**Reproduction Study of “Relative Attributes”**

Joseph G. Ellis

Columbia University – Spring 2013 – Reproducible Research

Original Work by: Devi Parikh and Kristen Grauman

*Devi Parikh and Kristen Grauman, “Relative Attributes”, International Conference on Computer Vision (2011).*

1. **Introduction**

“Relative Attributes” has quickly become a seminal paper within the field of Computer Vision and Visual Search and Retrieval. This work was given the Marr Prize for Best Paper at the International Conference on Computer Vision in 2011, and has since been sited almost 50 times in 2 years. The work presents a novel and intuitive framework for zero-shot learning, which is learning to classify or retrieve an image that a computer system has never seen before. This is obviously a very difficult task, but the author’s propose a framework that is both novel and intuitive. However, some very simple adjustments can be made to their framework that improves zero-shot learning performance with no added computational time. These modifications will be shown, and examples of the possible gains in zero-shot learning classification on a dataset will be shown. The ICERM guidelines for reproducibility will also be used describe the relative ease of reproducibility of this work, and suggestions to make this work more easily reproducible will be made.

1. **Relative Attributes Theoretical and Computational Overview**

We begin by giving a brief overview of the computational work completed within the “Relative Attributes” paper. Each of the sections below will be discussed as to their relative ease of reproduction later in Section 3.

**2.1 Dataset Collection**

The paper first details the use of two distinct datasets to test their zero-shot learning experiments on. The two datasets are the Outdoor Scene Recognition (OSR) dataset, and the second is a subset of the Public Figure Dataset (PubFig). Each dataset was made readily available by the author’s on the paper website (<http://filebox.ece.vt.edu/~parikh/relative.html>). The OSR dataset contains images 2,688 images from 8 different scene categories described as coast, forest, highway, inside-city, mountain, open-country, street, and tall-building. The subset of images used from the PubFig dataset contains pictures of Alex Rodriguez, Clive Owen, Hugh Laurie, Jared Leto, Miley Cyrus, Scarlett Johansson, Viggo Mortensen, and Zac Efron. This subset of the dataset used contains 800 images, one from each of the celebrities mentioned above.

**2.2 Feature Extraction**

To make each of these images useable in the ranking formulation and framework presented below in section 2.3 feature descriptors must be extracted for each of these images. For the OSR dataset the 512-dimensional Gist descriptor was used. This is a very popular feature descriptor in the computer vision field, and is widely used in a variety of applications. The features extracted for the PubFig dataset were the Gist descriptor and a 30 dimensional Lab color histogram concatenated together to create the feature descriptor for the PubFig dataset images.

**2.3 Creating Relative Attributes**

Each of the image categories then must be ranked against one another as to how much they exhibit a given attribute. For example, an image from the “forest” category is deemed as exhibiting more of the “natural” attribute than an image from the “tall-building” category. The OSR dataset is described by 6 attribute categories: natural, open, perspective, large-objects, diagonal-plane, and close-depth. The PubFig dataset images are then described using 11 attributes, which are: masculine-looking, white, young, smiling, chubby, visible-forehead, bushy-eyebrows, narrow-eyes, pointy-nose, big-lips, and round-face. A relative ordering for image categories for each attribute within is then created. The relative orderings for both the OSR and PubFig Dataset can be seen below in Table 1 and Table 2. These are the relative orderings that were used by the author’s in all of their experiments and tests.

Table 1. Relative Ordering of Attributes for OSR Dataset



Table 2. Relative Ordering of Attributes for PubFig Dataset



**2.4 Learning Relative Attributes**

Given the relative ordering of the attributes in Tables 1 and 2 the author’s propose to create a ranking framework to learn the ranks above based on the descriptor of any given image. To do this the author’s proposed a novel implementation of RankSVM to learn the ranks with imposed similarity constraints. The formulation presented in the paper is as follow. We are given a set of training images *I = {i}*, that are each described by a feature vector ***xi***, and a set of attributes *{am}*. Based on the relative ordering of attributes for each image category we can also deduce some constraints, namely we create a set of ordered pairs of images *Om* = {(*i,j*)} and set of unordered pairs of images *Sm* = {(*i,j*)} such that , i.e. image *i* has a higher attribute rank than image j for attribute*am.* Similarly, , i.e. image *i* and *j* have the same attribute rank for a given attribute *am*. Given these constraints the goal here is to learn a ranking function,

( 1 )

such that for a given set of images, we can recreate the relative ordering described above. To do this our ranking function must satisfy the following constraints:

( 2 )

This problem is NP-hard and intractable, but we can compute an approximation using non-negative slack variables, which makes this work very similar to SVM classification. The author’s use the RankSVM formulation proposed in [Ref], and to do this they use the freely available implementation created by Olivier Chappelle, which can be found at <http://olivier.chapelle.cc/primal/>.

**2.5 Zero-Shot Learning Framework from Relative Attributes**

Once the relative ranks are learned the author’s then discuss a simple framework for zero-shot learning classification based on these learned attributes. We assume that for the zero-shot learning task we have a list of seen and unseen image from which we want to perform classification. We want to compare the unseen image classes to those image classes that are seen. To do this the author’s presented a simple framework to describe the unseen images based on their seen counterparts. For each seen image class the author’s find the mean learned ranking score (*um*) for each attribute *am*, and with these learned ranking attributes create a mean vector for all attribute rank scores ***ui***, where I represents the image class (forest, tall-building, etc.). The covariance matrix for all of these attribute rank scores is also estimated and denoted as ∑i. These parameters are then used to model the scene classes as a Gaussian distribution such that each seen class can then be described as,

( 3 )

Given the relative ordering of the classes based on their attributes and the found seen class distributions the author’s attempt to generate distributions for the unseen classes. Their framework is presented below, and is reprinted directly from the paper so as to minimize any confusion.

1. If is described as > ­ > , where and are seen categories, then we set the m-th component of the mean to .
2. If is described as > ­ , we set to , where *dm* is the average distance between the sorted mean ranking scores ’s for seen classes for attribute *am*. It is reasonable to expect the unseen class to be as far from the specified seen class as other seen classes tend to be from each other.
3. Similarly, if is described as ­ > , we set to .
4. If *am* is not used to describe , we set to be the mean across all training image ranks for *am*, and the *m*-th diagonal entry of to be the same.
5. In the first three cases, we simply set .

Then for a test image *i,* the ranking scores are computed, and denoted as ***xi***. We classify this image as the class that returns the maximum probability based on the given Gaussian parameters of each class.

1. **Reproduction of Experiments**

This section discusses the ability to which I was able to reproduce the experiments performed by the author’s in “Relative Attributes”. The section also highlights the actions that I performed to attempt to recreate this work. All of this work was completed on an HP Z620 Workstation with 8GB of RAM. Matlab was the software that was used for all computation throughout this experiment.

**3.1 Dataset Collection**

Collection of the datasets used for this work was accomplished with relative ease. The author’s made their datasets freely available, and packaged them with information that was very useful in working with these datasets. The datasets were downloaded from the author’s website at <http://filebox.ece.vt.edu/~parikh/relative.html>. Each dataset also came with a “.mat” data file that stored the class of each image, which images were used for training the ranking algorithm, the learned rank weights, the extracted features from each image, and the learned ranking score for each image. This data was readily available and easily accessed. I was able to successfully complete this portion of the reproduction effort with little problems.

* 1. **Feature Extraction**

The next step in the processing pipeline after collecting the dataset was to extract features form the images to utilize the ranking formulation proposed in section 2.4. The GIST descriptor was used as the feature descriptor for the scene images, and the PubFig dataset images were described using a concatenation of the Gist descriptor and a 45-bin Lab color histogram according to the paper. However, after collecting the data from the author’s website I found that the images in the PubFig dataset were described by a feature vector that was 542-dimensional within the provided “data.mat” file. The GIST descriptor is 512-dimensions long, and therefore I assumed that instead of a 45 bin Lab Color Histogram the author’s instead used a 30 dimensional Lab color histogram concatenated onto the GIST descriptor to create the feature vector for the PubFig dataset images.

The most commonly used implementation of the GIST descriptor provided by the original author takes some parameters as input besides simply the image. No parameters are stated within the relative attributes paper discussing how to extract GIST descriptors from the images. However, I used the extraction procedures detailed in the GIST author’s tutorial which can be found at <http://people.csail.mit.edu/torralba/code/spatialenvelope/>, to perform my feature extraction. The results were satisfactory, and I was able to extract features that were very similar to those provided by the author’s. I compared my extracted features to the author’s using the formula below in equation (4), where ***x****i*is the feature descriptor I extracted from image *i* and ***y****i*is the feature descriptor provided by the authors. The percent difference results between my features and the author’s can be seen in Table 3.

( 4 )

Table 3. Percent Difference of Features Extracted

* 1. **RankSVM Computation**

After feature extraction the next step to conduct the author’s experiments was to implement their ranking formulation provided in section 2.4 of this paper. To do this I took the freely available Matlab implementation of RankSVM created by Olivier Chappelle, and modified the ranking formulation to take similarity constraints as input. This was the same procedure that was outlined in the paper and pursued by the authors. The authors have also made this portion of code freely available, and I have compared my implementation to theirs. Our results compared very favorably and it can be deduced that our ranking algorithms are doing the same thing.

Table 4. Percent Difference of Features Extracted



* 1. **Training Scheme for Experiments**

The author’s also developed and detailed the training scheme that was developed for their experiments in Section 4.2 to learn the ranking scores from a given set of training images. The description of their training scheme can be seen in the excerpt below taken directly from the paper.

*“Unless speciﬁed, we use 2 unseen and 6 seen categories. To train the ranking functions, we use 4 category pairs among seen categories, and unseen categories are described relative to the two closest seen categories for each attribute (one stronger, one weaker). We use 30 training images per class, and the rest for testing”*

In my personal opinion this statement is highly ambiguous. There are multiple ways that the above statement could be interpreted and I therefore attempted multiple ways of choosing the O and S set constraints described in section 2.4 of this paper. A complete list can be seen below, including the last member of the list, which is the way that the author has stated they trained this ranking function. I began an email correspondence with the author, and only after this email correspondence was I able to deduce the proper way to select the constraints to train the ranking scores for the images.

* Create 4 category pairs from the seen classes, and then compare each of the 30 training images from the first category in each pair to the second category in each pair for every attribute. This gives us a total of 30 constraints per category pair.
* For each seen category, randomly choose 4 other seen classes and compare each image in the seen category to one image from each of the other 4 seen classes. This gives 120 constraints per image category.
* For 4 seen categories, randomly choose 4 other seen classes and compare each image in the seen category to one image from each of the other 4 seen classes. This gives 120 constraints per image category.
* Create 4 unique categories pairs from the seen classes, and then compare each of the 30 training images associated with the first category in the pair to each of the 30 training images from the second category in the pair. This result in 900 constrains per image category.

The author of the paper has also quickly validated my code and work and stated that it looks fine, and she believes this is also how she trained her ranking function. However, the ranking scores that I have found using this training scheme are slightly different than the ranks that are provided with the data by the author’s. The results and differences can be seen in the figures below, and correspond to the OSR dataset. The results presented below were found with no unseen classes, because we assume that the learned ranks provided with the author’s dataset were learned with no unseen classes, or this would have been expressly stated. All of these results were performed given the extracted features from the author to minimize the impact of all possible variables that could influence the ranking outcome except for the training scheme. Figure 1 shows the results of their learned ranks in blue, and my learned ranks in red with different numbers of training pairs.

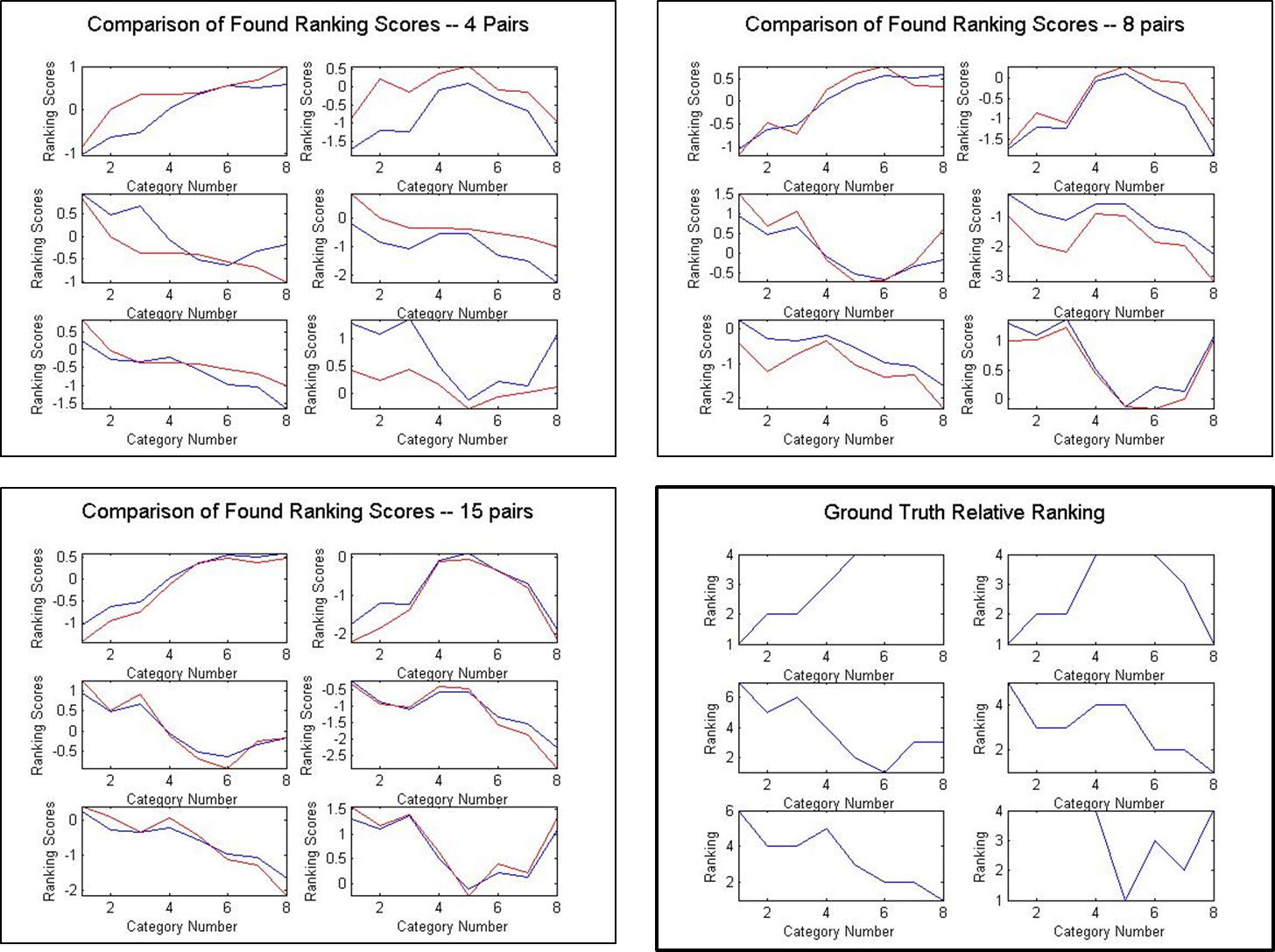


Figure 1. Accuracy of Found Ranking Scores

The shape of the desired curve based on the ground truth ranks of the objects seen in the Tables 1 and 2 can be seen in the plot in the bottom right. Each ranking function then in turn is desired to be similar to this ground truth ranking. We can see that when only using 4 pairs for training of the ranks our values are very different than the learned values given by the author with their data. However, once we use 15 pairs for training of the weights and learning the image ranks we have very similar plots. These are the results presented for the OSR dataset. This leads me to believe that to have strong classification accuracy of the unseen categories it is important to have many pairs of image categories for training. However, the author’s show that more training pairs is not incredibly helpful for classification; this contradicts what I have found.

* 1. **Final Results of Papers**

In this section the final results from my pipeline are presented and compared to the results that were given within the paper by the authors. Within the paper there were no actual values given for classification accuracies, simply graphical plots of the accuracies for different circumstances. Therefore, I have done my best to estimate what these values were for each plot within the paper and the presented results can be seen in the figures below.

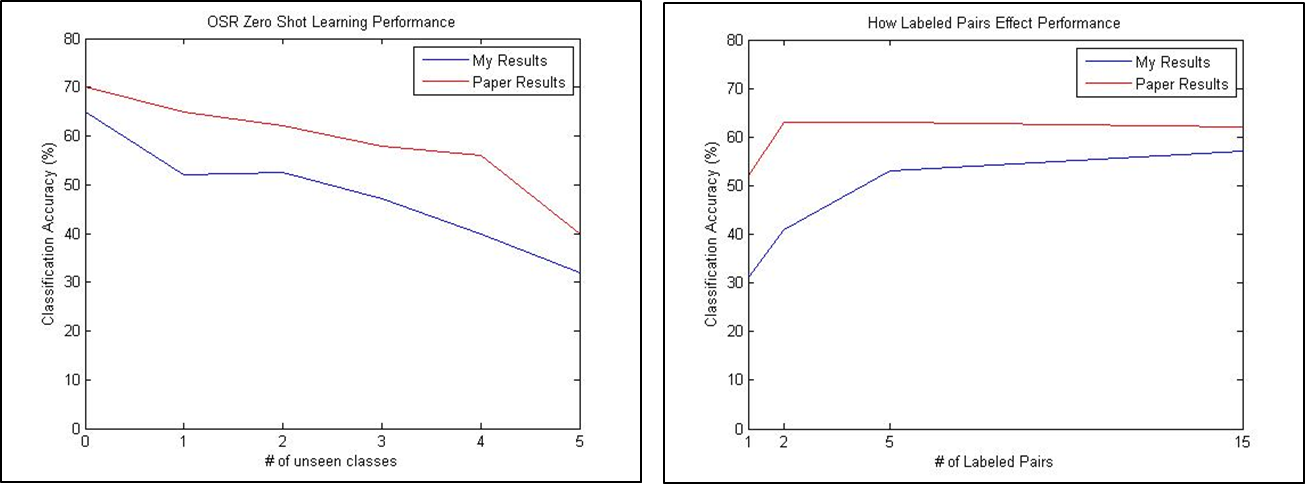


Figure 2. OSR Results – Classification Accuracy and How Labeled Pairs Effect Performance

Figure 2 shows how closely I was able to replicate the results of the author’s on the OSR dataset, and then we can see here that the general curvature of the accuracies between their experiments and mine are very similar. However, we can see that my results are slightly lower than their reported performance for both metrics in Figure 3. The most telling plot is the one entitled “How Labeled Pairs Effect Performance”, we can see that as the number of pairs used to train the ranking functions increase my classification accuracy increases and is very close to the presented accuracy within the paper. This agrees with my results presented in Figure 1, showing that many training pairs is imperative for learning a ranking function that resembles the ground truth relative ranks of the images. Each of these tests as stated in the paper were done with 6 seen categories, and 2 unseen categories and the ranking functions were trained with 4 category pairs, unless implied by the graph otherwise.