Supplemental Material: Exploratory Font Selection Using Crowdsourced Attributes

1 Attribute Estimation

Number of Pairwise Comparisons. Selecting the number of pairwise comparisons n is an important choice when estimating font attributes. If n is too low, the estimated attributes are inaccurate. Exhaustive testing of all pairs (i.e., $n = 200$) is cost-prohibitive and unnecessary since the attribute values will converge to accurate values after far fewer comparisons. To determine a reasonable n, we ran a smaller study on 5 attributes varying n from 2 to 15. For each n, we evaluated the mean log-likelihood of a testing set of 5 additional comparisons per font. In Figure 1, we show a figure plotting the log-likelihood for n for each attribute. We found the log-likelihood plateaued after $n = 8$, so we used this value in our final study for the remaining attributes.

Modelling Error. In Table 1 we list the error in modelling the pairwise comparisons for each attribute. These results show which attributes are easiest to model ("thin", and "strong" for example), as well as the most difficult ("sharp" and "boring"). In Fig. 2 we show histograms of 4 different attributes. These histograms show that the attributes being learned have different distributions.

User Weights. In Fig. 3 we show a histogram of different user weights. Most users had positive weights, though a few users had negative weights, indicating they generally provided answers opposite to the majority opinion. We also plot the user reliability weights against the number of HITs completed by the user. This figure shows that the number of HITs completed is not correlated with the user's reliability.

2 Attribute Prediction Model Comparison

In Sections 4 and 5 of the main paper, we describe our approach for predicting attributes. In Section 4, we describe our approach from estimating attribute values from pairwise comparisons; in Section 5 we describe how we predict these attributes using geometric features with Gradient Boosted Regression (GBR) Trees [1] with a maximum depth of 2. We also experimented with a linear LASSO model [6]. On leave-one-out cross-validation tests, the GBR had better performance with a mean average error of 8.51 compared to LASSO's 10.76; we therefore use this model for all further tests.

An alternative approach is to use the pairwise comparison data to directly train the mapping from geometric features to attributes. The method of Parikh and Grauman [3] takes this approach by using a ranking SVM (SVMrank). We also modify our method in Section 4 of the main paper to take this approach by learning weights for a distance of geometric features rather than attributes. This model is a simple extension of the method we describe in Sec. 4.3 to estimate attributes. In that section, we model

Attribute	Mean Log-Likelihood	Classification Rate
thin	0.181	93.16%
strong	0.352	84.80%
wide	0.422	81.11%
soft	0.501	75.86%
disorderly	0.507	73.88%
artistic	0.509	73.80%
complex	0.520	73.09%
playful	0.537	71.75%
calm	0.548	70.80%
dramatic	0.549	71.08%
sloppy	0.559	70.20%
clumsy	0.563	70.14%
bad	0.570	68.53%
attention-grabbing	0.573	69.23%
formal	0.576	69.44%
attractive	0.579	68.08%
legible	0.580	68.28%
charming	0.586	67.67%
gentle	0.601	66.19%
modern	0.601	65.86%
pretentious	0.611	66.02%
happy	0.611	65.63%
graceful	0.619	65.05%
fresh	0.619	64.16%
friendly	0.622	64.39%
delicate	0.623	63.83%
warm	0.630	64.53%
technical	0.634	62.94%
angular	0.635	62.80%
boring	0.636	63.05%
sharp	0.649	62.02%
average	0.558	69.52%

Table 1: Estimation of relative attributes. We report the negative log-likelihood(lower is better), as well as the classification rate, the fraction of pairwise comparisons correctly predicted (higher is better)

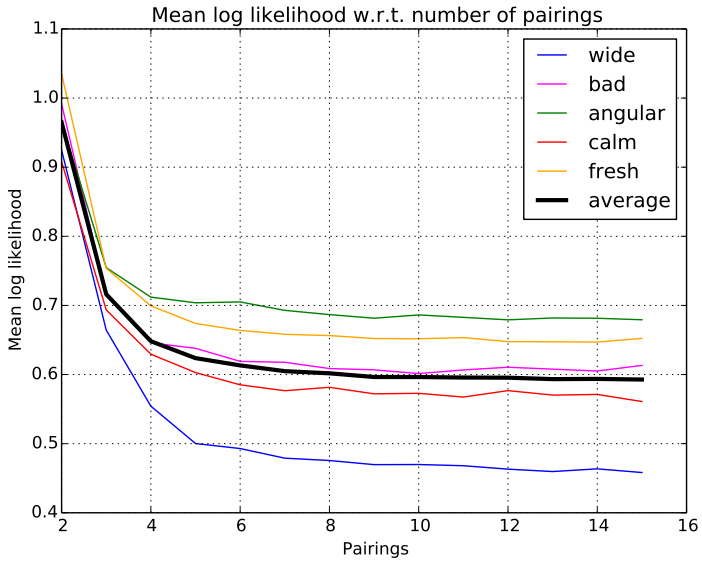


Figure 1: Mean log-likelihood values of the objective function with varying numbers of pairings per font. For the final study on all 31 attributes, we used 8 pairings per font.

the probability q, that a rater judges font f_i to have more of the attribute a than font f_j. We represent this probability using a sigmoid:

$$p(q = 1|f_i, f_j, a, u) = \frac{1}{1 + \exp(r_u(v_{j,a} - v_{i,a}))} \tag{1}$$

Where $v_{i,a}$ and $v_{j,a}$ be the unknown values of attribute a for the fonts i and j, and r_u is a per-user reliability values. These values are then estimated using gradient descent.

In the **Feature Weight** model, the values are computed by taking the product of fixed font features $\mathbf{x}$ with a learned per-attribute weight vector $\mathbf{w}_a$:

$$p(q = 1|f_i, f_j, a, u) = \frac{1}{1 + \exp(r_u(\mathbf{x}_j\mathbf{w}_a - \mathbf{x}_j\mathbf{w}_a))} \tag{2}$$

To evaluate the methods, we repeatedly train a model on the comparisons for 199 fonts and test on the hold-out font comparisons. We report the negative log-likelihood(NLL) of the test data and the classification rate (the fraction of comparisons correctly predicted). We also report an upper-bound "oracle" classifier which chooses the majority opinion. Table 2 shows that all methods perform similarly and do quite well, considering the high level of user disagreement. User modelling does improve results, as evidenced by a lower NLL, but only slightly.

When comparing the models, the likelihood is a more accurate continuous measure of performance as it uses the attribute distances. For example, the classification rate for two attribute values would be the same if the two attributes are very close or far apart, as long as the relative ranking is correct. Intuitively, such a discontinuous measure is not ideal since we expect that when the attribute distances are very small, users will have a harder time ranking them correctly.

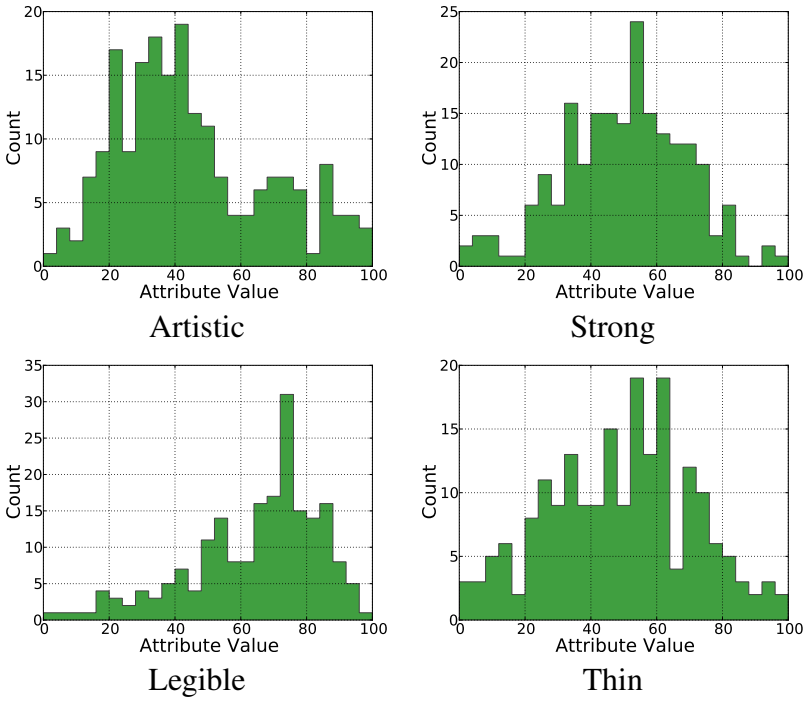
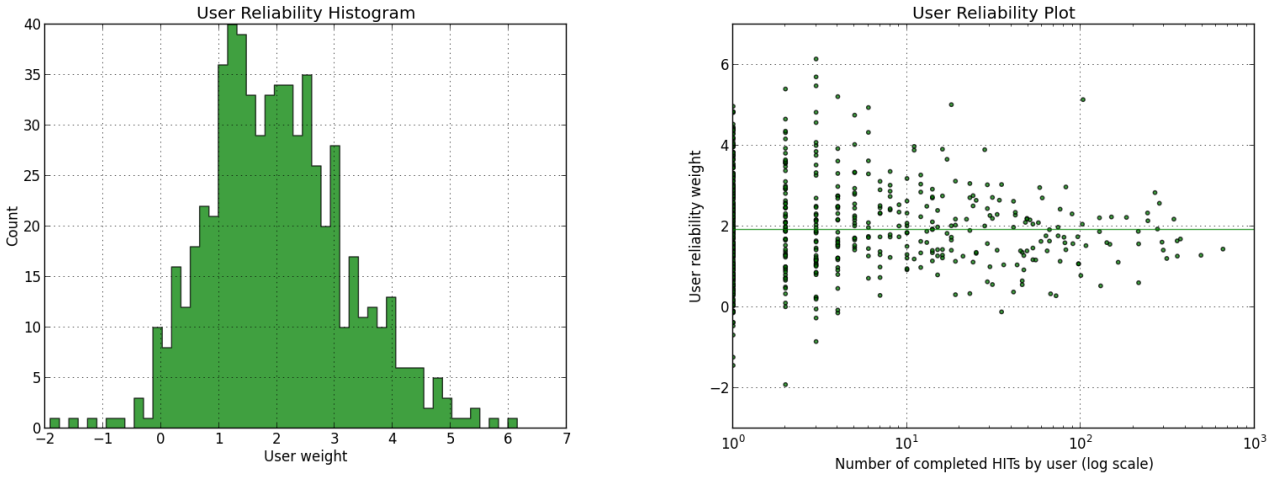


Figure 2: Histograms of selected attribute values estimated from crowdsourced data, demonstrating that attributes have different distributions.



(a) (b)

Figure 3: A histogram of user reliability weights assigned by fitting the objective function (a), and a plot of user reliability weights with respect to the number of responses (log scale) with the median weight indicated by a green horizontal line (b). Note that some user weights are negative, implying that the user is judged to have chosen the opposite answer of what attribute values would predict. There is no correlation between the number of HITs completed and the user reliability.

3 Metric Learning

Learning distance metrics is a well-studied problem in the machine learning community [2]. Distance metric learning aims to create distance function between objects; we refer to the distance between object i

Model	NLL	Classification
Feature Weights (w/o user)	0.6124	65.73%
Feature Weights	0.6087	65.69%
Direct Attributes + GBR (w/o user)	0.6128	65.63%
Direct Attributes + GBR	0.6086	65.62%
SVMRank		65.62%
Oracle (upper bound)		72.51%

Table 2: Comparison of attribute prediction models. Our Feature Weight method learns a set of linear weights on font features to match the pairwise comparisons. Our Direct Attribute method estimates the attributes values independently of features, and then trains a GBR using features for generalization. We show results without user reliability modelling, denoted as "w/o user". To evaluate the algorithms we report the negative log-likelihood(NLL) for testing data (lower is better) and the classification rate of the pairwise comparisons. NLL data is unavailable for the Oracle classification algorithm, which classifies a comparison as correct if it matches the majority opinion, as well as the SVMRank algorithm [3], which uses a non-probabilistic objective function.

and j as $d_{i,j}$. Typically, objects are embedded in some feature space $\mathbf{x}_i$; metric learning attempts to create a new embedding whose distances better matches a set of input distance constraints. For example, we might specify relative distances in the new embedding space using triplets (i.e, $d_{i,j} > d_{i,k}$).

Linear metric learning methods learn a matrix $\mathbf{W}$ such that $d_{i,j} = ||\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j||$. Schultz and Joachims [4] use a non-linear approach based on SVM learning to model distances. Learning these transformations can be posed as a constrained optimization problem which leads to efficient convex solutions. Unfortunately, most of this prior work is not probabilistic and therefore not designed to handle noisy crowdsourced data.

Recently, Tamuz et al. [5] defined a probabilistic sigmoid model for crowdsourced triplet comparisons: $p_{ijk} = \mathcal{S}(d_{i,j} - d_{i,k})$. Given the triplets, an embedding for each object is learned in a Euclidean space that requires no knowledge of object features. Our probabilistic model is very similar, but we learn a linear embedding matrix for features, thus allowing our approach to extend to unseen objects. We also model the reliability of individual users.

The data we use to train our models is a set of font triplets, expressed in the form $\mathcal{D} = \{(f_i, f_j, f_k, q, u)\}$, where f_j and f_k are the two fonts being compared to font f_i, q is the user's choice ($q = 1$ if the user judges font f_j is closer to font f_i than font f_k, $q = 0$ otherwise), and u is the user ID. To model the probability of q, we use a logistic function similar to the one in Section 4:

$$p(q = 1|f_i, f_j, f_k, u, \mathbf{W}) = \frac{1}{1 + \exp(r_u(d_{i,j} - d_{i,k}))} \tag{3}$$

where $d_{i,j}$ is the Euclidean distance between fonts i and j in some embedding space, parameterized by $\mathbf{W}$, and r_u is a per-user reliability weight.

The key question is how to model these distances $d_{i,j}$ and $d_{i,k}$. We first consider the simple **unweighted** Euclidean distance of a font feature vector:

$$d^E_{i,j} = ||\mathbf{x}_i - \mathbf{x}_j|| \tag{4}$$

where $\mathbf{x}_i$ is a feature vector for font f_i with size n. In this model, we use either the geometric features of Sec 4.4 of the main paper ($n = 80$), or the predicted attribute vector ($n = 37$). We evaluate both feature sets in the next subsection.

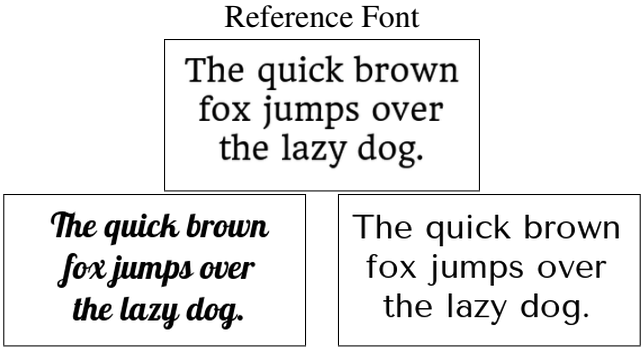


Figure 4: Example task for font distance study. The user is asked to decide which bottom font is closer to the reference font above.

The second model is a **weighted** Euclidean distance:

$$d_{i,j}^{\mathbf{w}} = ||\mathbf{w}^T(\mathbf{x}_i - \mathbf{x}_j)|| \tag{5}$$

Where $\mathbf{w}$ is a learned vector of weights.

The third **subspace** model computes the Euclidean distance after embedding the fonts in a lower n-dimensional subspace:

$$d_{i,j}^{\mathbf{W}} = ||\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)|| \tag{6}$$

Where $\mathbf{W}$ is a learned embedding matrix of dimensionality $m \times n$ where $n >> m$, where $n = 37$ or $n = 80$, depending on the feature set used, and $m = 7$, as selected by cross-validation.

Given the pairwise comparisons $\mathcal{D}$, the negative log-likelihood objective function is therefore:

$$\begin{aligned} E(\mathbf{M}, \mathbf{r}) \quad = \quad & -\sum_n q^n \log p(q = 1 | f_i^n, f_j^n, f_j^n, u^n, \mathbf{W}) \\ & -\sum_n (1 - q^n) \log \left(1 - p(q = 1 f_i^n, f_j^n, f_k^n, u^n, \mathbf{W})\right) \end{aligned} \tag{7}$$

We jointly solve for the embedding parameters $\mathbf{W}$ and rater reliabilities $\mathbf{r}$ using gradient descent.

3.1 MTurk Similarity Study

To obtain data to train the model, we conduct a crowdsourced study focused on font similarity. Workers are presented with a reference font A and two fonts (B and C) and are asked to decide whether B or C is more similar to A than the other. See an example of one such task in Figure 4. Triplets were randomly sampled from the previously described 200 font training set.

Each Mechanical Turk HIT contains 16 such tasks, as well as 4 control questions. For consistency, we repeat two tasks by swapping the fonts B and C. For correctness, we include tasks where one of B and C is the same font as A; the user is expected to recognize that a font is more similar to itself than any other font. We reject HITs that fail at least two of the four control questions. We also ignore users if 20% or more of their HITs were rejected, leading to a final rejection rate of 9.11%. Workers are paid \$0.07 per HIT. To obtain multiple opinions per triplet we created 130 HITs with at most 15 workers per HIT. After rejection, the average number of users per triplet was 13.6. The total dataset has 2340 triplets and 35,387 individual comparisons. Every font in the training set was in at least 21, and at most 57, triplets.

	Geometric Features		Attributes	
Model	NLL	Classification	NLL	Classification
Unweighted	0.5777	69.61	0.5441	71.11
Weighted	0.5092	74.66	0.4876	76.04
Subspace	0.4954	75.12	0.4833	75.83
Oracle		80.79		80.79

Table 3: Results of font distance metric learning using different distance models and features. We evaluate using geometric features extracted from the fonts directly (Sec. 4.4) as well as our predicted attribute values. We report the negative log-likelihood (NLL) of test font comparisons. We also report the classification rate: the fraction of test responses that are correctly predicted. The oracle is the upper-bound on the classification rate, given user disagreement. The improved performance of the attributes over geometric features demonstrate that attributes are a useful mid-level representation for describing fonts.

To evaluate the methods, we repeatedly train a model on the triplets for 199 fonts and test on the hold-out font triplets. Along with the negative log-likelihood of the test data, we report the classification rate. Given a triplet and user responses labelling one font as nearer than the other, the classification rate is the number of user responses which match our prediction. It is also worth noting that the data contains considerable noise. We therefore report the classification rate of an oracle algorithm which always correctly chooses the majority opinion for each triplet; this rate is an upper-bound.

We find that embedding fonts in a subspace model gives the best results (see Table 3), though the weighted model also performs quite well. Given the disagreement between users, the subspace model achieves a classification rate of 75.83% on testing data, out of a possible 80.79%. We also report results of the same learning procedure using the geometric features described in Section 4.4 rather than attributes; the performance of the different models is similar, though slightly better, with attributes. This result demonstrates that learning a mid-level representation of attributes for fonts is useful for tasks such as modeling similarity.

4 Evaluation Study Details

4.1 Font Matching Study

Each HIT contained 5 fonts to match, with 10 HITs total. The same sequence of target fonts was also used for each interface. Each interface is used by 15 workers, giving 750 font selections for each interface. Workers were paid $\$0.50$ per HIT. Bonuses were also promised to users for the nearest font selections, with the top 25% of users receiving a bonus ranging from $\$0.10$ to $\$0.50$.

4.2 Design Task Study

We first conducted a study using Mechanical Turk. For each interface, we asked 30 workers to choose fonts for 15 designs, giving 1350 designs total. Each HIT contained 5 designs and used a single interface. Workers were paid $\$0.50$, and bonuses were also promised to users for the best designs, as evaluated by other users. The top 25% of users received a bonus ranging from $\$0.10$ to $\$0.50$. 200 workers completed the HITs.

We next evaluated the designs created using the new interfaces against the designs created by the baseline interface. 2AFC testing was performed with designs selected from the baseline interface and either the attribute or group interface; each selected design was compared to 10 other designs. Each HIT contained 18 comparisons, with 2 duplicates added for consistency. 8 users performed each HIT and were paid $\$0.07$. 455 HITs were created, producing 62,766 individual comparisons.

5 Designer vs. MTurk Font Evaluation

The high level of disagreement when evaluating fonts makes it unclear whether using novices is appropriate. We therefore conducted a study comparing font evaluation between MTurk users and three professional designers, recruited online.

In the main paper, we use pairwise AB comparisons between fonts to evaluate, and estimate rankings from multiple MTurk workers. However, this approach is not appropriate for directly comparing novices and designers. To allow a simpler comparison, we created ranking HITs where users were shown 9 font selections and ranked them from best to worst. The professional designers were paid $20 an hour, and completed 104 ranking tasks. MTurk users were paid $\$0.07$ per HIT, which included a single ranking task. MTurk workers could complete as many ranking tasks are they desired.

Due to the subjective nature of font evaluation, there is no correct ranking. However, given the experience and training of professional designers, it is expected that their responses are more trustworthy than novices. To compare these two groups, we use Kendall tau rank correlation coefficient. A score of 0 indicates no correlation between rankings, and a score of 1 indicates an exact agreement between rankings. The final Kendall tau score is found by averaging over all comparisons over all ranking questions. To control for the higher number of MTurk workers, for each ranking task, we selected the response of three workers at random.

We first compute the intra-group Kendall tau scores to measure how consistent the group members are. We found the consistency of users within both groups were similarly low, with mean of the absolute values of the Kendall tau coefficients of 0.369 and 0.365 for MTurk and designers respectively. The Kendall tau coefficient between the two groups was 0.322. These results suggest that font evaluation is highly subjective, for both novices and professionals. Furthermore, the agreement between novices and professionals is only slightly lower than between professionals themselves, suggesting MTurk evaluations are reasonable.

6 In-Person Testing.

To further evaluate our interfaces, we also conducted an in-person study with 31 participants; 17 were second-year design students and the rest were recruited from a study participant mailing list. Each participant used all three interfaces in shuffled order, creating 5 designs with each interface, with a time constraint of 2 minutes per design. After the font selection, each participant rated each interface based on various factor (overall preference, ease-of-use, ease-of-learning), and commented on each interface. We also showed participants all 15 designs in random order and asked them to rate their satisfaction with the font selection.

Table 4 shows the mean ratings for each interface. Participants generally preferred the new interfaces to the baseline interface, though the difference between the group and baseline interface was not statistically significant (using the Mann-Whitney U Test). As expected, participants found the list interface easiest to

Interface	Design Satisfaction	Ease of Learning	Ease of Use	Overall Preference
List	$\mathbf{m:4, \mu: 3.79 \pm 0.15}$	$\mathbf{m:4, \mu: 3.87 \pm 0.40}$	$m:3, \mu: 3.35 \pm 0.41$	$m:3, \mu: 2.81 \pm 0.33$
Group	$m:4, \mu: 3.70 \pm 0.16$	$m:3, \mu: 3.42 \pm 0.31$	$m:4, \mu: 3.58 \pm 0.38$	$m:3, \mu: 3.07 \pm 0.33$
Attribute	$m:4, \mu: 3.76 \pm 0.18$	$m:4, \mu: 3.60 \pm 0.34$	$\mathbf{m:4, \mu: 4.03 \pm 0.32}$	$\mathbf{m:4, \mu: 3.81 \pm 0.36}$

Table 4: Interface Ratings, median (m) and mean (μ) with 95% confidence intervals. After using all three interfaces, users were asked to rate the ease of learning, use, and overall preference for each interface. Users were shown their design choices from all three interfaces, in random order, and asked to rate their satisfaction with the font selection. The was no significant difference in final design satisfaction between the interfaces, though users preferred the attribute interface and found it easier to ease than other interfaces.

learn, given it's similarity to existing font selection interfaces.

Comments on the attribute interface were mostly positive, with 23 participants enjoying the interface, including: *"This was a great interface. I like that I had the option to select or not select attributes based on how I was feeling about the design."*, *"I liked this interface the best. When choosing a font, I thought of a few words that would describe my objective for the poster. After selecting my attributes, the 'similar fonts' button allowed me to find a few more fonts that suited my needs."* However, the interface took longer to learn: *"This is tricky without knowing what the attributes are initially, but quite handy after multiple uses"* and *"was a little overwhelming at first."* One participant felt the attributes *"were too broad. 'Artistic' can capture a number of ideas."*

Comments on the cluster interface were more mixed, with 15 participants enjoying the interface: *"This was the easiest to use, as it knew exactly which one I am going for."* However, other participants found the interface complex: *"If I chose one font, there were a lot of other fonts to choose from so it was confusing,"* and found the groups hard to interpret. This interface does present the steepest learning curve of the three; it is possible that once users were familiar with the menu selection, these issues would be diminished. The font grouping could also be refined to remove redundant groups, or further organized by an expert.

Comments on the baseline interface were mostly negative, with 20 of the participants mentioning the difficulty of dealing with a large number of fonts: *"Definitely the most time consuming and irritating of the bunch since I had to scroll through a lengthy list just to find a specific font."* However, some participants did prefer the simplicity: *"This interface is okay if the list of possible fonts were not too large....the interface was very simple to learn."* It is worth noting that if the design task is an extremely simple one, such as choosing a font for an essay, then an exploratory interface is not appropriate. Users would be better served by a small list of high-quality fonts.

References

[1] Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 29, 2000.

[2] Brian Kulis. Metric Learning: A Survey. In *Foundations and Trends in Machine Learning*, 2011.

[3] Devi Parikh and Kristen Grauman. Relative Attributes. In *Proc. ICCV*, 2011.

[4] Matthew Schultz and Thorsten Joachims. Learning a Distance Metric from Relative Comparisons. In *Proc. NIPS*, 2004.

[5] Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Kalai. Adaptively Learning the Crowd Kernel. In *Proc. ICML*, 2011.

[6] R. Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Royal. Statist. Soc B*, 58, 1996.