quite dependent on positioning and orientation, while our algorithms for color and texture are particularly dependent on lighting and would require similar orientation for any sort of reasonable comparison between two images. As such, future extensions of our work include working with noisy images, extending our analysis to other types of retail goods (e.g. shirts, pants, jackets, etc.), and potentially exploring other algorithmic approaches such as deep neural networks. In the coming months, we look forward to making our similarity score calculations more accurate, dealing with noisy images, and building out a mobile application with this technology that people can use to help them shop in the brick-and-mortar setting.

Figure 5: This is a randomly selected output of our recommendation system where the shoe on the left is the original image that the user queried and the 3 shoes on the right are the top-3 recommended shoes.

References

[1] *Basic Thresholding Operations - OpenCV Documentation*. http://docs.opencv.org/doc/tutorials/imgproc/threshold/threshold.html.

[2] *CIELAB Color Models Technical Guide*. http://dba.med.sc.edu/price/irf/Adobe_tg/models/cielab.html.

[3] *lbp.h File Reference*. http://www.vlfeat.org/api/lbp_8h.html.

[4] *National Shoe Retailers Association*. http://www.nsra.org/?page=About_Us.

[5] *OpenCV Histogram Documentation*. http://docs.opencv.org/modules/imgproc/doc/histograms.html.

[6] *SciPy Cluster VQ Kmeans Documentation*. http://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.kmeans.html.

[7] PING CHUAN WANG, STEPHEN MILLER, M. F. T. D., AND ABBEEL, P. Perception for the manipulation of socks.

[8] S. LAZEBNIK, C. S., AND PONCE., J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York, June 2006, vol. II, pp. 2169-2178.*