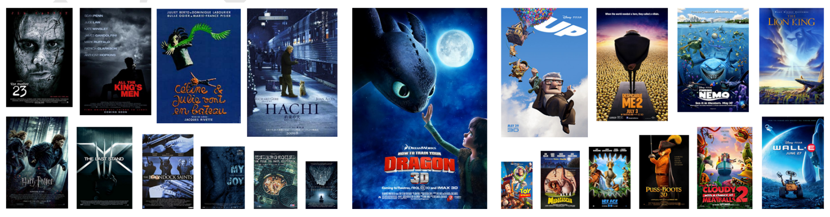
Toward Semantic Image Similarity from Crowdsourced Clustering

Yanir Kleiman George Goldberg Yael Amsterdamer
Daniel Cohen-Or

Abstract

Determining the similarity between images is a fundamental step in many applications, such as image retrieval and image browsing. Automatic methods for similarity estimation often fall short when semantic context is required for the task, raising the need for human judgment. Such judgments can be collected via *crowdsourcing* techniques, based on tasks posed to web users. However, to allow the estimation of image similarities in reasonable time and cost, the generation of tasks to the crowd must be done in a careful manner. We observe that distances within *local neighborhoods* provide valuable information that allows a quick and accurate construction of the global similarity metric. This key observation leads to a solution based on clustering tasks, comparing relatively similar images. In each query, crowd members cluster a small set of images into bins. The results yield many relative similarities between images, which are used to construct a global image similarity metric. This metric is progressively refined, and serves to generate finer, more local queries in subsequent iterations. We demonstrate the effectiveness of our method on datasets where ground truth is available, and on a collection of images where semantic similarities cannot be quantified. In particular, we show that our method outperforms alternative approaches, and prove the usefulness of clustering queries, and of our progressive refinement process.



(a) (b)

Figure 1: Nearest neighbors of the center image in a collection of movie posters, computed using (a) image descriptors, and (b) crowdsourced queries. Smaller images mark farther neighbors.

1 Introduction

In recent years, there have been many advances in image capturing capabilities of mobile devices, encouraging end users to capture more images of higher quality. As a result, there is an abundance of constantly growing collections of images, both on personal computers and on websites such as Facebook, Flickr and Instagram. Such vast collections require efficient methods for image retrieval and image browsing, which allows users to quickly locate images suitable for their needs. An important component in such methods is a similarity metric between images, which should be aligned with user perception and intuition.

State-of-the-art analytical methods for computing image similarity metrics fall short when user intuition is based on a broad *semantic context*. This may include elusive relations such as a similar emotion or sensation evoked by the images (e.g., images that convey "fear" or "comfort"); images of things which are semantically related (e.g., different types of garden furniture); likeness between the photographed people; and so on. Consider, for instance, the similarity between the movie posters in Figure 1. Identifying such similarities is usually easily done by a human observer, but poses a hard computational problem nonetheless.

The natural solution is thus gathering information about semantic similarities between images from people, for example using a crowdsourcing technique.[1] This approach was taken in recent work [8, 10, 12] to collect style similarity measures. The typical comparison task that the crowd performs is of the following form: given three images A, B, and C, choose whether A is more similar to B or to C (a *triplet query*). Assuming consistent query responses, querying every image triplet yields the full relative similarity metric over the set of images. However, the number of triplets is prohibitively large. Thus, typically only a sample of the triplets are queried and the rest are estimated based on extracted image features [8, 10, 12].

Another challenge in this respect is that people often need *context* in order to perform comparison tasks. For example, consider the triplet in Figure 2. Is the image of a bridge in London (b) more similar to another image of a different bridge in London from a different angle (c) or to an image of a Parisian bridge from the same angle (a)? In a larger context, it often becomes clearer which option is more reasonable, e.g, in the context of Figure 3 image (b) is more similar to (c) than (a).

In this work, we propose an alternative approach for learning image similarities based on *clustering queries* posed to the crowd. Instead of queries of three images, crowd members are given a small set of images and are asked to cluster them into bins of similar images using a drag-and-drop graphical UI (see Figure 4). While a single clustering task requires more effort than comparing three images, our approach has two important advantages. First, the results of a single clustering task provide a great deal of information that is equivalent to many triple comparison tasks: images placed in the same bin are considered closer to one another than to images in other bins. Second, each query provides crowd members with additional context that assists them in performing a more faithful and meaningful comparison.

[1] *Crowdsourcing* is a general name for processes that involve posing many small-scale tasks to the crowd of web users, and piecing together the crowd's answers to fulfill a larger-scale task, such as constructing a large knowledge base.

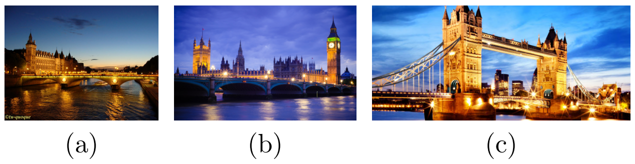


Figure 2: Without context, similarity between images can be ambiguous.

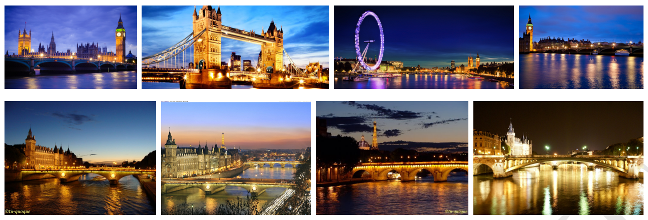


Figure 3: With context, it can be seen that images belong to two distinctive locations, London and Paris.

A key observation of this work is that a similarity metric can be constructed more efficiently by performing comparisons on similar images rather than non-similar ones. This is true in particular in the context of semantic similarities, where local similarities are oftentimes more meaningful. We build on this observation in the development of a novel, adaptive algorithm that aims to generate queries that are as local as possible. The challenge here is that similarities are of course unknown in advance. Thus, our algorithm works iteratively, such that at each phase we generate and pose to the crowd some clustering queries. At first, given no prior knowledge about image similarity, one cannot do much better than posing random queries. But as information is collected, our algorithm progressively refines the queries to focus on similar images in a narrower local neighborhood. The local similarity comparisons collected at each phase are embedded in Euclidian space to obtain a refined estimation for the global similarity metric. This refined metric is then leveraged for computing more locally-focused queries in the next phase. The progressive method thus efficiently converges to a meaningful similarity estimate.

Evaluation and experimental study. To test the efficiency of our approach, we implement our technique in a prototype system, and use it to conduct a thorough experimental study, with both synthetic and real crowd data. First, we test our technique over two image datasets where the ground truth is known, examine the results and compare them to a baseline approach that uses the same number of queries but chooses them randomly. Second, we compute the k-NN images for real-world image datasets, where the ground truth is unknown, and evaluate the results manually. Last, in order to isolate and examine the effect of different parameters of the problem, such as number of phases and queries, we vary these parameters in a series of synthetic experiments. Our experimental results prove the efficiency of our approach for computing semantic image similarity based solely on the answers of the crowd, while using a relatively small number of clustering queries.

2 Related Work

The classification of images is a well-studied problem. A common paradigm is based on image descriptors, such as the color histogram of images, SIFT based descriptors [7], or GIST descriptors [11]. The distance between two images is then defined as the Euclidean distance between the image descriptors, on top of which different machine learning techniques can be employed to find similarities or clusters of the images (e.g., [17, 23]). Other methods employ a bag of features (BoF) approach, using either visual segments [14] and/or textual annotations, either attached to the images manually or from the textual context of a web page (e.g., [17, 23]). However, such methods fall short when classification relies on semantically-rich features, which may be hard to learn from the images, and may only be partially reflected in the labels. For instance, the images of London and Paris bridges contain many semantic features such as the situation, style, building materials and general atmosphere. The images and labels describing them may not capture all of these features, nor their relative importance for determining similarity.

The problem of lacking semantic features can be alleviated by semi-supervised learning methods that rely on manual labeling of a small set of *image pairs or triplets*, rather than per-image labels for the entire set. A large body of work has attempted to classify images using such methods, typically, by pair-wise labeling consisting of equivalence (and sometimes inequivalence) constraints, i.e., whether (or not) the pair belongs to the same class [1, 2, 19, 21]. Triple-wise constraints are more relevant to *relative* comparisons of images, and as explained in the Introduction, they compare the distances of two image pairs [5, 8, 10, 12, 16]. The constraints can then be used to learn a distance metric between images. In particular, the work of [16] focuses on adaptively selecting optimal triplets based on crowd input. In the recent work of [8, 10, 12], use triple-wise comparisons in the spirit of [16] in order to learn about style similarities from the crowd. While these studies highlight the need in crowd-provided similarity comparisons, the use of triplet comparisons has shortcomings that our work addresses, as mentioned in the Introduction: the triple-wise approach requires many crowd tasks, and users are not given context for comparison. These shortcomings were also noted by [20], a study that focuses on redesigning the user interface to derive more image comparisons from each crowd task. This is done by asking users to select the X most similar images to a given image, out of Y images. The new interfaces of [20] form a step forward from triplets, but in contrast with our work, their study does not consider how to effectively choose images to compare.

Another work highly related to ours is *Crowdclustering* [6], which considers clustering images with the crowd, as follows. Each crowd member obtains a sample of a few images (a *query*) and classifies them into groups. This input is used to train a Bayesian model which captures the ways in which different crowd members may classify each image. This work resembles our in letting the user cluster a small set of images, and also in the idea of refining the clustering results by re-applying the technique on the obtained clusters. However, their technique is not designed to compute image similarities. In contrast, the progressive refinement that we employ allows converging faster to determining image similarities. We compare the performance of our techniques with [6] in Section 4.

In [6] images can only be classified if they appeared in at least one query, thus, the work of [22] suggests to only obtain query answers for a small fraction of the data, and use dedicated matrix completion techniques to complete the missing classifications. This work is orthogonal to ours, and can be employed in our case if the number of queries that can be asked is small relative to the number of images.

Crowdsourcing has been employed for other tasks related to ours such as record matching based on images [9], grouping and top-k [4], and entity matching [18]. However, none of theses solutions can be applied in an effective manner for estimating image similarities. E.g., k-NN may be viewed as finding the top-k most similar images for each image; however, applying the method of [4] for each image separately is inefficient.

3 Algorithm

We next describe our method of generating queries to the crowd based on an estimated similarity metric, and of refining the similarity metric based on answers from the crowd. In light of our key observation from the Introduction, queries involving images from the same local neighborhood are indeed more effective for determining the global similarity metric.

The queries that our algorithm generates are clustering queries, namely, the algorithm selects sets of n_q images. The answer obtained from crowd members is a division of this image set into n_c clusters. The crowd is a relatively expensive resource in terms of latency, monetary cost (if crowd members are paid to answer the queries) and human effort. Therefore, in many practical cases, the total number of queries that can be asked is restricted by a predefined budget. Given such a budget, the goal of the algorithm we develop is to utilize the queries in the best way possible, by considering only local neighborhoods. This inevitably yields an iterative process, where local neighborhoods change according to queries results.

Our method estimates local distances by maintaining an embedding in a Euclidian space of the entire set, in which the distances are calculated. The embedding is initialized randomly, and local neighborhoods are progressively improved. The embedding ensures that even distances that were not queried are consistent with the partial information derived from queried distances. To improve the embedding of local neighborhoods, we pose queries to the users in small batches, and update the embedding after each batch. Interestingly, querying local neighborhoods of the embedding proved beneficial even in early stages when the images are not necessarily semantically close, since such queries provide many constraints on the same neighborhood. In addition, in each iteration we wish to preserve the close neighbors which are already semantically similar. Even in a random embedding, local neighborhood based queries help detect cases where some neighbors are also semantically similar and maintains them.

The main steps of the algorithm are illustrated in Algorithm 1: As input, the algorithm takes `budget` – the total number of allowed queries and `batch_size` – the number of queries to be generated at every iteration. The results of the queries are integrated into the embedding (**E**) and the induced global distance metric (**D**). The output of the algorithm is the distance metric computed based on the last, most refined embedding.

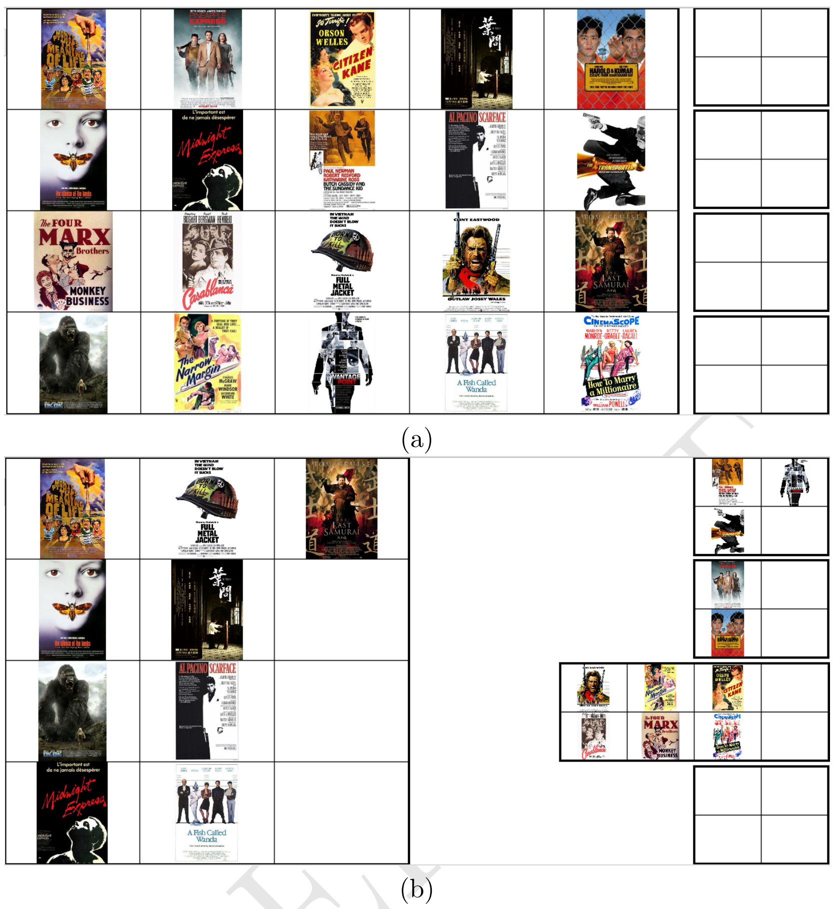


Figure 4: An example of the clustering interface. (a) The user is presented with 20 images to cluster into the four bins on the right. (b) The bins may contain as many images as necessary. When all images are clustered, the user can submit the query and receive another one.

Clustering query. We define a crowd query and its answers as follows. For the full set $\mathcal{I}$ of images, a query Q is a subset of $\mathcal{I}$ containing n_q images. The crowd's answer is a division of Q into disjoint clusters $C_1, \ldots, C_{n_c} \subseteq Q$. From these answers we extract similarity comparisons: given two images x, y in cluster C_i, and a third image z in a different cluster C_j, we infer that $\Delta(x, y) < \Delta(x, z)$, where Δ represents the similarity metric. Therefore, as n_q increases, we obtain more comparisons. However, the number of images in the query should be small enough to allow a crowd member to view them [9]. In our experiments, we found that $n_q = 20$ is a good balance of this tradeoff between effectiveness and simplicity. Following this, we found that setting the number of clusters n_c to 4 is optimal, since it balances between inferring more comparisons (smaller n_c values) and quickly pruning the less similar images (larger n_c values).

Generating queries. Queries are generated in our algorithm based on the embedding from previous phases. In each phase, we generate queries that (a) are

Algorithm 1: CrowdSter(budget, batch_size)

```
 1: E = EmbedData()       // random embedding
 2: num_of_queries = 0
 3: while num_of_queries < budget do
 4:   Q = SelectQueries(E, batch_size)
 5:   R = RunQueries(Q)   // using the crowd
 6:   D = DistanceFromEmbedding(E)
 7:   D = UpdateDistances(D, R)
 8:   E = EmbedData(D)
 9:   num_of_queries += batch_size
10: end while
11: D = DistanceFromEmbedding(E)
12: Output D
```

local, and (b) cover the set of images as evenly as possible. To do so, we sample random images uniformly while making sure they are not nearest neighbors of each other. When no such samples remain we start over. For each image, we find its k nearest neighbors in the given embedding. Then, out of these neighbors we sample a random subset of size n_q and use it as the next query.

Embedding. We maintain an embedding of all images in the dataset, which is gradually improved with each batch of queries. The embedding infers a consistent distance between every pair of images, which is used in the next phase. Before the first queries are sent to the users, the images are embedded into a Euclidian space using a uniform random distribution. To improve the embedding according to the query results, we calculate the distance between each pair of images from the embedding, update the distances accordingly, and embed the images again using the updated distances. This consolidates the updated distance and resolves any inconsistencies among them. To compute the embedding we use multidimensional scaling (MDS), whose input is the distance between each pair of images.

More specifically, we want to find an embedding by taking into account only distances that we have information of (via query results), ignoring all other distances. For this we use Sammon Projection [13], which is a multidimensional scaling technique that computes an embedding using a stress function and gradient descent. The weighted stress function can take into account the relevant distances and ignore other distances by giving them a very small weight. All weights are initialized to a very small value ε. In each phase, we set the weight for each updated distance to 1. Distances that were updated in previous phases maintain a weight of value 1, so once a pair of images is queried its distance is always taken into account when computing the embedding in subsequent phases.

Updating the distance. To update the distance, all the query results in the batch are aggregated and analyzed. For each pair of images in each query, we refer to a query result as *positive* if the images were assigned to the same cluster, and *negative* if the images were assigned to different clusters. The distance between a pair of images is shortened if the pair has more positive than negative query results, and made longer if the pair has more negative query results. The distances between pairs of images for which there was a tie and pairs of images that did not appear in the same query are not affected.

Distances are shortened by dividing by β and are made longer by multiplying by β. In our experiments β is set to 4. Note that we do not take into account the number of times a pair of images appeared in the same query. For example, a pair of images that has two out of two positive query results is updated in the same manner as a pair of images that has three out of four positive query results. Since the phases tend to be short, the probability that the same pair of images will appear in many queries is small, and inferring from the exact ratio between positive and negative results is too sensitive to randomness.

4 Experiments

To evaluate the efficiency of our approach, we conduct three sets of experiments, described below. First, to verify the correctness of our approach, we conduct a set of small-scale experiments with real crowd but for a data set where the ground truth is known. This ground truth allows evaluation of the result quality. Second, we test the practicality of the approach for semantically-rich image similarities, again using real crowd and larger sets of images, where the ground truth is unknown. Finally, to further investigate each component of our solution, we conduct synthetic experiments where the ground truth similarity is known, and the crowd answers to queries are simulated accordingly. We vary different parameters of our system, and observe the effect on the output quality. In all sets of experiments, we further compare the results we obtained to alternative, baseline algorithms.

- **Random:** An algorithm which randomly selects the questions to the crowd, equivalent to executing our algorithm in a single phase.
- **Crowdcluster:** An algorithm is using the method of [6]. The results of this method are targeted to identify clusters, but also include a mean spatial location for every image, which we use as an alternative to our embedding.
- **Feature-based:** An algorithm that estimates the similarity of images based on automatically extracted image features, which serves as a baseline where ground truth is not available.

We now provide details about our implementation, and then describe our experimental results.

Implementation and Crowd UI Our crowdsourcing system includes a dedicated, user-friendly crowd interface. The UI of the system is implemented on the Google App Engine platform.[2] The back-end analysis of the crowd answers and the computation of the next queries to be posed to the crowd is performed in MATLAB R2014b. A screenshot of the UI is shown in Fig. 4. Initially, we display 20 images on the left-hand side of the screen (the query), and the crowd member is asked to drag and drop the images in one of the 4 right-hand side bins (and also move images between bins). Crowd members can also decide to leave images outside of any bin if they are unrelated to any of the other images, in which case our algorithm only infers that the leftover images are less similar

[2]Google App Engine. `https://cloud.google.com/appengine/docs`

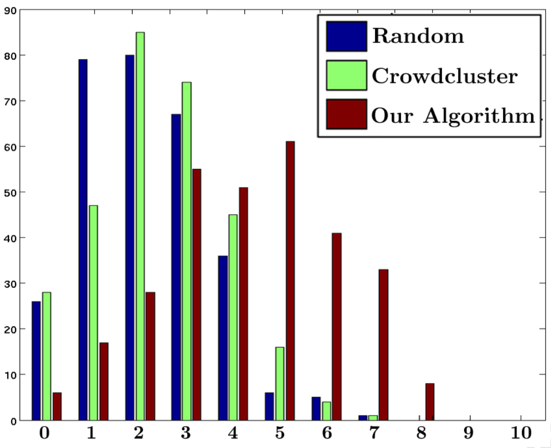


Figure 5: The number of correct 10-NN images based on real crowd input, comparing the results of our algorithm with the two baseline alternatives.

to the images within the bins. This UI was used in the experiments described below.

4.1 Crowd Experiments with Ground Truth

As a sanity check, we have executed our algorithm over two sets of images where the ground truth is known, and for two different computation tasks: top-k and clustering. The experiments were conducted with about 10 crowd members[3] crowd members that answered both the baseline algorithms' and our algorithm's queries.

Top-k similar colors. The simplest set of images that we have used is a set of 300 solid colors, whose ground truth similarity can be measured, e.g., by embedding the colors into 3-dimensional space according to their RGB or HSL values (we have used RGB). The goal was to compute, for each color, the k-NN most similar colors for varying values of k. We have compared the results of our algorithm to the results of the baseline random and crowdcluster algorithms, the same number of queries overall in the three algorithms.

The results of this experiment indicate that our algorithm identifies a larger percentage of the nearest neighbors for a larger percent of the images. Figure 5 illustrates the 10-NN results for the three algorithms using 235 queries overall, and 5 phases of our algorithm. For each algorithm, we show a histogram of intersection between the true 10-NN (according to the ground truth) and the computed 10-NN. Note that crowdcluster slightly outperforms the random baseline, but our algorithm generally identifies a larger fraction of the true 10-NN images, "pushing" the histogram rightwards (red bars). Overall, our algorithm identifies 43.4%-50% more of the true nearest neighbors than the baseline alternatives, which demonstrates the effectiveness of our progressive refinement approach.

Clustering fonts. In this experiment we have tested the ability of our algorithm

[3]We were only tracking IPs, so the number of different crowd members is estimated.

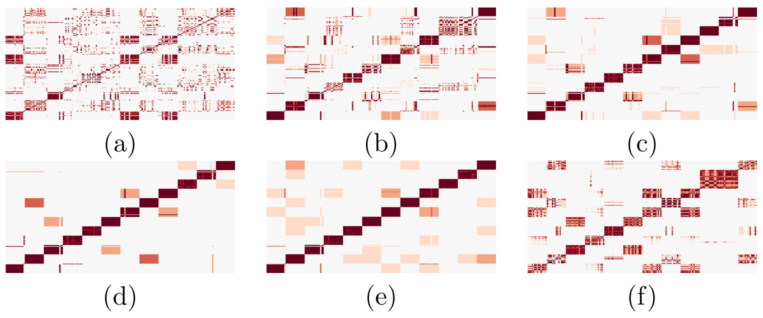


Figure 6: Heatmaps displaying the accuracy of clustering for the font dataset. Figures (a)-(e) illustrate the cluster quality after phases 1-5 of our algorithm, respectively, and 123 queries in total. For comparison, Figure (f) displays the cluster quality after 123 random queries.

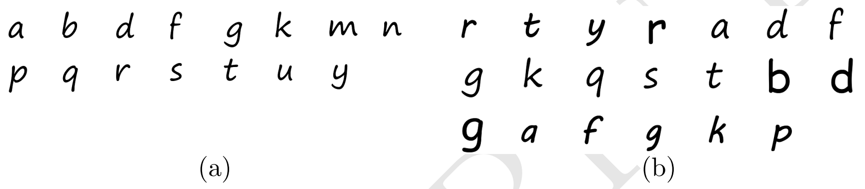


Figure 7: Two examples for clusters produced for the same letter "a" (on the top left), based on the similarity metric of (a) our algorithm, and (b) random baseline.

to cluster letter images into fonts, where the ground truth is the font to which the letters belong. We have used 180 letters of 12 different fonts, and asked crowd members to evaluate the similarity of letters with respect to their appearance. The results have been used to compute 12 letter clusters, which should ideally match exactly the 12 original fonts. Our algorithm has used 123 queries in total over 5 refinement phases. For comparison, we have executed the same task with 123 random queries.

Figure 6 illustrates the experimental results and in particular the progressive refinement, via heatmaps that represent the cluster quality after each of the 5 phases. The results of the algorithm are almost perfect, with only 1.1% errors (two letters). In comparison, the random query selection resulted in around 60% errors, and was outperformed by our algorithm already after the second phase. Figure 7 displays an example cluster produced by our algorithm, and the corresponding cluster produced by the random baseline. The latter cluster makes sense in the broader context of the fonts, since it contains only handwriting fonts; but the progressive refinement in our method allows distinguishing also between the different handwriting fonts.

4.2 Crowd Experiments with Real-world datasets

Next, we have executed experiments with two real-world datasets where the image similarity is highly semantic and therefore image features may not be

Dataset	Number of Images	**Success (%)**	Δ
Movie posters	910	87.2%	2.5
Chairs	1024	76.2%	3

Table 1: Real-world dataset results

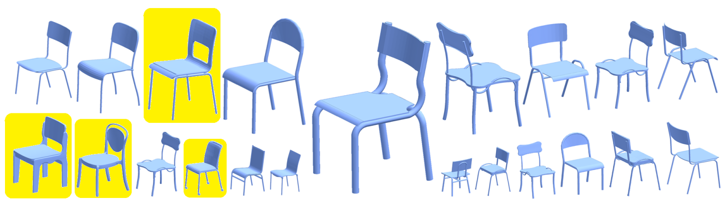

(a) (b)

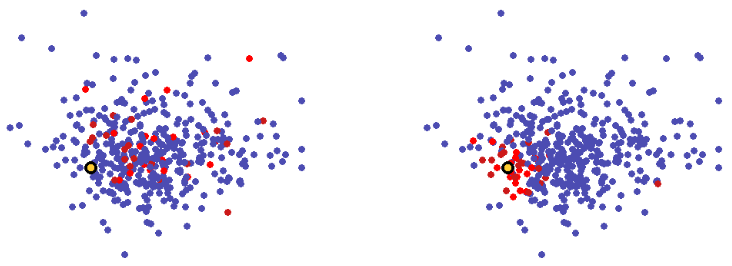
Figure 8: Nearest neighbors of the center image in a collection of chairs, computed using (a) HoG descriptor, and (b) crowdsourced queries. Smaller images mark farther neighbors. Less similar chairs are highlighted.

sufficient for estimating this similarity. The first dataset consists of 910 images of movie posters downloaded from the movie pages in Wikipedia, where similarity is usually based on genre, style of the poster, characters, and so on. For this set we have collected 547 query answers from about 60 crowd members.

The second dataset consists of 1024 chairs, of different types and angles from the ShapeNet dataset [15]. Similarity in this dataset is based, among others, on semantic features such as the usage of the chairs, the material they are likely to be made of, and their assessed level of comfort. For this set we have collected 559 query answers from about 60 crowd members.

As in many real-life scenarios, for these sets there does not exist a ground truth or a gold-standard. Hence, we have manually examined the results of our algorithm by sampling images with with their k-NN images, and comparing these results with the results obtained by automatic means based on image features. For the movie dataset we have used features based on the following color descriptors. First, a color histogram with 64 bins, four bins for each of the RGB channels. Second, an image thumbnail of four by four pixels, or a total of 16 RGB values, i.e. a vector of length 48. The two descriptors were concatenated and treated as a single vector for the distance calculation. For the chair dataset we have used features derived from HoG descriptor [3].

For the manual examination, we used 50 random “seed” images sampled from each of the datasets. For each seed image, we took its 10 NN images from the dataset according to both our algorithm and the feature-based baseline. Each of the images was labeled “very similar”, “similar”, or “unrelated” with respect to its seed image. We counted the percent of seed images for which our algorithm finds a greater number of similar images than the baseline, breaking ties by the number of “very similar” images. The results are displayed in the **Success %** column of Table 1. To quantify by how much we outperform the baseline, we also computed the average difference between the number of similar images our algorithm has discovered and the baseline. This difference is marked



(a) (b)

Figure 9: Visualization of the images that appeared in the same query as the image marked in gold. The images are ranked by the number of mutual queries and the top 10% images are colored red. (a) Mutual queries after 400 random queries. (b) Mutual queries after 400 queries using our algorithm.

by the Δ column in the table.

We illustrate a specific example of the observed difference in Fig. 1. The figure displays the 10-NN images (a) computed by our algorithm based on clustering queries and (b) according to color descriptors. The seed image is displayed in the middle. In this case, the results of our semantic similarity estimation retrieve movies of the same genre (animated adventure films). Within that genre, most of the closest neighbors (four out of the top five) have the same visual appearance (blue background) as the seed image. On the other hand, the movies retrieved using image descriptors have a similar visual appearance in terms of color scheme and mood but are very different semantically. Note that while we use rather simple image descriptors, even extremely sophisticated descriptors would fail to associate posters of movies in the genre which has a visual appearance different than the seed image.

Figure 8 displays similar results for the chair dataset, but where the baseline k-NN results (a) are computed according to HoG descriptor. The seed chair is a school chair with curvy tubes supporting the back. The 10-NN chairs given by our algorithm are all school chairs and many of them contain similar style elements such as curvy tubes. In contrast, the chairs computed using the HoG descriptor seem superficially similar (and also have the same orientation) yet include office and dinning room chairs, and vary more in their style (the less similar chairs are highlighted in the figure).

Figure 11 displays a few more selections of k-NN results for movie posters and chairs. In each set the top left image is the seed and its 7 nearest neighbors are presented from left to right. In many cases, the similarity between images can be both semantic *and* visual. We have deliberately selected cases which present a purely semantic relation which may be very hard or impossible to capture using image descriptors. The semantic connection between movie posters vary greatly, and spans movies from the same genre (a), posters that have dominant typographic elements (b), posters of old movies (c), or the same expression of the faces in the poster (d). The semantic connection between chairs may be similar style elements (e), similar overall shape (f), similar function (g) or even chairs with wheels (h). The k-NN results for all movie posters and chairs in the

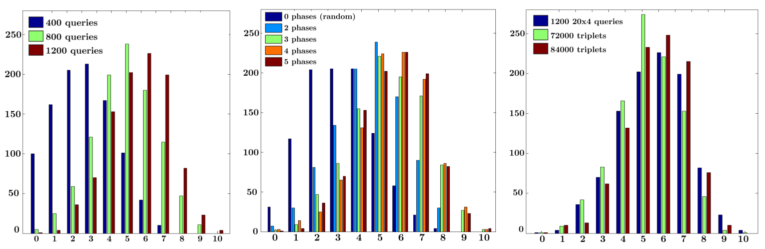


Figure 10: Number of correct 10-NN images as a function of number of queries (left) and number of phases (center), and versus a triplet-based algorithm (right).

dataset can be seen in the supplemental material.

The results of this experiment indicate that our algorithm is able to capture nontrivial semantic dependencies between images, relying only on the answers of the crowd. We note that in the examples we have shown, textual context such as a "wooden school chair" label could have also been leveraged, for enhancing the image similarity estimation or for creating an improved initial embedding; in this paper, however, we focus on algorithms that are purely based on crowd answers, and leave such optimizations for future work.

4.3 Synthetic Experiments

We next provide further analysis of our algorithm via synthetic experimental results. The experiments have been conducted on datasets with available ground truth, and with answers from a simulated crowd. The simulated answer of the crowd for a given query have been computed by a k-means algorithm, which has split the 20 images in the query into 4 clusters. Using synthetic answers allows us to test the performance of our algorithm in a variety of scenarios.

Effect of locality. In the Introduction, we have stressed the importance of using queries about local neighborhoods of images. To test this claim in isolation, we have conducted a dedicated synthetic experiment, as follows. We have used a set of 1000 colors sampled uniformly. Since the true similarities are known for this image set, we could vary the locality of queries: for each query we started from a seed image, then sampled the rest of the images from within a certain distance from the seed image. We have then used the results of the queries to compute the embedding as usual. We have observed an almost linear decrease in the average precision of the computed 10-NN images as the distance between images in each sample increases.

Co-occurrence of similar images. One of the indications for the effectiveness of the progressive refinement in our algorithm is the frequent co-occurrence of similar images in the same query. Ideally, as the similarity metric that we compute converges to the true one, similar images are more likely to appear in a query together. Moreover, the distance between pairs that appeared together in many queries is expected to be more accurate, since more data is available. Since the budget of queries is limited, each pair that is queried comes at a cost of another pair for which there will be less available information. We show that our algorithm effectively favors pairs which are close to each other and therefore need more accurate information.

Figure 9 illustrates this. We simulate a two dimensional embedding of images, where each point represents an image in the dataset. The distance between each pair of points (or images) is taken from the embedding, which simulates ground truth similarity. The dataset contains 400 images, and we ran 400 simulated queries, once using our algorithm and once with random queries. We then select an arbitrary image (marked in gold) and count how many times each image in the dataset has co-occurred with it. We rank the images according to their mutual queries count. The top 20 images (5% of the dataset) that were queried together the most with the golden image are colored bright red. The next 20 images (5%) are colored dark red. The rest of the images (360 or 90%) are colored light blue.

Figure 9(a) shows that using random query selection, the images that co-occurred the most with the golden image are randomly scattered, as expected. In contrast, using our algorithm to select the queries (Figure 9(b)), the frequently co-occurring images are centered around the golden image. Evidently, we do not spend queries on pairs which are known to be far away, since their distance from each other matters less and is expected to be less accurate. This allows our algorithm to better estimate the relative local similarities, and use them to estimate the global similarities.

Varying the algorithm parameters. We next execute our algorithm while varying the value of two parameters: the total number of queries and the number of phases, to demonstrate the impact of these parameters on the query results. Figure 10(right) illustrates the effect of varying the total number of queries, for a synthetic 1000 random color dataset, and 5 phases of our algorithm. As expected, there is a positive correlation between the number of queries we use and the quality of the results, measured by the size of the intersection between the true 10-NN images and the 10-NN images that we compute. This means that with a greater budget we can improve the estimation of the similarity metric.

Figure 10(center) illustrates the impact of number of phases on the quality of the results (using the same image set as above, the same quality metric, and 1200 queries overall). The number of phases ranges from 0 (which is equivalent to random query selection) to 5. Note that increasing the number of phases increases the result quality, since recomputing the embedding more frequently allows creating better queries. However, the margin by which the quality improves decreases, so the difference between 4 and 5 phases is small.

Queries versus triplets. As described in the Introduction, a common solution for collecting image comparisons from the crowd is based on triplet queries, i.e., queries of the form “Is image A more similar to image B or to image C?”. We have already noted that one advantage of our approach over the triplet-based one is that clustering queries provide context for comparison. In this synthetic experiment we ignore context, and focus on the number of questions needed for each type of solution. As shown in Figure 10(right), our algorithm’s performance using 1200 queries is comparable to the triplet-based algorithm’s performance using 84000 queries.

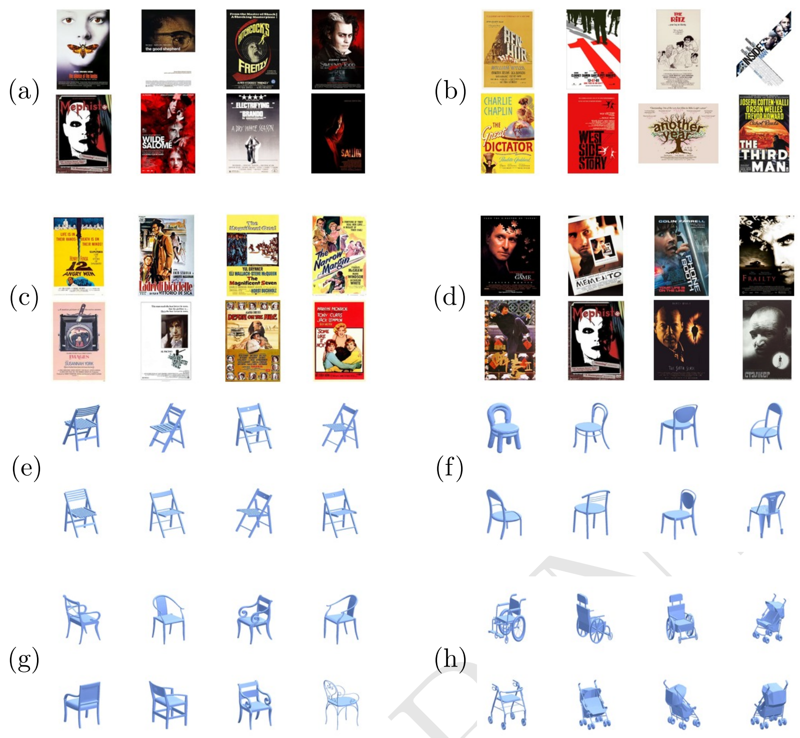


Figure 11: Example of K-NNs of images from the movie posters and chairs datasets.

5 Conclusion

In this paper, we have presented an efficient approach for estimating the similarity of images, based solely on the input of the crowd. Our system progressively refines the images posed to the crowd, in order to obtain similarity comparisons between images in the same neighborhood, allowing faster convergence to an accurate similarity metric. In our experimental study we have used a particularly small number of queries, and have shown that even on this basis we can obtain a fair estimate of the semantic similarity.

Limitations and future work. This work focuses on input from the crowd alone. However, it is often the case that some clues for the semantic similarity of images are available in the form of image features or textual context. Even if these clues do not account for the full range of semantic connections, it would be interesting to examine how to leverage them in conjunction with our algorithm. This direction may benefit the method's scalability, since in very large image sets, the affordable number of queries might not even be linear in the size of the set. A straightforward approach for integrating semantic clues would use our algorithm to learn similarities for a small fragment of the image set, and then apply machine learning techniques to complete the rest, using features based on semantic clues (in the spirit of Lun et al. [8], Saleh et al. [12], and Yi et al. [22]). A more interesting solution may further combine the clues within the query generation phases. This is non-trivial, since the usage of other estimates can potentially cause semantically similar images to be overlooked.

Another challenging direction for future work includes a more elaborate treatment of the uncertainty stemming from the crowd. Crowd members often disagree on the similarity of images, or even provide some inconsistent answers. So far, we have assumed that the embedding we perform mitigates the impact of such inconsistencies. However, we may want to explicitly account for inconsistencies, by a probabilistic modeling of the crowd's behavior, e.g., as done in [6] for the purpose of clustering. It would thus be interesting to develop probabilistic models dedicated for the learning of a similarity metric. In particular, this method should support efficient computations, due to the interactive nature of our algorithm.

References

[1] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *J. Machine Learning Research*, 6(6), 2005.

[2] A. Biswas and D. W. Jacobs. Active image clustering with pairwise constraints from humans. *Int. J. Comp. Vis.*, 108(1-2), 2014.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[4] S. B. Davidson, S. Khanna, T. Milo, and S. Roy. Using the crowd for top-k and group-by queries. In *ICDT*, 2013.

[5] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 2007.

[6] R. G. Gomes, P. Welinder, A. Krause, and P. Perona. Crowdclustering. In *NIPS*, 2011.

[7] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[8] Z. Lun, E. Kalogerakis, and A. Sheffer. Elements of style: Learning perceptual shape style similarity. *ACM Transactions on Graphics*, 34(4), 2015.

[9] A. Marcus, E. Wu, D. Karger, S. Madden, and R. Miller. Human-powered sorts and joins. *PVLDB*, 5(1), 2011.

[10] P. O'Donovan, J. Libeks, A. Agarwala, and A. Hertzmann. Exploratory font selection using crowdsourced attributes. *ACM Trans. Graph.*, 33(4), 2014.

[11] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[12] B. Saleh, M. Dontcheva, A. Hertzmann, and Z. Liu. Learning style similarity for searching infographics. *arXiv preprint arXiv:1505.01214*, 2015.

[13] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 18(5):401–409, 1969.

[14] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[15] H. Su, E. Yi, M. Savva, A. Chang, S. Song, F. Yu, Z. Li, J. Xiao, Q. Huang, S. Savarese, T. Funkhouser, P. Hanrahan, and L. Guibas. Shapenet: An ongoing effort to establish a richly-annotated, large-scale dataset of 3d shapes. `http://shapenet.org`, 2015.

[16] O. Tamuz, C. Liu, S. J. Belongie, O. Shamir, and A. Kalai. Adaptively learning the crowd kernel. In *ICML*, 2011.

[17] C. Wang, D. Blei, and F.-F. Li. Simultaneous image classification and annotation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1903–1910. IEEE, 2009.

[18] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11), 2012.

[19] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2005.

[20] M. J. Wilber, I. S. Kwak, and S. J. Belongie. Cost-effective HITs for relative similarity comparisons. In *HCOMP*, 2014.

[21] E. P. Xing, M. I. Jordan, S. Russell, and A. Y. Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, 2002.

[22] J. Yi, R. Jin, S. Jain, T. Yang, and A. K. Jain. Semi-crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning. In *NIPS*, 2012.

[23] Z.-J. Zha, X.-S. Hua, T. Mei, J. Wang, G.-J. Qi, and Z. Wang. Joint multi-label multi-instance learning for image classification. In *CVPR 2008*. IEEE, 2008.