Mark Fuge[1]
Department of Mechanical Engineering,
Berkeley Institute of Design,
University of California,
Berkeley, CA 94709
e-mail: mark.fuge@berkeley.edu

Bud Peters
Department of Mathematics,
Berkeley Institute of Design,
University of California,
Berkeley, CA 94709
e-mail: dbpeters@berkeley.edu

Alice Agogino
Department of Mechanical Engineering,
Berkeley Institute of Design,
University of California,
Berkeley, CA 94709
e-mail: agogino@berkeley.edu

Machine Learning Algorithms for Recommending Design Methods

Every year design practitioners and researchers develop new methods for understanding users and solving problems. This increasingly large collection of methods causes a problem for novice designers: How does one choose which design methods to use for a given problem? Experienced designers can provide case studies that document which methods they used, but studying these cases to infer appropriate methods for a novel problem is inefficient. This research addresses that issue by applying techniques from content-based and collaborative filtering to automatically recommend design methods, given a particular problem. Specifically, we demonstrate the quality with which different algorithms recommend 39 design methods out of an 800+ case study dataset. We find that knowing which methods occur frequently together allows one to recommend design methods more effectively than just using the text of the problem description itself. Furthermore, we demonstrate that automatically grouping frequently co-occurring methods using spectral clustering replicates human-provided groupings to 92% accuracy. By leveraging existing case studies, recommendation algorithms can help novice designers efficiently navigate the increasing array of design methods, leading to more effective product design.
[DOI: 10.1115/1.4028102]

1 A Wealth of Design Methods

Every year, researchers and practitioners alike continue to devise different methods for solving increasingly complex design problems. This wealth of design methods is both a blessing and a curse: on one hand, having a wider set of methods deepens our problem-solving toolbox, allowing us to find better solutions; on the other hand, the array of choices quickly becomes overwhelming. For example, the largest current database lists over 300 different design methods [1]—a conservative estimate that easily exceeds any designer's ability to learn or even manually search through.

Dealing with this abundance of methods begs several questions: What makes methods similar, and how does one effectively categorize them? How does one apply methods in different situations, and which differences help designers decide which method to apply? How should designers approach new, unused methods? Past attempts to answer these questions have meticulously reviewed method collections or design case studies (summaries of a particular design problem along with which methods the designer used) to uncover why some methods work in some contexts, but not in others [2–5]. While those types of studies provide a valuable foundation, they necessarily only cover a narrow slice of design methods—scaling that type of analysis to the plethora of current methods requires prohibitive effort.

This paper offers a scalable solution to that problem: it proposes algorithms that automatically learn patterns in design method use by computationally analyzing design case studies. By using case studies as training data, the proposed algorithms can recommend the method or set of methods that are most suitable for a given case. Fortuitously, the wealth of data that previously impeded the analysis of design methods instead now acts as an asset, improving the quality of method recommendations over time.

We evaluate several recommendation algorithms over a corpus of 800+ design case studies from HCD Connect, an online community where designers post case studies of problems they faced along with the user research methods they used to address them. Front-end user research methods have important implications for later design stages [6,7] and Van Pelt and Hey [8] note that decisions made using human-centered design methods directly inform more function-driven methods such as TRIZ [9]. The users on HCD Connect include both IDEO designers along with non-IDEO users working in the social or development design sector such as freelance designers, design students, managers, and entrepreneurs.

The front-end design methods contained in the HCD Toolkit exemplify methods designed to gather and process information about the user requirements of the design. They contrast with later-stage methods, such as axiomatic design, function decomposition, and morphological charts, in that they do not prescribe a step-by-step process for completing a later stage of mechanical design, but rather present methods for understanding user needs and design requirements. Throughout the rest of the paper, "design methods" refers to these earlier-stage methods. Section 5 revisits how one might apply the techniques described in this paper to later-stage design methods.

By analyzing whether methods frequently co-occur with one another or not (method covariance), we find two main results: First, predictions based on method covariance have higher precision–recall performance than predictions based on problem content, and that the combination of the two does not significantly improve performance over just using covariance. Second, by using spectral clustering on the method covariance data, we can automatically divide methods into expert-given groupings with 92% accuracy.

By using the wealth of case studies and methods as an asset, rather than a burden, this paper creates a scalable way of helping novice designers find appropriate methods for a given problem. Moreover, our approach provides a means of exploring the structure of design methods through the use of case study covariance. We have made all of our code and case study data freely available to promote further research to that effect.

2 Background and Related Work

This paper builds off of prior work in two areas: recommender systems and categorizations of design methods.

[1]Corresponding author.
Contributed by the Design Theory and Methodology Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received January 9, 2014; final manuscript received July 21, 2014; published online August 18, 2014. Assoc. Editor: Irem Y. Tumer.

2.1 Recommender Systems. Recommender systems refer to a class of algorithms that recommend content to a user. Some popular applications include Netflix, which uses a person's movie watching habits to recommend new movies, or Google, which uses keywords as well as past browsing behavior to recommend webpages. There is a vast amount of research on this topic, including yearly conferences, such as ACM's RecSys,[2] and we direct interested readers to two recent review papers by Resnick and Varian [10] and Adomavicius and Tuzhilin [11] for a more complete overview. For the purposes of this paper, related efforts can be broken down into three camps, depending on the type of data they use to produce their recommendations: content-based filtering, collaborative filtering, and hybrid filtering.

Content-based filtering bases its recommendation solely on the content of the item itself. For example, if a user says they like comedic movies, Netflix might recommend movies tagged with "comedy" more frequently than those tagged with "drama." This was one of the earliest approaches to recommending content, with its roots in the Information Retrieval community [12]. Popular examples include Google's Page-Rank algorithm [13] as well as text-modeling approaches, such as latent semantic analysis (LSA) [14,15] and latent Dirichlet allocation [16], which build content features by summarizing text content. In the context of design methods, these content features might include the method's textual description or the time required to execute the method. A related field of research which is gaining popularity is the field of "Learning to Rank," which formulates item ranking as a statistical learning problem and uses classification and regression techniques from machine learning to solve ordinal ranking problems. For example, RankNet [17] and ListNet [18] both utilize artificial neural network architectures to determine ranking functions over content features. For a comprehensive overview of learning to rank methods, Liu [19] provides an excellent review. Over the past decade, solely content-based approaches have fallen out of favor for either collaborative filtering models or hybrid models that combine both approaches, as we discuss below.

In contrast to content-based filtering, collaborative filtering bases its recommendation solely on the covariance between users and items. For example, if user A likes the movies "Titantic" and "Caddyshack," and user B likes "Titantic," then the algorithm might conclude that user A and user B are similar, and thus user B might also like "Caddyshack," regardless of the content of the movie itself. For design methods, this would be which cases use which methods—if case study A uses methods 5 and 17, then the algorithm learns something about the relationship between 5 and 17 that it can leverage for future predictions, despite not knowing anything in particular about method 5 or 17. The earliest collaborative filtering methods were neighborhood methods, such as that of Herlocker et al. [20], which used weighted averages of scores from similar users to estimate a new item score. These techniques have been largely replaced by matrix factorization approaches that uncover a latent set of user and item features, representing the score as a cross-product between the two. Their wide-spread usage and popularity are due in part to their independence from content features and in part to the "Netflix Prize" competition, which spurred research from academia and industry alike. Notable examples that emerged from that area include the Bell-Kor system [21], which won the Netflix Prize, as well as techniques, such as Bayesian probabilistic matrix factorization [22], variants of which are currently under active research. A complementary approach used by Nazemian et al. [23] extends standard collaborative-filtering models by encoding new similarity metrics based on transitive properties of user trust to share information beyond the immediate user neighborhood.

Hybrid filtering mixes the above two models by using both content and collaborative features to inform the recommendation, often at the cost of additional computation and complexity. For example, if user A likes "Titantic," "Caddyshack," and "The Shawshank Redemption"; user B likes "Titantic"; and "Titantic" is considered a drama, then a hybrid filtering algorithm might conclude that user A and user B are similar and enjoy dramas, and thus user B might prefer "The Shawshank Redemption" over "Caddyshack," since it is both similar to what user A selected, but also within the "drama" category. This hybrid approach ameliorates some of the disadvantages of the above two models: for new items which do not have collaborative features (referred to as the "cold-start" problem), hybrid models can use content information to improve recommendations; likewise, hybrid models can use collaborative information when item content is not available or informative. Most modern, successful recommender systems use some form of Hybrid Filtering [10,11,21]. For example, Badaro et al. [24] utilize weighted combinations of content- and collaborative-filtering approaches, while Ghazanfar and Prugel-Bennett [25] use neighborhood-based content and collaborative features that are combined using boosting [26]. Hybrid approaches are not without their own problems, however; Yujie and Licai [27] highlight the fact that the increased number of parameters and data sparsity among those parameters can make it difficult to accurately train hybrid methods without sufficient data.

2.2 Categorizing Design Methods. Motivated by a groundbreaking conference on design methods in 1962 [28], researchers have collected and discussed design methods to understand how people design. Collections of design methods, both in print [2–5] and on the web [1,29], manually group methods into various categories by appealing to the author's opinion. We do not know of any studies that validate their categorizations in any formal way such as using inter-rater reliability across multiple raters.

In contrast, this paper uses case studies to train an algorithm that can both group existing methods as well as recommend new methods for a given problem—all without requiring the manual organization needed by previous research. Unlike prescriptive design method categories, this paper takes a descriptive lens (similar to case-based reasoning techniques), by assuming that methods and cases provided by practicing designers represent the ground truth. This complements existing literature on design methods and acts as a means to compare what designers do with what they say they do.

To our knowledge, no previous studies have used computational techniques to categorize and recommend design methods in this way. Within the broader context of engineering design, Panchal and Messer [30] investigate using hierarchical clustering techniques to structure tags on collaborative design platforms, and Li and Ramani [31] parse design documents and concepts to extract a design-specific ontology and retrieve specific design cases. Neither of these applications deals with design methods, specifically.

We should also highlight that there are many kinds of methods used at different stages of the design process. This paper uses methods from IDEO's HCD toolkit, which primarily focuses on early stage design, wherein the goal is to research the appropriate user needs to drive later functional requirements. In contrast, other popular design methods, such as TRIZ [9], axiomatic design [32], or functional decomposition [33], focus on later stages of the design process, once the requirements are set. While the dataset we use in this paper focuses on the front-end design, the method recommendation strategies we propose are not tied to this particular dataset; you can apply the described techniques to methods across a broad scale of design stages, provided that you have case studies that use those methods. Presently, Roschuni et al. [1] are developing such a set of cases, and that dataset could also use the recommendation strategies we describe herein for a wider range of stages in the design process.

[2]http://recsys.acm.org/

Table 1 Examples of the 886 design method case studies from HCD Connect. They contain problem descriptions as well as the human-selected methods used to solve that problem and the tagged "Focus Area" of the problem.

Example problem statements	HCD Connect methods used in case study
As we worked side by side with small-holder farmers in Peru to harvest coffee, we learned that there were many things we could improve to make our device easier to use. **Focus area:** agriculture	Individual interview, in-context immersion, community-driven discovery, capabilities quick sheet, and participatory codesign
Butcher block furniture is popular in the United States. However, in India, there is a whole market for recycling waste wood. This recycling can be better if the wooden pieces are adhered together and then made into furniture. As in butcher block furniture, here also pieces of wood are put together to for a plank for furniture. **Focus areas:** environment and community development	Storytelling with purpose, try out a model, individual interview, inspiration in new places and innovation 2×2
In collaboration with the American Refugee Committee and IDEO.org design team, IDEO.org colead, Jocelyn Wyatt, shares her experiences facilitating codesign sessions with women in the Democratic Republic of Congo in order to gain insights on bringing health, water, and nutrition solutions to the community. **Focus areas:** water, community development, and health	Storyboards, role-play, track indicators, and evaluate outcomes

3 Methodology

To evaluate different algorithms for design method recommendation, we used data from IDEO's HCD Connect platform,[3] an online community of design practitioners who upload design case studies along with which methods they used to address the problem—at the time of our experiment, the site contained information about 886 design cases. Table 1 lists some illustrative case study examples. Each case study used some subset of the 39 methods (Fig. 1) available through IDEO's Human-Centered Design Toolkit,[4] and contained four pieces of information (Fig. 2): (1) a textual description of the design problem; (2) the location of where the problem originated from; (3) one or more "focus areas" such as "healthcare"; and (4) a description of the user who contributed the problem. Some cases also included photos, dates, and follow-ups from HCD Connect users, but we did not use those attributes in our experiments.

It should be noted that individual cases in HCD Connect can utilize multiple methods. As we will see below, this is because different methods complement each other; for example, a project evaluating mobile phone applications for healthcare might use interviewing methods to gather user feedback on a prototype, while a storyboarding method could evaluate a user's workflow. In such situations, methods would be positively correlated with one another. In other situations, one might expect methods to substitute for one another; for example, if someone has already conducted an individual interview then they might be less likely to perform other types of interviews. In that case, methods would be negatively correlated with one another. For the 39 methods in IDEO's HCD Toolkit, we found almost no incidence of methods being negatively correlated with one another, meaning that IDEO's methods did not frequently substitute for one another.

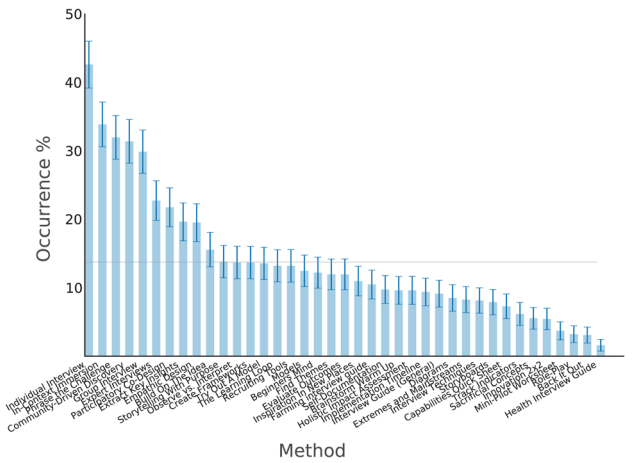

Fig. 1 HCD Connect users use different methods with different frequencies. Error bars represent 95% confidence bounds around the frequency estimates, calculated using bootstrap resampling. The gray line represents the average method frequency (≈14%).

3.1 Content-Based Filtering. Our content-based filtering strategy for recommending design methods involves summarizing the problem descriptions and then using that text to predict which methods are most relevant for a given problem—the intuition being that design problems that have similar problem descriptions may use similar methods. We use latent semantic indexing (LSI is also referred to as LSA) to quantify that similarity [12]. LSI employs the bag-of-words model for representing a text document, which ignores word order and grammar, and considers only frequency of word occurrence. Given the 886 case study descriptions, we use singular value decomposition to project the word count matrix into a latent space of 50 topics. The resulting 886×50 matrix M contains a row for each case study and 50 columns representing the case's similarity to each of the 50 topics. The algorithm then uses this topic matrix to train a classifier, which outputs the probability that a given method m will be used in each design case c.

We evaluated four different algorithms:

- Random forests: an ensemble classification technique that fits a number of decision tree classifiers to randomized subsamples of the dataset; it uses these subsamples to rule out non-useful features and gives us a straightforward method of discarding unimportant text topics.
- Support vector machines: nonlinear classifiers that construct a hyperplane in high-dimensional space; it identifies complex boundaries between topics and their interactions.
- Logistic regression: a type of generalized linear model used for the classification; it provides a simple method for determining important text topics (like random forests) and is computationally efficient as the number of cases increases.
- Naive Bayes: a probabilistic classifier that assumes feature independence across variables and applies Bayes' rule to label categorical data; it also provides a simple method for

[3] http://HCDConnect.org/
[4] http://www.ideo.com/work/human-centered-design-toolkit/

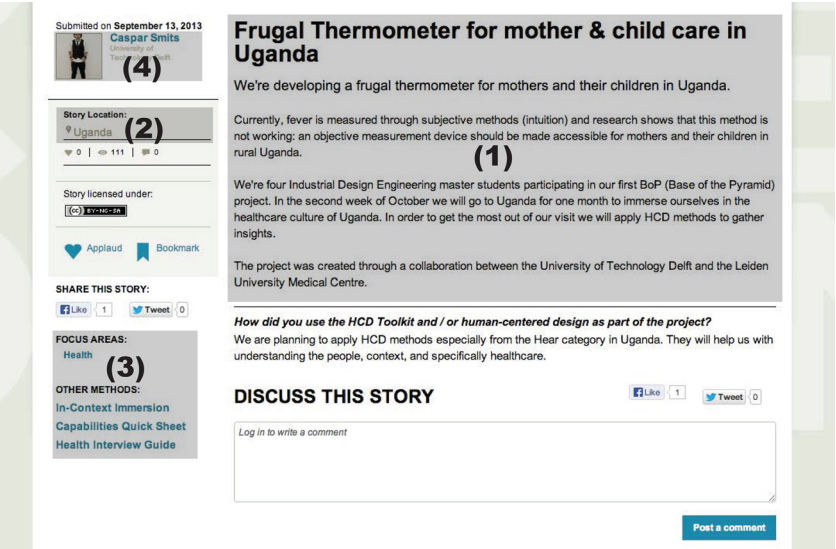

Fig. 2 Each case page on HCD Connect contains a textual description about the design problem (1), as well as contextual labels such as location (2), focus area (3), and user occupation (4)

determining important text topics as well as easily scaling to a larger number of cases.

For all algorithms, we optimized any hyperparameters using randomized search with cross validation using the Scikit-Learn library [34]. Interested readers can download our experiment code and data to reproduce our results or get more specific information about which parameters we optimized over.[5]

3.2 Collaborative Filtering. Instead of using problem content to determine which methods are most relevant for a given problem, collaborative filtering approaches analyze the methods that commonly occur together. Such an approach is valuable when the collaborators, in this case, the contributors to HCD Connect, provide high quality content. Our results benefit from a well-curated dataset, consisting primarily of cases contributed by IDEO designers [35].

To visualize the co-occurrence, we calculate the method covariance matrix—this matrix describes how different methods covary with one another, capturing common usage patterns (similarly to a correlation coefficient). Specifically, we use the graphical Lasso [36] because the empirical covariance produces poor eigenvalues estimates (necessary for the spectral clustering we describe next).

To gauge the suitability of method covariance, we performed spectral clustering on the covariance matrix and colored the resulting clusters to visualize method groups (Fig. 3)—colors refer to clusters and opacity refers to the covariance. Using three clusters, we compare the groupings to those of HCD Connect, who grouped their 39 methods into three categories (Hear, Connect, Deliver). Our clusters agreed with the HCD Connect provided clusters to an average of 92% accuracy (36 of 39 methods).

Given the utility of method covariance in clustering, we construct a collaborative filtering model for recommendation inspired by the BellKor solution to Netflix Prize Challenge [21]:

$$f(c,m) = b_c + b_m + \langle v_c, v_m \rangle \tag{1}$$

where $f(c, m)$ represents the score for a particular method m when applied to case c. b_c represents a baseline score for a given case (some cases use more methods than others), and similarly b_m represents a baseline score for a given method (some methods are more popular than others, regardless of the case).

The inner product $\langle v_c, v_m \rangle$ captures the interaction between methods and cases; v_c and v_m refer to latent dimensional vectors of length k, with a separate vector for each case and method, respectively. For example, using $k = 2$ places each case and method onto a 2D plane. If the two vectors lie close to one another in the 2D space, they get a large positive score; if far away, a large negative score. One can optimize the exact dimension k, which we address below.

The algorithm determines the values for k, b_c, b_m, v_c, and v_m, by minimizing prediction error: the mismatch between the methods that it recommends and the methods that were actually used. To encode this error, we use a logarithmic loss of the following form, where $y_{(c,m)} \in \{1, -1\}$ represents whether or not the case actually used the method

$$\mathcal{L}(f(c,m),y) = \sum_{c \in \mathcal{C}} \sum_{m \in \mathcal{M}} \ln\left(1 + \exp(-y_{(c,m)} \cdot f(c,m))\right) \tag{2}$$

Evaluating Eq. (1) for each c, m pair yields the expected recommendation score, and Eq. (2) encourages that score towards $+\infty$ for appropriate methods, and towards $-\infty$ for unused methods. The algorithm uses quadratic (L2) regularization to prevent the latent factors from overfitting the training data (improving performance on future data). Combining the loss function in Eq. (2) with the regularization, the total loss function across the entire dataset becomes

$$\mathcal{L}(f(c,m),y) + \frac{\lambda}{2}\left[\sum_c \|v_c\|^2 + b_c^2 + \sum_m \|v_m\|^2 + b_m^2\right] \tag{3}$$

In our experiments, we use stochastic gradient descent to minimize the combined loss function in Eq. (3), though any descent-based optimizer would suffice since the loss function is convex.

3.3 Hybrid Models. Our hybrid model incorporates both content and collaborative information by adding case-dependent "focus-area" terms into our collaborative filtering model. Focus areas (tags given by the HCD Connect community) describe which areas the case focuses on, such as "health," "education," or

[5] www.markfuge.com/hcdconnect

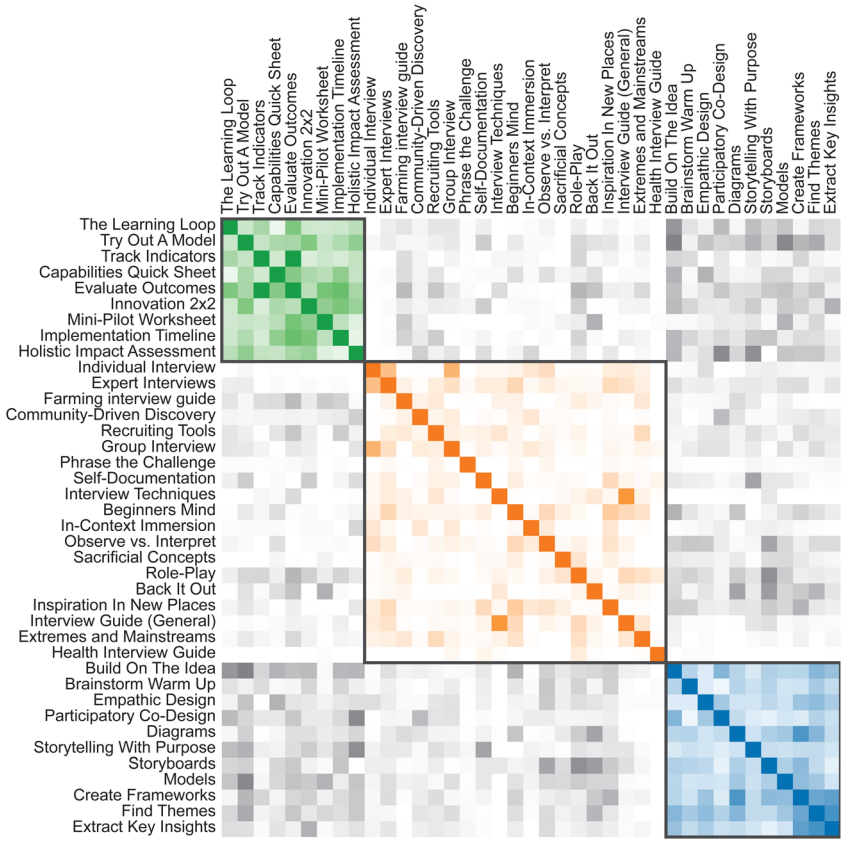

Fig. 3 We performed spectral clustering on the 39 × 39 method covariance matrix revealing groups of methods that covary together. Lighter tones represent low covariance, while darker tones represent high covariance. The different hues denote different clusters, with a dark gray box around each cluster of methods. The clusters found by spectral clustering accurately reflect the expert-given categories used by IDEO in their HCD Toolkit; from left to right, the boxes on the diagonal correspond to "Deliver," "Hear," and "Create" methods, respectively.

"development" among others (see the first column of Table 1 for examples).

To add these content features to our collaborative filtering model, we give each focus area its own k-dimensional latent vector (like the methods and cases) and then optimize the locations of those vectors for each focus area. We chose to use focus areas as a content feature because, intuitively, the useful methods for one focus area (e.g., agriculture), may not be the most useful in a different focus area (e.g., healthcare). This adds an additional term (v_{cont}) to the collaborative filtering model in Eq. (1):

$$f(c,m) = b_c + b_m + \langle v_c, v_m \rangle + \left\langle v_c, \sum_{\text{cont} \in c} v_{\text{cont}} \right\rangle \tag{4}$$

where the last summation refers to the addition of all content vectors present in the case. The innerproduct $\langle v_c, v_{cont} \rangle$ acts like the previous innerproduct $\langle v_c, v_m \rangle$, measuring the similarity between the case vector and the combined content vectors. With the exception of the added content vectors, all other aspects of the model are identical to the collaborative filter.

4 Results

On all models, we used an 80/10/10 stratified random split of the data for training, optimization, and testing, respectively. For hyperparameter optimization, we performed 100 iterations of randomized search with fourfold cross validation over all model parameters in each model. For each algorithm, we tested the best performing parameter choices on the remaining unseen testing data to compute their respective performances.

We evaluated each algorithm's performance using a precision–recall curve, a standard way of comparing different recommender systems [37,38]. The curve trades off two quantities: precision and recall. Precision is the percentage of recommended methods that were actually used in a case—if the algorithm recommends ten methods, but only five of those ten were used in the case study, the precision would be 50%. Recall is the percentage of methods actually used in a case that were recommended by the algorithm—if the case actually used eight methods, and the algorithm only correctly recommended six of those methods, then the recall would be 75%. By changing the number of methods, the algorithm was allowed to recommend (i.e., from 0 to 39 methods); we evaluated the system's performance over a range of precision and recall values—this created a precision–recall curve, which we plotted to evaluate algorithm performance. This curve essentially summarizes how well the algorithm presented users with meaningful methods for their design problem. (Companies such as Google use similar metrics to evaluate performance for related tasks like webpage ranking [39].) In our use case, higher degrees of precision are more important than higher degrees of recall. Namely, a small set of highly relevant methods is a more valuable recommendation than a complete set including many lower-ranked methods.

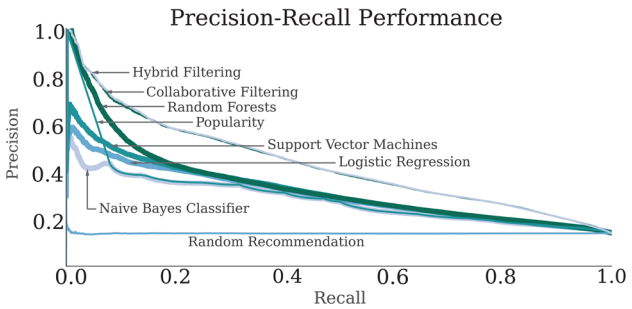

Fig. 4 Precision plotted as a function of recall. The higher the AUC, the better the algorithm's performance.

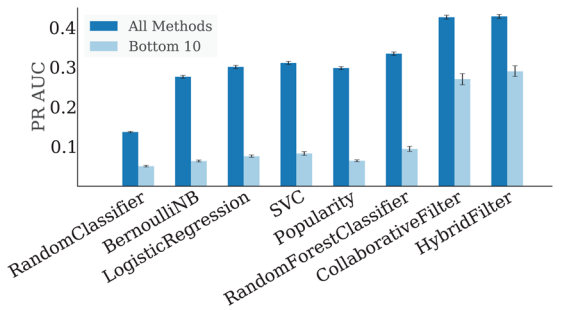

Fig. 5 The area under the precision–recall curve (AUC) across the models. The error bars represent the 95% empirical confidence bounds about the median AUC for each method, calculated using bootstrap resampling. The hybrid and collaborative filtering models perform substantially better than the popularity baseline. The Random Forest classifier produces a detectable, but small, improvement over the popularity baseline.

For comparative purposes, we also tested the performance of two baseline algorithms: randomized recommendation and popularity-based recommendation. Randomized recommendation randomly selects k of the 39 methods for recommendation. Popularity-based recommendation rank-orders the most frequently used methods and simply recommends the first k most popular methods, regardless of the case.

Figure 4 shows the precision–recall curve for each algorithm. Figure 5 demonstrates the 95% confidence bounds (using bootstrap resampling) for the area under those curves—higher area indicates better average precision, and thus better recommendations. As expected, all models outperform the randomized baseline. Popularity performs slightly below that of text-based analysis using random forests, while using only a single, efficient predictor. Collaborative filtering uniformly outperforms both the popularity baseline as well as the text-based content features; the added content features in the hybrid model do not discernibly improve the performance. (We expect that future research will uncover different content features that positively affect performance.)

In addition to general performance, one might also be more interested in how the algorithms perform on more specific or uncommon methods. A designer would likely use popular methods regardless (possibly out of habit), but might only use certain uncommon methods when particularly appropriate—a successful method recommendation algorithm should perform well over uncommon methods, as well as popular ones. To test this, we compute the precision–recall performance on the ten least frequently used of the 39 methods, and integrate the area under the precision–recall curve in Fig. 5; this total area is called the area under the curve (AUC) and measures overall recommendation performance (Higher AUC is better). Collaborative and hybrid filtering still perform significantly better than the alternatives.

5 Implications for Recommending Design Methods

Our results offer up several possible implications for design method recommendation systems. First, one should not ignore collaborative features in favor of text features, especially when the collaborators have a reasonable level of expertise. Second, future research needs to maximize the benefits of combining content and collaborative features. Finally, collaborative features help in not only recommending methods, but also in grouping and understanding the methods themselves.

5.1 Collaborative Features Have Higher Predictive Accuracy Than Text-Based Features. Comparing the precision–recall performance, collaborative-based approaches perform substantially better than the content-based approaches that relied solely on text features. We did not expect this, given the prevalence of text-based recommendation for ranking documents. However, given the use case, this is also understandable—the time needed to apply a method or the people required to execute it (among many other factors) could both affect a method's usage in ways not discernable from the case's description.

One possible explanation for the fact that the content-based features offered little improvement is that the methods and focus areas could be too general to meaningfully distinguish themselves. For HCD Connect, prior studies have demonstrated that a small subset of methods do occur more frequently depending on the specific focus area [35], so we would expect the addition of focus areas to have a meaningful effect. That said, a more thorough description or ontology of methods that accounts for these differences between methods or categories may improve future performance of content-based recommender systems, and some recent work has begun to collect this information [1]. Incorporating improved content features would be a fruitful area of future research. Regardless, we recommend starting with collaborative filtering as a baseline.

5.2 Combinations of the Features Offered Only Marginal Improvement. In our experiments, combining content features (in the form of focus areas) with collaborative filtering did not offer a detectable improvement in performance. This could be attributed to our choice of content features, or possibly to our choice of model in Eq. (4) (although similar models are effective in other domains [21]). We recommend choosing a hybrid model over models that rely solely on method or case content, since the techniques useful for improving content-based recommendation can also be applied to improve hybrid models. More advanced collaborative filtering models, such as Bayesian probabilistic matrix factorization models, could also improve recommendation performance by incorporating prior knowledge or more sophisticated content features [12,22].

5.3 Collaborative Features Offer a Useful Means of Grouping Methods for Further Analysis. We found that collaborative features, such as method covariance, accurately grouped related methods together—in this case replicating human-given groupings to 92% accuracy. When attempting to decompose methods into different types, we recommend exploring method co-usage as a similarity criterion. Future work could certainly explore more complex methods of clustering or grouping methods, including using a combination of collaborative and content features to define method similarity.

5.4 Additional Applications. Although we have chosen to evaluate this recommendation system against the HCD Connect dataset and the corresponding methods, the proposed algorithms

do not depend on the choice of methods used in the dataset; they are agnostic towards the choice of methods and evaluate only the relationships between cases and methods. This means that design method recommendation systems can be generalized beyond HCD Connect and user research methods at the beginning of the design process. Design practitioners can use these techniques for other classes of design methods, such as design decision methods [40], mechanical design techniques [41], functional synthesis [42], design affordance techniques [43], design team formation [44], and more, so long as one can find appropriate case studies that use those methods. An appropriate dataset for that scope of analysis is the DesignExchange [1], and we intend to expand these proposed algorithms to their larger dataset in the future. One possible area of future work would involve extending design method recommendation systems beyond the realm of product case studies and into applications like analyzing methods used in different patents (similar in spirit to TRIZ [9]); this is beyond this paper's scope, but could be an interesting future application.

6 Conclusion

This paper explored different machine learning algorithms for recommending design methods. Collaborative filtering approaches that leverage information about method covariance had better precision–recall performance than models that used exclusively textual descriptions or method popularity. A hybrid model that blended collaborative filtering with the focus area of the design problem did not offer a large improvement—more informative content features (e.g., those based on structured ontologies [1]) may increase the performance of hybrid models in future research. Finally, spectral clustering using method covariance grouped methods to 92% accuracy with human-provided groupings. Future research can build off this paper by using our code and dataset to further explore how methods connect to one another.

The results imply that merely looking at textually similar problems cannot discern what design methods to use for a given problem—a strategy employed by most search engines or information retrieval systems. Rather, looking at methods covariance provides a stronger basis for making recommendations. While there may be content features that make hybrid models more successful, we saw no benefit from including focus area specializations, such as "energy" or "health" in our collaborative filtering model—future research should extend our understanding of possible alternate features.

Our research enables novice designers to quickly come to grips with which methods to use by recommending design methods for a given problem. In addition, our research helps further the understanding of how design methods relate to one another by proposing covariance as a meaningful basis for similarity. Both of these have benefits not only for practicing designers, but also for education and training—they provide a scalable way to make the increasing web of design methods more manageable.

Acknowledgment

We would like to thank the community members of HCD Connect, without whose efforts this work would not have been possible. We also thank the reviewers for their efforts in improving the manuscript. This work was supported by NSF CMMI-1334361 and the Department of Defense (DoD) through the National Defense Science and Engineering Graduate Fellowship (NDSEG) Program.

References

[1] Roschuni, C., Agogino, A., and Beckman, S., 2011, "The DesignExchange: Supporting the Design Community of Practice," International Conference on Engineering Design, International Conference on Engineering Design (ICED '11), Vol. 8, pp. 255–264.
[2] Broadbent, G., and Ward, A., 1969, *Design Methods in Architecture*, AA Papers, Lund Humphries.
[3] Broadbent, G., 1979, "The Development of Design Methods," Des. Methods Theor., **13**(1), pp. 41–45.
[4] Jones, J. C., 1992, *Design Methods*, 2nd ed. Wiley, John Wiley and Sons, New York.
[5] Margolin, V., and Buchanan, G. R., 1996, *The Idea of Design*, The MIT Press, Cambridge, MA.
[6] McCarthy, J. M., 2005, "Engineering Design in 2030: Human Centered Design," ASME J. Mech. Des., **127**(3), p. 357.
[7] Collopy, P., 2013, "Opportunities in Engineering Design Research," ASME J. Mech. Des., **135**(2), p. 020301.
[8] Van Pelt, A., and Hey, J., 2011, "Using TRIZ and Human-Centered Design for Consumer Product Development," Procedia Eng., **9**, pp. 688–693.
[9] Altshuller, G., Shulyak, L., Rodman, S., and Fedoseev, U., 1998, *40 Principles: TRIZ Keys to Innovation*, Vol. 1, Technical Innovation Center Inc., Worcester, MA.
[10] Resnick, P., and Varian, H. R., 1997, "Recommender Systems," Commun. ACM, **40**(3), pp. 56–58.
[11] Adomavicius, G., and Tuzhilin, A., 2005, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," IEEE Trans. Knowledge Data Eng., **17**(6), pp. 734–749.
[12] Manning, C. D., Raghavan, P., and Schütze, H., 2008, *Introduction to Information Retrieval*, Cambridge University Press, New York.
[13] Page, L., Brin, S., Motwani, R., and Winograd, T., 1999, "The PageRank Citation Ranking: Bringing Order to the Web," Technical Report No. 1999-66, Stanford InfoLab. Previous No. SIDL-WP-1999-0120.
[14] Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A., 1990, "Indexing by Latent Semantic Analysis," JASIS, **41**(6), pp. 391–407.
[15] Dong, A., Hill, A. W., and Agogino, A. M., 2004, "A Document Analysis Method for Characterizing Design Team Performance," ASME J. Mech. Des., **126**(3), pp. 378–385.
[16] Blei, D. M., Ng, A. Y., and Jordan, M. I., 2003, "Latent Dirichlet Allocation," J. Mach. Learn. Res., **3**, pp. 993–1022.
[17] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G., 2005, "Learning to Rank Using Gradient Descent," Proceedings of the 22nd International Conference on Machine Learning, ICML'05, ACM, pp. 89–96.
[18] Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H., 2007, "Learning to Rank: From Pairwise Approach to Listwise Approach," Proceedings of the 24th International Conference on Machine Learning, ICML'07, ACM, pp. 129–136.
[19] Liu, T.-Y., 2007, "Learning to Rank for Information Retrieval," Found Trends Inf. Retrieval, **3**(3), pp. 225–331.
[20] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J., 1999, "An Algorithmic Framework for Performing Collaborative Filtering," Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'99, ACM, pp. 230–237.
[21] Bell, R. M., Koren, Y., and Volinsky, C., 2007, The BellKor Solution to the Netflix Prize.
[22] Salakhutdinov, R., and Mnih, A., 2008, "Bayesian Probabilistic Matrix Factorization Using Markov Chain Monte Carlo," Proceedings of the 25th International Conference on Machine Learning, ICML'08, ACM, pp. 880–887.
[23] Nazemian, A., Gholami, H., and Taghiyareh, F., 2012, "An Improved Model of Trust-Aware Recommender Systems Using Distrust Metric," IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 1079–1084.
[24] Badaro, G., Hajj, H., El-Hajj, W., and Nachman, L., 2013, "A Hybrid Approach With Collaborative Filtering for Recommender Systems," 9th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 349–354.
[25] Ghazanfar, M. A., and Prugel-Bennett, A., 2010, "A Scalable, Accurate Hybrid Recommender System," Proceedings of 3rd International Conference on Knowledge Discovery and Data Mining, pp. 94–98.
[26] Freund, Y., and Schapire, R. E., 1997, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," J. Comput. Syst. Sci., **55**(1), pp. 119–139.
[27] Yujie, Z., and Licai, W., 2010, "Some Challenges for Context-Aware Recommender Systems," 5th International Conference on Computer Science and Education (ICCSE), pp. 362–365.
[28] Jones, J. C., and Thornley, D., eds., 1962, *Conference on Design Methods: Papers Presented at the Conference on Systematic and Intuitive Methods in Engineering*, Industrial Design, Architecture and Communications, Pergamon, Pergamon Press, Oxford, UK.
[29] Helen Hamlyn Centre for Design, 2013, "Designing With People: Methods," http://designingwithpeople.rca.ac.uk/methods
[30] Panchal, J. H., and Messer, M., 2011, "Extracting the Structure of Design Information From Collaborative Tagging," ASME J. Comput. Inf. Sci. Eng., **11**(4), p. 041007.
[31] Li, Z., and Ramani, K., 2007, "Ontology-Based Design Information Extraction and Retrieval," AI EDAM, **21**(4), pp. 137–154.
[32] Suh, N. P., 2001, *Axiomatic Design: Advances and Applications* (The Oxford Series on Advanced Manufacturing), Oxford University, Oxford University Press, New York.
[33] Pahl, G., Beitz, W., Feldhusen, J., and Grote, K.-H., 1984, *Engineering Design: A Systematic Approach*, Springer-Verlag, London, UK.

[34] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., 2011, "Scikit-Learn: Machine Learning in Python," J. Mach. Learn. Res., **12**, pp. 2825–2830.
[35] Fuge, M., and Agogino, A., 2014, "User Research Methods for Development Engineering: A Study of Method Usage With IDEO's HCD Connect," ASME International Design Engineering Technical Conferences, Buffalo, NY, August 17–20.
[36] Friedman, J., Hastie, T., and Tibshirani, R., 2008, "Sparse Inverse Covariance Estimation With the Graphical Lasso," Biostatistics, **9**(3), pp. 432–441.
[37] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T., 2004, "Evaluating Collaborative Filtering Recommender Systems," ACM Trans. Inf. Syst., **22**(1), pp. 5–53.
[38] Wu, W., He, L., and Yang, J., 2012, "Evaluating Recommender Systems," 7th International Conference on Digital Information Management (ICDIM), pp. 56–61.
[39] Gordon, M., and Pathak, P., 1999, "Finding Information on the World Wide Web: The Retrieval Effectiveness of Search Engines," Inf. Process. Manage., **35**(2), pp. 141–180.
[40] Wood, W. H., and Agogino, A. M., 2005, "Decision-Based Conceptual Design: Modeling and Navigating Heterogeneous Design Spaces," ASME J. Mech. Des., **127**(1), pp. 2–11.
[41] Hernandez, N. V., Schmidt, L. C., and Okudan, G. E., 2013, "Systematic Ideation Effectiveness Study of TRIZ," ASME J. Mech. Des., **135**(10), p. 101009.
[42] Kalyanasundaram, V., and Lewis, K., 2014, "A Function Based Approach for Product Integration," ASME J. Mech. Des., **136**(4), p. 041002.
[43] Srivastava, J., and Shu, L. H., 2013, "Affordances and Product Design to Support Environmentally Conscious Behavior," ASME J. Mech. Des., **135**(10), p. 101006.
[44] Fuge, M., Tee, K., Agogino, A., and Maton, N., 2014, "Analysis of Collaborative Design Networks: A Case Study of OpenIDEO," ASME J. Comput. Inf. Sci. Eng., **14**(2), p. 021009.