To appear in ACM TOG 33(4).

Exploratory Font Selection Using Crowdsourced Attributes

Peter O'Donovan* University of Toronto
Jānis Lībeks* University of Toronto
Aseem Agarwala Adobe
Aaron Hertzmann Adobe / University of Toronto

Abstract

This paper presents interfaces for exploring large collections of fonts for design tasks. Existing interfaces typically list fonts in a long, alphabetically-sorted menu that can be challenging and frustrating to explore. We instead propose three interfaces for font selection. First, we organize fonts using high-level descriptive attributes, such as "dramatic" or "legible." Second, we organize fonts in a tree-based hierarchical menu based on perceptual similarity. Third, we display fonts that are most similar to a user's currently-selected font. These tools are complementary; a user may search for "graceful" fonts, select a reasonable one, and then refine the results from a list of fonts similar to the selection. To enable these tools, we use crowdsourcing to gather font attribute data, and then train models to predict attribute values for new fonts. We use attributes to help learn a font similarity metric using crowdsourced comparisons. We evaluate the interfaces against a conventional list interface and find that our interfaces are preferred to the baseline. Our interfaces also produce better results in two real-world tasks: finding the nearest match to a target font, and font selection for graphic designs.

1 Introduction

Typography is fundamental to graphic design. A well-chosen font can make a design more beautiful and more effective in communicating information. Font selection is also subtle: many professional designers take entire courses on typography, and, for novices, the process can be frustrating and opaque. Surprisingly, the standard interface for selecting fonts — a long list of font names — has not changed in decades, and often overwhelms users with too many choices and too little guidance. As a result, users often proceed with the default font, or stick with a few familiar, but poor, choices.

The problem of font selection is challenging for many reasons. First, the space of possible fonts is quite large. Most computers are now equipped with hundreds of fonts, and online repositories provide hundreds of thousands. Second, there is no obvious method for categorization that supports a user's goals. Modern font listings use categorizations like "serif" or "display," but these must be hand-annotated, and they don't necessarily correspond to a user's goals. Font names themselves are rarely meaningful. Third, there are a variety of font selection tasks with different goals and requirements. One designer may wish to match a font to the style of a particular image. Another may wish to find a free font which looks similar to a commercial font such as Helvetica. A third may simply be exploring a large set of fonts such as Adobe TypeKit or Google Web Fonts. Current methods for font selection fail to address any of these needs well. Exhaustively exploring the entire space of fonts using an alphabetical listing is unrealistic for most users.

This paper proposes interfaces for selecting fonts based on the idea that fonts can be described by *attributes*: adjectives that describe their visual personality or appearance, such as "formal," "friendly," or "legible." We use these attributes as the basis for three font selection interfaces that are designed to support different types of exploration and search. First, we describe an **Attribute Interface** that allows a user to select one or more descriptive attributes; the system then shows a list of fonts that are sorted by how highly they score along the selected axes, e.g., we can show fonts that are both friendly and legible. Second, we propose a **Group Interface** that shows the user a hierarchical menu of fonts, clustered according to their visual similarity. Third, both interfaces include a **Search-By-Similarity** feature which gives a list of fonts sorted by similarity to their currently-selected font, allowing users to fine-tune their choices. These interfaces allow users to search in a variety of ways, from high-level exploration using attributes or font groups, to refinement of font choices using the similarity search.

In this work, we propose an approach that estimates attribute values and visual font similarity by learning models from crowdsourced data. We first collect a set of attributes commonly used by designers to describe fonts, and then compute attribute values for a set of fonts using crowdsourced data. We also learn a function that predicts the visual similarity of fonts from crowdsourced data. Finally, we learn to predict attribute values and font similarity from geometric features of fonts, so that our system can handle new fonts without any further data collection. The dataset and interfaces are available at the project page.

We evaluate our approach by two real-world design tasks, tested with in-person experiments and online with Amazon Mechanical Turk. First, we test users' ability to find a specific font in an interface, given an image of text in that font. For this task, users are three times more likely to find the font by using either of our interfaces, compared to a conventional linear list. Second, we perform a more subjective test of selecting a good font for a given design. Our proposed interfaces show a statistically-significant improvement over the conventional list, though the effect size is small since font choices are highly subjective and multimodal. Participant surveys show that users frequently prefer our interfaces to the conventional interface.

2 Related Work

To our knowledge, no prior work directly studied font selection interfaces. There is work in a number of related areas.

Font Traits and Features. Researchers have studied the personality traits of certain typefaces and how they affect the appropriateness of typefaces for specific tasks. Shaikh [2007] provides an extensive list of typeface personality attributes that have been used in academic publications since 1920. Li and Suen [2010] and Mackiewicz and Moeller [2004] present small scale studies that explore the range of personality values for tens of fonts using a Likert scale. Shaikh et al. [2006] performed an online study with hundreds of participants where 20 fonts were rated for 15 adjective pairs (i.e., "sad/happy") and design type appropriateness. Users consistently attributed personality traits to fonts, and felt specific document types were more appropriate for certain fonts. For example, children's documents were seen as more appropriate for "funny" fonts. Lewis and Walker [1989] showed that when a word's meaning does not match its font personality traits, participants take longer to analyze the word. These results suggest that attributes are useful in font interfaces. We differ from prior work in that we build a working attribute-based interface. We also use data from a much larger study (31 attributes and 200 fonts) to train predictive models.

Optical Character Recognition systems use low-level raster features for text recognition [Srihari 1999; Solli and Lenz 2007]. We use

* Both authors contributed equally

Figure 1: ***Font selection interfaces.*** *The interfaces are shown next to the graphic being designed so that users can see font choices in context. (a) Attribute selection menu. The user may select one or more attributes and/or complements. When mousing over an attribute, examples of fonts with and without the attributes are shown on the right. (b) As the user selects attributes, a list of fonts with the given attributes are shown in the list; here, the user has selected* delicate *and* not thin*. (c) The user can fine-tune their selection by viewing fonts ordered by similarity to a query font. Fonts are shown with the term 'handgloves' rather than their name to make them easier to compare.*

vector features, which are not available in the OCR setting, and should be higher-quality indicators of style.

Another approach for representing fonts is HP's PANOSE standard [Laurentis 1993], which assigns a set of category numbers based on visual characteristics like weight or serif style. Proprietary mapping software can then be used to search by similarity. Unfortunately, there are no automatic methods of classifying fonts with PANOSE numbers, with Doyle [2005] reporting an adoption rate of less than 10% in his analysis of the system. We instead use an empirical approach to measure font similarity, and train models based on crowdsourced data.

Commercial Font Interfaces. While the majority of font selection interfaces in existing applications are simple linear lists, more sophisticated approaches have been developed. Several websites allow searching by font similarity, including Identifont, MyFonts' WhatTheFont, and Fontspring. The TypeDNA Photoshop plugin allows searching by similarity, as well as 4 attributes ("weight," "width," "italic," and "optical"). Unfortunately, the details of these searching algorithms are proprietary and may be partly based on hand annotation, so it is difficult to directly compare with our approach. Sites like Fonts.com and Dafont have a large number of attributes and categories for fonts, including subjective attributes. Unlike our automatic approach however, these binary labels are hand-annotated.

Attributes. Object attributes have recently become an active topic in computer vision. While binary attributes have been used for image search [Kumar et al. 2011; Tao et al. 2009], recent work for estimating relative attributes [Parikh and Grauman 2011] has been used in search interfaces. Our work is also inspired by this approach.

Chaudhuri et al. [2013] use relative attributes to help users find parts to assemble into 3D models. Though their focus is 3D modelling, they also show a proof-of-concept web-design interface where page elements such as fonts or background color are automatically swapped using an attribute slider. The system also uses font features, including size and Fourier coefficients to measure shape. However, their dataset only included 30 WordPress templates, and their attribute model and interface were not evaluated in any way. By contrast, our work focuses on font selection, uses a much larger dataset, and includes a rigorous evaluation.

WhittleSearch [Kovashka et al. 2012] is especially similar to our work as it allows searching image collections using relative attributes. However, we simplify the interaction considerably, as well as introduce cluster and similarity-based interfaces that support fluid switching between different types of search.

Exploratory Search. Interfaces that support exploratory search through large datasets are a common topic for the HCI and information retrieval communities [White et al. 2006]. One approach is to visualize high-dimensional data points plotted within a low-dimensional embedding such as t-SNE [van der Maaten and Hinton 2008]. We experimented with a t-SNE interface, but found it difficult to interpret the 2D layout; it is hard to know where to look in this space for a particular font, particularly with thousands of fonts. Another approach that we take is to create hierarchical categorizations of the data; for example, Huang et al. [2013] hierarchically cluster 3D shapes based on a hand-designed distance metric. A related problem is the exploration of continuous parametric spaces for design [Marks et al. 1997; Talton et al. 2009]; our approach is more appropriate for exploring a discrete design space. We also experimented with an adaptive grid interface, similar to Marks et al. [1997], where users could select several fonts and the system would adaptively display similar fonts. In practice this approach was far more time-consuming than our group interface; numerous selections were required to converge on reasonable fonts, and the approach did not provide a high-level view of the entire font space.

3 User Experience

We first describe our interfaces from the perspective of a user (see the Supplemental Video for a demonstration). We offer two ways to begin exploring fonts, as well as a Search-by-Similarity option to fine-tune an initial font selection.

Attribute Interface. The attribute interface is useful when a user has a conceptual rather than mental image of the desired font. For example, a user may want "happy" and "playful" fonts for a child's birthday card, "formal" fonts for legalese, or "legible" fonts for a wall sign. The interface uses a menu listing the set of attributes

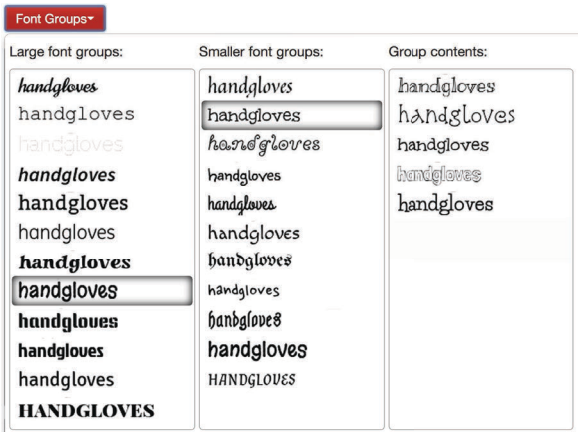


Figure 2: ***Group interface.*** *The interface shows a three-level perceptual clustering of fonts. Mousing over the clusters allows a user to quickly get a sense of the range of options, and to explore individual clusters. Once a user has selected a font, they may further refine their query by searching for similar fonts as in Figure 1.*

available (Figure 1a). The user selects an attribute by clicking it, or by clicking on the adjacent "not" button to select fonts that do not exhibit an attribute (e.g., picking "not strong" produces "weak" fonts). When the user mouses-over an attribute, five example fonts with that attribute are shown as examples, as well as five fonts lacking that attribute. Once the user selects an attribute, or a combination of attributes, fonts are shown sorted in order according to how well they match the selected constraints (Figure 1b). Multiple attributes may also be selected, in which case fonts are sorted by the average of how well they match the selected attributes.

Group Interface. The group interface supports a more visual exploration of the space of fonts for users who will recognize desirable fonts on sight. The group interface organizes the space of fonts into a tree-based hierarchy of visually similar fonts (Figure 2). That is, the leftmost list contains twelve groups of visually similar fonts, with a representative sample shown from each. When the user mouses-over a font, the middle list shows subgroups of that font's group. Each subgroup is shown by a representative font. Mousing-over any of these reveals their subgroups. The user clicks on a font in any list to select it. The groups are created automatically with a bottom-up hierarchical clustering algorithm.

Search-by-Similarity. Both of the above methods are useful for quickly identifying a reasonably appropriate font. Once a user has made an initial choice with either of the above methods, the Search-by-Similarity option (Figure 1c) helps the user refine their choice. This option lists fonts ordered by their visual similarity to the selected query font. Users can replace the query font with another by pressing a button next to any selected font. They may again show the most similar fonts to the new selection, thus exploring the space of fonts "near" their current favorites.

For all three interfaces the current design is shown to the right, and the user may choose between three text sizes ("small," "medium," and "large"); users can also save "favorite" fonts to a separate list. We display all fonts with a word "handgloves" that is often used by typographers for font comparison [Garfield 2011] since it contains many different letter strokes and shapes. Removing font names from the interface also reduces familiarity biases during evaluation.

To support these interface tools, we require models for relative attributes and font similarity. We next describe our approach for training these models from crowdsourced data.

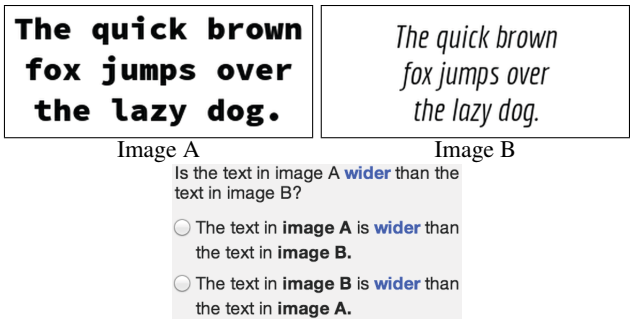


Figure 3: *Example of a study task for the attribute "wide."*

4 Estimating and Predicting Font Attributes

This section describes our technique for estimating and predicting relative font attributes. We gather pairwise comparison data by asking Mechanical Turk workers to compare a small set of training fonts according to different attributes. We then estimate relative scalar values for each training font and attribute, and use these values to train a model that maps from fonts to attributes. We then compute attributes values for a much larger font database using this model. We gather data from novices on MTurk instead of professionals because novices are the target users of our interfaces.

4.1 Font Selection

We gathered a large and diverse set of 1278 fonts that combines 1138 fonts from Google Web Fonts with a selection of web-fonts that appear frequently on a relatively small set of ≈3800 design-oriented web-pages, seeded from the Adobe Typekit Blog. Typefaces within the same font family often have very different personalities, and so we treat each separately, e.g., Gill Sans is treated as a separate font from Gill Sans Light and Gill Sans Bold. We then randomly sampled a training set of 200 fonts for the MTurk experiments.

4.2 Attribute Selection

We chose 31 attributes from a list of font personality attributes gathered by Shaikh [2007], reflecting adjectives that we expect novice users would use to describe fonts. We included concrete attributes such as "thin" and "angular," and more nebulous concepts like "friendly" and "sloppy." We also added 6 common typographical binary attributes: capitals, cursive, display, italic, monospace, serif. All 37 attributes are listed in Fig. 1. Relative attribute values range from 0 to 100, whereas binary attribute values may be either 0 or 100. We hand-label the 6 binary attributes in the training set, leaving the relative attributes to be estimated through crowdsourcing. We also performed an earlier version of this study with 36 relative attributes, and then pruned five after finding high correlations with other attributes. For example,"masculine" and "strong" were highly correlated, so "masculine" was removed.

4.3 Attribute Estimation

The goal of estimation is to determine a scalar value describing how much a given font embodies a given attribute. For example, a font that is often considered "stronger" than other fonts should have a higher value for the "strong" attribute. Directly asking people to provide these scores would be unreliable. Instead, we follow a standard approach and ask comparison questions. A Mechanical Turk worker is shown a pair of fonts, and asked to rate which is better described by the attribute (Figure 3). We use a two-alternative forced

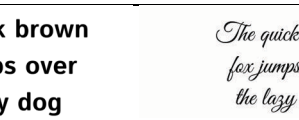
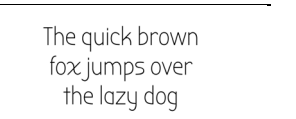
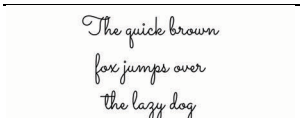
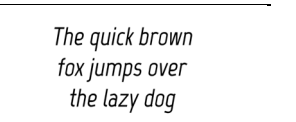
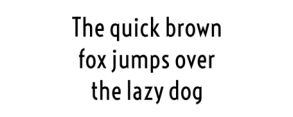
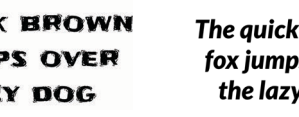
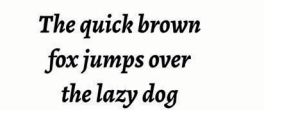
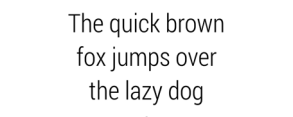
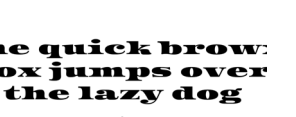
Dramatic	Legible	Delicate	Thin
			
100	100	100	100
			
66.44	66.70	66.65	66.32
			
33.62	33.73	33.34	33.37
			
0	0	0	0

Figure 4: ***Examples of estimated attribute values.*** *We show the fonts with the least of the attribute ($v = 0$), most ($v = 100$), and intermediate values ($v = 33, 66$).*

choice (2AFC) design, i.e., raters cannot answer "no difference," in order to allow us to better measure small differences.

Given these pairwise comparisons, we can then estimate the attribute value for each font. We use a Maximum Likelihood approach to attribute value estimation using the Bradley-Terry model [1952]. See the survey by Tsukida and Gupta [2011] for details. However, we augment it with a model of rater reliability, similar to item-response theory and the work of Welinder et al. [2010].

The measured pairwise responses is a set of tuples $\mathcal{D} = \{(a, f_i, f_j, u, q)\}$, where a is an attribute, f_i and f_j are the two fonts being compared, u is the rater's ID, and q is the rater's choice. $q = 1$ if the rater judges font f_i to have more of the attribute a than font f_j; $q = 0$ otherwise. In the standard approach, the likelihood of a rater's response q given the fonts and attribute is modeled as follows. Let $v_{i,a}$ and $v_{j,a}$ be the unknown values of attribute a for the two fonts. A rater is more likely to answer $q = 1$ if $v_{i,a} > v_{j,a}$, and $q = 0$ otherwise. However, the rater's response is more random (harder to predict) if the difference in attribute values is small. In the extreme case where $v_i = v_j$, the rater's response is entirely random ($p(q = 1) = 0.5$). To model the rater's response, we use a logistic function:

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

When performing MTurk evaluations, some raters may be more reliable than others. Hence, we introduce a per-user reliability weight r_u. Raters with low r_u produce more-random answers; raters with $r_u = 0$ are completely random, and raters with $r_u < 0$ tend to produce wrong answers. In the reliability model, the likelihood of a comparison is:

$$p(q = 1|f_i, f_j, a, u) = \frac{1}{1 + \exp(r_u(v_{j,a} - v_{i,a}))} \tag{2}$$

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

$$\begin{aligned} E(\mathbf{v}, \mathbf{r}) &= -\ln p(\mathcal{D}|\mathbf{v}, \mathbf{r}) && (3) \\ &= -\sum_k q^k \ln p(q = 1|f_i^k, f_j^k, a^k, u^k) && (4) \\ &\quad -\sum_k (1 - q^k) \ln \Big(1 - p(q = 1|f_i^k, f_j^k, a^k, u^k)\Big) \end{aligned}$$

where k indexes over all training tuples. We jointly minimize this objective with respect to all attribute values $\mathbf{v}$ and all rater reliabilities $\mathbf{r}$ by gradient descent. The resulting attribute values are scaled to lie in the range 0 to 100. We gathered the comparison data by a large-scale study on Mechanical Turk, described in Appendix A. Figure 4 shows examples of the estimated values.

We find that raters agree more on certain attributes than others. The attribute "thin" is easiest to fit, with 93.16% of the rater responses correctly predicted given our estimated attributes values, while "sharp" is the hardest with only 62.02% correctly classified. The average over all 31 attributes is only 69.64%, indicating substantial disagreement. Details for individual attributes are given in the Supplemental Material.

4.4 Attribute Prediction

We now describe an approach to learning a mapping from geometric font features to font attributes, using the estimated attributes as training data. The features, denoted $\mathbf{x}_i$ for font i, are computed from font files using the raw glyph outline control points and points sampled from the glyph outline curves. Features were selected in part to capture typographic font qualities (italics, thickness), as well as other vector-based qualities. We include features which measure the size, area, orientation, stroke width, and spacing of characters. We include vector-based features such as curvature, number of curves per glyph, arc lengths, etc. See Appendix B for more details.

We use these features to learn separate models for each of our 37 attributes. We learn the attribute values using Gradient Boosted Regression (GBR) Trees [Friedman 2000] with a maximum depth of

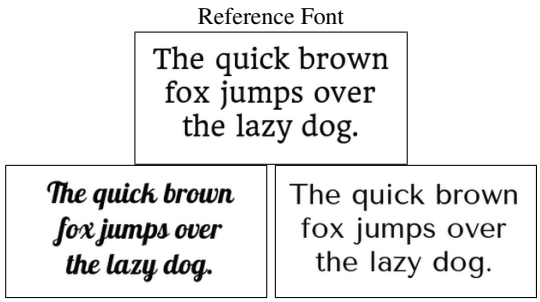


Figure 5: *Example task for font distance study. The rater decides which bottom font is more similar to the reference font above.*

2, and also a linear LASSO model [Tibshirani 1996]. 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.

4.5 Model Evaluation

To evaluate our approach, 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. 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. The models with and without user reliability have a similar classification rate of 65.63% and 65.62%, respectively. However, the model with user reliability has a NLL of 0.6087 versus 0.6124 without user reliability, indicating some value to the approach.

An alternative approach would be to use the pairwise comparison data to directly train the mapping from geometric features to attributes. The method of Parikh and Grauman [2011] takes this approach by using a ranking SVM (SVMrank). We can also adapt the approach in Section 4.3 to similarly learn weights for a distance of geometric features rather than attributes. Both of these methods perform equivalently to ours, and produce a classification error of 65.62% and 65.69%, respectively. See the Supplemental Material for more details.

5 Font Distance Metric

In this section we describe an approach to learning to predict the perceptual similarity between fonts, which is required for the Search-by-Similarity tool. A naive similarity metric would simply use the Euclidean distance between vectors of geometric font features, or between vectors of the 37 estimated attributes. However, it is unclear which attributes or features are more or less important when people evaluate the overall perceptual similarity. For example, it is possible that a difference in the "thin" attribute is more indicative of a visual difference than "serif."

We next describe the problem formulation and related work in metric learning, and then describe the results of our similarity study using MTurk. Our results show that font attributes outperform geometric font features for modeling font similarity, demonstrating their usefulness as a mid-level representation for fonts.

Model	NLL	Classification
Subspace(w/o user reliability)	0.4912	75.95
Subspace	0.4833	75.83
SVM		67.16
Oracle (upper bound)		80.79

Table 1: *We compare our metric learning with and without user reliability modeling, as well as the SVM approach of Schultz and Joachims [2004]. We report the negative log-likelihood (NLL) of the test data, as well as the classification rate: the fraction of responses that are correctly predicted. The oracle is the upper-bound on the classification rate, given user disagreement. NLL data is unavailable for the Oracle and SVM algorithms, as they do not output probabilities.*

5.1 Metric Learning

Learning distance metrics is a well-studied problem in machine learning [Kulis 2011]. The goal is to learn to compute the distance $d_{i,j}$ between any two objects i and j. We learn a linear metric parameterized by an *embedding matrix* $\mathbf{W}$:

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

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are feature vectors for fonts i and j, respectively. $\mathbf{W}$ has dimensionality $m \times n$, where the size of the feature vector is $n = 37$, and $m = 7$, as selected by cross-validation.

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. An example task is shown in Figure 5. Triplets were randomly sampled from the 200 font training set. See the Supplemental Material for study details including number of triplets, payment, and control questions.

We use a logistic model of a rater's triplet comparisons [Tamuz et al. 2011], augmented with a model of rater reliability [Welinder et al. 2010]. In particular, we train our model from a set of font triplets: $\mathcal{D} = \{(f_i, f_j, f_k, q, u)\}$, where f_j and f_k are the two fonts being compared to font f_i, u is the rater ID, and q is the rater's choice. $q = 1$ if the rater judges font f_j is closer to font f_i than font f_k, $q = 0$ otherwise. To model the probability of q, we use a logistic function similar to the one in Section 4.3:

$$p(q = 1|f_i, f_j, f_k, u, \mathbf{W}) = \frac{1}{1 + \exp(r_u(d_{i,j} - d_{i,k}))} \tag{6}$$

where r_u is a per-user reliability weight. Given the triplets $\mathcal{D}$, learning entails maximum likelihood estimation of $\mathbf{W}$ and the rater reliabilities r by gradient descent minimization of the negative log-likelihood, as in Sec. 4.3.

We evaluate the method using leave-one-out cross-validation, in which one font is omitted from training, and then classification is tested on triplets that include the hold-out font; results are averaged over each choice of hold-out font. In Table 1 we compare our method with and without user reliability modelling, as well as the SVM approach of Schultz and Joachims [2004]. For the SVM model, we found the RBF kernel produced the best results, with the parameters set using cross-validation. The upper bound on the performance is given by an oracle algorithm which always chooses the majority opinion for each triplet. Our probabilistic model outperforms the SVM approach, likely due to the considerable disagreement between users in our data. We also find that user reliability helps slightly, as evidenced by a lower NLL.

Figure 6: *Baseline selection interface with a list of fonts. Background photo courtesy of Nono Fara.*

In the Supplemental Material, we include further tests on the distance metric. We find that learning an embedding distance $\mathbf{W}$ performs better than using a vector of weights or an unweighted Euclidean distance. We also find that using attributes (Sec. 4.4) as features $\mathbf{x}$ performs slightly better than using just geometric features.

5.2 Grouping

For the Group Interface, we compute a hierarchical font categorization automatically with k-means clustering [Lloyd 1982] on fonts in the embedding space. Fonts are clustered in three levels in a bottom-up manner; the 1278 fonts are first clustered into 130 clusters, and then the centroids of each cluster are further clustered into 12 groups. The cluster sizes were chosen based on the constraints of a three-tier menu, and a target menu size of 10-12 items. The font nearest the center of each top-level group is shown in the first column of the interface (Figure 1c).

6 User Interface Evaluation

To compare the three interfaces, we conduct user studies on two separate design tasks. In the font-matching task, the user is presented with the image of a font, and attempts to find the font within a user interface. In the design task, the user attempts to select a font that is best suited to a given design. We compare against a baseline list-selection interface, similar to existing applications (Figure 6), in which fonts are shown in a random order.

In each trial, the user first reads the instructions for one of the three interfaces, then performs a brief tutorial task to familiarize them with the interface. The user then completes five font-matching or design tasks. The interfaces are reset between tasks. We impose a two-minute time limit on each task, in order to prevent highly-motivated users from exhaustively searching the lists of fonts.

6.1 Font Matching Task

We first test users' ability to find a given font within each user interface (Figure 7). Font recognition sites and apps such as MyFonts' WhatTheFont help users identify fonts used in existing designs, such as signs and advertisements; these sites are heavily used and demonstrate the usefulness of the task. The user might not be able to find the exact font within the two-minute time limit; in this case, the user should seek the most similar font. To simplify the task, the favorites box allows the user to keep a running list of the most relevant fonts for later review. For each interface, 10 task were created with 5 target fonts each. The target fonts are selected randomly, but, workers receive the same sequence of target fonts, regardless of the interface. Additional study details are in the Supplemental Material.

Interface	Distance to Target Font	Effect Size	Exact Match
Baseline	55.08 ± 2.13	-	5%
Attribute	52.97 ± 2.57	0.07	15%
Group	47.36 ± 2.20	0.26	15%

Table 2: *Results of the font matching study. Using the group interface, workers on average selected a closer font than using the baseline and found the exact font more often. For the attribute interface, workers also found the exact target more often, though there was no statistically significant difference for the mean distance. The distance between fonts is computed as the Euclidean distance in the learned embedding space of Sec. 5, with 95% confidence intervals.*

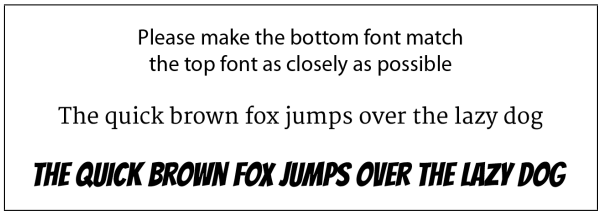


Figure 7: *Example of a font matching task. The users use the interface to make the bottom font match the top as closely as possible.*

Results from the MTurk study are shown in Table 2. We find that, using either of the new interfaces, a user is *three times* more likely to correctly find the target font, as compared to the baseline interface. We also compare the mean distance of the selected font to the target font for the new interfaces compared to the baseline, according to our learned metric (Sec. 5). The group interface performs significantly better than the baseline interface (using a two-sided t-test between the distances, $p < 0.05$). The effect size was 0.26, computed as $\frac{|\mu_n - \mu_b|}{\sigma_b}$, where μ_n and μ_b are the means of the new and baseline interface, and σ_b is the baseline standard deviation. There was no statistically-significant difference between the mean distances of the baseline and attribute interfaces (effect size 0.07). When users found inexact matches, they found better matches using the group interface rather than the attribute interface, perhaps because users had trouble identifying appropriate attributes for certain fonts.

6.2 Design Task

Our second evaluation uses a more subjective task: given a design, pick a suitable font for the main text, such as title or heading. Each design consists of a background image and modifiable text fields, with the background image usually containing text which the user cannot modify. We use 15 relatively simple designs created by three professional designers that reflect a variety of formal and informal event posters, invitations, announcements, and advertisements, similar to designs that an average user might create on their own (Figure 8). As with the matching task, workers have two minutes to complete the font selection task on each design.

We conducted the design task using Mechanical Turk, and with in-person studies with local university students. Mechanical Turk workers allow us to gather large amounts of data, whereas in-person studies allow us to perform more in-depth qualitative comparisons.

MTurk Study. For each interface we asked MTurk workers to choose fonts for 15 designs. Each study task contained 5 designs and used a single interface, with 1350 designs collected in total. We then 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, and evaluated by 8 workers. See the Supplemental Material for study details.

Figure 8: *Examples of the estimated design scores from crowd-sourced pairwise comparisons. The top three rows show MTurk designs ranging from 0 (worst) to 100 (best). The bottom row are the original designer font choices, along with the estimated relative score.*

Results are shown in Table 3. Our interfaces give a statistically significant improvement over the baseline (two-sided t-test with $p < 0.05$). However, the difference in preference effect size is relatively small, perhaps reflecting the highly variable preferences and abilities of MTurk workers at font selection and evaluation. We found considerable disagreement between MTurk raters: average votes had about a 74% majority, where 100% would indicate unanimous vote and 50% would indicate an exact tie.

We also compare the MTurk font selections to those originally chosen by the professionals that created each design (Table 3), using the same 2AFC study design as above. Each designer's font selection was compared to 12 baseline interface font selections, with random comparisons added to prevent a familiarity bias. This comparison provides a rough upper bound on the performance of new interfaces, as the designers selected the fonts when creating the original design, so the font should be a good choice for the design. Remarkably, the designer font choices were preferred over MTurk selections only 53.19% of the time, suggesting there is substantial noise and subjective preference in the evaluation.

We can also estimate a relative score using pairwise comparisons in the same manner as in Section 4.3. Figure 8 shows an example of the best, worst, and middle font selections for 4 of the 15 designs. We also include the scores of the original designer font choices.

Designer vs. MTurk Evaluation. The high level of disagreement between workers when evaluating fonts makes it unclear whether novices are capable of evaluating font selections. We therefore also collected pairwise comparisons from three professional designers, and compared the consistency of font selections among MTurk

Interface	Raw	Majority
Group vs. Baseline	$51.39\% \pm 0.86$	$51.93\% \pm 1.63$
Attribute vs. Baseline	$51.83\% \pm 0.84$	$52.60\% \pm 1.65$
Designer vs. Baseline	$53.19\% \pm 3.47$	$56.62\% \pm 8.33$

Table 3: ***MTurk interface evaluation.*** *Font selections from the new interfaces are compared against the baseline in forced AB comparisons. We report the raw percentage of people who preferred the design created by the new interface, as well as the percentage of comparisons where one interface had a clear majority (i.e., had 2 or more votes more than the alternative). The new interfaces perform statistically better than the baseline interface, though the effect size is small due to the high variance in user ability and font evaluation. We also compare the baseline interface fonts against the designer's original fonts, which provides a rough upper bound on the performance. Note the significant noise in the raw results, though designer's fonts are generally preferred. 95% confidence intervals are shown. The higher confidence intervals for the designer versus the new interfaces is due to a smaller sample size (180 vs. 3600).*

workers to the consistency among professionals. Surprisingly, the consistency among the two groups was similarly low; the mean of the absolute values of the Kendall tau correlations among MTurk workers was 0.369 versus 0.365 for professionals. The consistency between groups was 0.322. Kendall tau values ranges from -1 to 1, where -1 or 1 indicate perfect correlation between the ranks, and 0 indicates no correlation. These results suggest that font evaluation is highly subjective for both novices and professionals. The agreement between novices and professionals is also only slightly lower than between professionals themselves, suggesting MTurk evaluations are reasonable. See the Supplemental Material for details.

Qualitative MTurk Evaluation. Our tasks also included fields for providing general comments and suggestions on the task and interface. We received many positive comments, including *"I thought the interface was easy to use. It was helpful in picking out fonts based on attributes that you thought the design should have (i.e. "happy" for birthday card)"*, *"This would be an EXCELLENT tool for setting typefaces. Amazingly easy to use/edit. Fun!"*, *"The attribute selector was very helpful!"*, *"Its a great concept...People like me who are not pros would love this. Even professionals would love this."*, *"This actually seems like a really neat tool. As someone who occasionally needs choose a font I'm often overwhelmed with the flat list of choices. I look forward to finding this out in the wild."*

Some users also provided suggestions or negative comments. Most comments were related to task constraints (e.g., requesting more time for the task), or requesting more features (e.g., selecting bold or italic fonts, increasing the sizes of certain interface elements). A few users wished for the font groups to be labelled.

In-Person Testing. We also conducted in-person studies where 31 participants compared the three interfaces. 17 of the participants were second-year undergraduate design students and the others were students with little design experience. Participants created 5 designs with each interface, and rated each interface based on various factors (overall preference, ease-of-use, ease-of-learning) using a 5-point Likert scale, and then commented on the interfaces.

Participants significantly preferred the attribute interface to the other interfaces, using the Mann-Whitney U test with $p < 0.05$. However, there was no statistical difference between the group and the baseline interfaces. The median of the preference ratings were 4, 3, and 3 for attribute, group, and baseline interfaces. The mean preference ratings and 95% confidence intervals were 3.81 ± 0.36 and 3.07 ± 0.33 for the attribute and group interfaces, compared to 2.81 ± 0.33 for the baseline. The effect sizes on the mean ratings were 1.07 and 0.28 for the attribute and group interfaces, computed as in Sec. 6.1. In the comments, participants often positively described the attribute interface. The group interface received more mixed reviews; some users preferred it to the other interfaces, while others found the large number of groups confusing, as well as similarity between some groups. A supervised categorization with a smaller number of groups may help this issue. People found the baseline interface the easiest to learn, given its similarity to existing interfaces. However, many users commented on the difficulty of dealing with large numbers of fonts in that interface. People also found the Search-by-Similarity feature to be quite useful, giving it a score of 3.90 ± 0.27, with a median of 4. See the Supplemental Material for study details and more analysis of user feedback.

7 Future Work

We have proposed interfaces for font selection based on estimating attributes of fonts through a combination of crowdsourcing and machine learning. While we focus on fonts, this approach can extend to any domain where users must search large datasets. Potential domains include vector illustration (e.g., a sketchy drawing of a car), music (e.g., playful electronic music), and videos (e.g., a cute video of a dog).

Font attributes and font similarity are inherently subjective properties; however, there is enough agreement among human opinions to build reasonably effective models of these properties. Nonetheless, one limitation of our approach is that it learns the averaged response over all users, and ignores variation among individuals. Modelling an individual's perception of attributes or similarity is an open question; collaborative filtering techniques are one possibility for modelling individuals.

We have only scratched the surface of what can be done to improve font selection and make the difficult task of graphic design easier. Most immediately, it may be possible to combine the group and attribute interfaces into a single, more intuitive experience. We could also learn an "auto" button to automatically suggest a font for a design using a model trained from professional designs. Selecting fonts that are visually compatible (e.g., choose a header font to go with a specific body font) is also challenging; we could learn a joint model of compatible fonts from a corpus of designs. Finally, our set of attributes is limited; we could use natural language techniques to map any text to our existing set of attributes.

Acknowledgements

Thanks to John Hancock for his technical assistance. This work was supported in part by Adobe, NSERC, and CIFAR.

References

BRADLEY, R. A., AND TERRY, M. E. 1952. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika 39*.

CHAUDHURI, S., KALOGERAKIS, E., GIGUERE, S., AND FUNKHOUSER, T. 2013. AttribIt: Content Creation with Semantic Attributes. In *Proc. UIST*.

DOYLE, J. R. 2005. Evaluating the IBM and HP/PANOSE Font Classification Systems. *Online Information Review 29*, 5.

FRIEDMAN, J. H. 2000. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics 29*.

GARFIELD, S. 2011. *Just My Type*. Profile Books.

HUANG, S.-S., SHAMIR, A., SHEN, C.-H., ZHANG, H., SHEFFER, A., HU, S.-M., AND COHEN-OR, D. 2013. Qualita-

tive Organization of Collections of Shapes Via Quartet Analysis. *ACM Transactions on Graphics 32*, 4.

KOVASHKA, A., PARIKH, D., AND GRAUMAN, K. 2012. Whittle-Search: Image Search with Relative Atribute Feedback. In *Proc. CVPR*.

KULIS, B. 2011. Metric Learning: A Survey. In *Foundations and Trends in Machine Learning*.

KUMAR, N., BERG, A. C., BELHUMEUR, P. N., AND NAYAR, S. K. 2011. Describable Visual Attributes for Face Verification and Image search. In *IEEE PAMI*.

LAURENTIS, M. S. D. 1993. Panose 2.0 White Paper. Tech. rep., Hewlett-Packard Document.

LEWIS, C., AND WALKER, P. 1989. Typographic Influences on Reading. *British Journal of Psychology 80*.

LI, Y., AND SUEN, C. Y. 2010. Typeface Personality Traits and Their Design Characteristics. In *Proc. DAS*.

LLOYD, S. P. 1982. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory 28*, 129–137.

MACKIEWICZ, J., AND MOELLER, R. 2004. Why People Perceive Typefaces to Have Different Personalities. In *Proc. IPCC*.

MARKS, J., ANDALMAN, B., BEARDSLEY, P. A., FREEMAN, W., GIBSON, S., HODGINS, J., KANG, T., MIRTICH, B., PFISTER, H., RUML, W., RYALL, K., SEIMS, J., AND SHIEBER, S. 1997. Design Galleries. In *Proc. SIGGRAPH*.

PARIKH, D., AND GRAUMAN, K. 2011. Relative Attributes. In *Proc. ICCV*.

SCHULTZ, M., AND JOACHIMS, T. 2004. Learning a Distance Metric from Relative Comparisons. In *Proc. NIPS*.

SHAIKH, A., CHAPPARO, B. S., AND FOX, D. 2006. Perception of Fonts: Perceived Personality Traits and Uses. *Usability News*.

SHAIKH, A. 2007. *Psychology of Onscreen Type*. PhD thesis, Wichita State University.

SOLLI, M., AND LENZ, R. 2007. FyFont: Find-Your-Font in Large Font Databases. In *Proc. SCIA*, 432–441.

SRIHARI, S. 1999. Multifont Classification Using Typographical Attributes. In *Proc. ICDAR*.

TALTON, J. O., GIBSON, D., YANG, L., HANRAHAN, P., AND KOLTUN, V. 2009. Exploratory Modeling with Collaborative Design Spaces. *ACM Transactions on Graphics 28*, 5.

TAMUZ, O., LIU, C., BELONGIE, S., SHAMIR, O., AND KALAI, A. 2011. Adaptively Learning the Crowd Kernel. In *Proc. ICML*.

TAO, L., YUAN, L., AND SUN, J. 2009. Skyfinder: Attribute-based Sky Image Search. *ACM Transactions on Graphics 28*, 3.

TIBSHIRANI, R. 1996. Regression Shrinkage and Selection Via the Lasso. *Royal. Statist. Soc B 58*.

TSUKIDA, K., AND GUPTA, M. R. 2011. How to Analyze Paired Comparison Data. Tech. rep., University of Washington.

VAN DER MAATEN, L., AND HINTON, G. 2008. Visualizing Data Using t-SNE. *Journal of Machine Learning Research 9*.

WELINDER, P., BRANSON, S., BELONGIE, S., AND PERONA, P. 2010. The Multidimensional Wisdom of Crowds. In *Proc. NIPS*.

WHITE, R., KULES, B., DRUCKER, S., AND SCHRAEFEL, M. 2006. Supporting Exploratory Search. *Commun. ACM 49*, 4.

Appendix A: Attribute MTurk Study Details

Each Human Intelligence Task (HIT) on MTurk consists of 16 comparison tasks, along with four control tasks for quality control. Two of the four control tasks check for consistency; we repeat two of the 16 tasks with the order of the fonts swapped. The other two control for correctness; we add two font pairs with the attribute "thin" which are unambiguous and should have a clear answer. We discard any HITs in which the worker fails two or more control questions; the final rejection rate was 8.1%. Workers were paid \$0.07 per HIT.

For each attribute, the total number of comparison tasks is $\frac{mn}{2}$, where m is the number of fonts, and n is the number of pairwise comparisons per font. The division by 2 appears since two fonts appear in each comparison. For our dataset, $m = 200$ and $n = 8$, providing 800 comparison tasks for each attribute, with font pairs selected randomly. Each comparison task was completed by 8 unique workers, providing $8n = 64$ individual responses for each font/attribute pair. Over all attributes and fonts, this produces a final dataset of 198,400 individual responses for 639 unique workers. In the Supplementary Material, we justify our choice of $n = 8$ with a study which varied this parameter.

Appendix B: Font Features

Size and Area. For both the lower and upper case 'X', we measure the width, height, and width/height ratio. We find each character's area, and the ratio of the area with the bounding box. We then find the min/max/mean over all characters. We also compute the height of the biggest ascender or descender in the character set: the length that extends above/below the mean line of the font.

Spacing. We measure the horizontal and vertical spacing between upper and lower case 'X' . That is, for the horizontal spacing, we measure the space between the two characters 'xx' and 'XX'. We also compute vertical spacing for characters with descenders.

Outlines. For each character, we compute the sum of the outline arc lengths, as well as the minimum and maximum of the set. We then compute the min/max/mean for all characters.We also find the mean number of independent curves for each character.

Curvature Histograms. We compute curvature histograms for curved and non-curved characters. The curvature points are computed by iterating over the character, and computing the angle between adjacent points. We use 10-bin histograms, along with histogram entropies. We then compute the Earth Mover's Distance between lower and upper case curvature histograms. For each character, we also find the max and mean curvature, and the entropy of the curvature histograms, then compute the min/max/mean over all characters.

Orientation and Width. To measure the orientation of the characters, such as italic or slanted fonts, we create a point set from the character 'L', compute PCA on the points, then find the angles and magnitudes of the first two principal components. We estimate the stroke width by taking the horizontal width of 'l', 'I', and the '1'. We also estimate the stroke width by the character 'O', both along the x-axis and the y-axis, as well as the ratio of both widths.