Personalizing Software and Web Services by Integrating Unstructured Application Usage Traces

Longqi Yang†, Chen Fang‡, Hailin Jin‡, Matthew D. Hoffman∗, Deborah Estrin†
†Computer Science, Cornell Tech, Cornell University; ‡Adobe Research; ∗Google
†ylongqi@cs.cornell.edu, destrin@cornell.edu
‡{cfang, hljin}@adobe.com; ∗matt@matthewdhoffman.com

ABSTRACT

Users of software applications generate vast amounts of unstructured log-trace data. These traces contain clues to the intentions and interests of those users, but service providers may find it difficult to uncover and exploit those clues. In this paper, we propose a framework for personalizing software and web services by leveraging such unstructured traces. We use 6 months of Photoshop usage history and 7 years of interaction records from 67K Behance users to design, develop, and validate a user-modeling technique that discovers highly discriminative representations of Photoshop users; we refer to the model as utilization-to-vector, util2vec. We demonstrate the promise of this approach for three sample applications: (1) a practical user-tagging system that automatically predicts areas of focus for millions of Photoshop users; (2) a two-phase recommendation model that enables cold-start personalized recommendations for many new Behance users who have Photoshop usage data, improving recommendation quality (Recall@100) by 21.2% over a popularity-based recommender; and (3) a novel inspiration engine that provides real-time personalized inspirations to artists. We believe that this work demonstrates the potential impact of unstructured usage-log data for personalization.

Keywords
User modeling; application usage; recommendation

1. INTRODUCTION

Modern software services typically record user actions for the purpose of collecting application usage statistics and reproducing program errors. Relative to structured data traces, such as text, image, and search queries, application usage records are noisy and unstructured, and service providers may find it more difficult to extract value from them. Just as social interaction traces have enabled great success in personalizing online communities, we explore how integrating application usage traces can further empower novel, effective and personalized services, as shown in Fig. 1.

In this paper, we explore this largely untapped space using data from a large number of creative professionals who use Photoshop for work and actively socialize on Behance, a popular large-scale online community where millions of professional photographers, designers and artists share their artwork. We demonstrate that by leveraging the data traces from shared users, Photoshop and Behance can provide significantly improved personalized services and create new user experiences. Our contributions in this work are summarized below.

*Work done while working at Adobe Research

©2017 International World Wide Web Conference Committee (IW3C2), published under Creative Commons CC BY 4.0 License.
ACM 978-1-4503-4914-7/17/04.
http://dx.doi.org/10.1145/3041021.3054183

Figure 1: Services that can benefit from the integration of application usage traces. By leveraging user representations learned from sequences of actions, we explore how software service providers and social platforms can improve their personalized services and empower new user experiences.

1https://www.behance.net/

485
2. Related Work

Our work benefits from and has implications to multiple threads of user modeling research, and the util2vec model is inspired by previous work on distributed representation learning.

2.1 Distributed representation learning

Distributed representation learning was first introduced in the area of natural language processing [23]. The goal is to learn a vector space for all words so that they can be used as inputs to natural language understanding algorithms [21]. Recently, such an approach has been extended and successfully applied to paragraphs [17], medicine [4] and online purchases [6]. For instance, Grbovic et al. [4] proposed a framework to learn a vector representation for each product and user given the historical purchasing records, and Choi et al. [4] demonstrated that a similar approach can be applied to learn hierarchical representations for medical records. Ultimately, our model enables personalized recommendations for a massive number of new users with Photoshop usage histories (Section 5.2).

3. Dataset

We propose a two-phase recommendation method that generates more accurate recommendations for cold-start users on Behance by leveraging their previous Photoshop usage traces. The performance improvements over the popularity baseline are significant on all tested metrics including Area Under Curve (AUC) (6.8%) and Recall@K (21.2% when K = 100). Ultimately, our model enables personalized recommendations for a massive number of new users with Photoshop usage histories (Section 5.2).

3. We design a novel application, named the inspiration engine, for Photoshop users by leveraging the co-occurrences of application usage traces and uploaded art projects on Behance. The qualitative results demonstrate how integrating these data sources enable new user experiences (Section 5.3).

Although the data used in this paper comes from creative professionals, the models and frameworks studied may be applied to personalize services in a broader range of scenarios. Given the evolution of app ecosystems, user activities across stand-alone software applications and social platforms are more easily associated via a small number of identity systems, e.g., Gmail, Facebook, Creative Cloud, Apple, and Github. This opens up a new and fruitful space of future user-modeling research for service providers as well as open source communities.

The technical content of this paper is structured as follows. We introduce our dataset in Section 3 followed by the util2vec model in Section 4. Then we present three models and applications leveraging usage traces in Section 5.

2. Software user and command modeling

Modeling software users’ proficiency based on the actions they perform has been previously studied in the context of command-recommendation systems [20, 18, 6]. The goal of such a recommender is to help users learn commands in a complex software application. However, the user modeling under such a circumstance is limited to a specific application because of the narrow scope that the modeling system is exposed to. In this work, we show that by integrating application usage traces with online social interactions, the potential applications that such data traces can empower are much broader and diverse. Specifically, we demonstrate that the Photoshop service provider can conduct better user tagging and create new user experiences.

Another line of related work around application usage records attempts to understand the semantic meanings of software actions [1]. By training a word2vec model [21] on online documents, previous work [1] discovered correspondences and relationships between natural language and software actions, which was used to fuel tutorial-recommender systems. Although our work is not directly optimized for this task, we can still extract semantic meanings of actions and their relationships to users’ social interactions, because the actions, along with the users, are embedded in the same high-dimensional feature space (Section 4).

Although previous research on user modeling has achieved great success, most models only consider data from within the online social platforms. In our work, we demonstrate that by leveraging users’ digital traces from application usage records, online social platforms can better understand users and provide more effective recommendations.

3. Dataset

We associate action histories from Photoshop with social interactions on Behance through Creative Cloud accounts as people use them to log into both services. The reasons why we choose these two platforms are three-folded. (1) Photoshop is one of the most popular computer software applications used by creative professionals, and it is an indispensable daily component for people across many creative occupations including graphical designer, photographer, and architect. Therefore, it is an ideal context in which to study and impact users’ working behavior at a large scale. (2) Behance possesses an abundant user base as millions of creative professionals share their work and socialize with each other on the platform. Also, it is one of the major websites for creative talent search. (3) As Photoshop and Behance both serve creative professionals, there are many shared users for us to investigate.

In Photoshop, all of the actions performed in the application, e.g., buttons clicked and features applied, are collected from the users who enabled application usage reporting. An example of the action sequence is shown in Fig. 2. We target a group of users from the U.S. and their action histories from January 2015 to June 2015. We selected 22 billion actions from 3 billion unique Photoshop users. From the Behance platform, we collected users’ social interactions in three categories: (1) self-disclosed areas of focus, e.g., Cartooning, Interaction Design and Fashion; (2) user-uploaded projects; and (3) users’ view and appreciate history on these projects. An example of the collected information from Be-
Figure 2: Data samples of Photoshop usage records (left) and social interactions on Behance (right). We collected three categories of social interactions for each user: projects viewed, self-disclosed areas of focus and uploaded projects.

... an example of the collected user’s social interactions on Behance uploaded projects

(a) An example of action sequence in Photoshop
(b) An example of the collected user’s social interactions on Behance

hance is shown in Fig. 2 In this paper, we select 0.86 million behance users where 67 thousand of them are also among the Photoshop users mentioned above.

4. SOFTWARE USER REPRESENTATION

In this section, we propose an accurate and robust user modeling framework to model the action histories of software users. We start by introducing the model, followed by implementation details and performance evaluations.

4.1 util2vec framework

Given the action history \( H_u = (a^u_1, a^u_2, ..., a^u_n) \) from a software user \( u \), our goal is to learn a fixed-length real-valued vector \( v_u \) that represents his/her software usage pattern. We propose a framework named util2vec to learn the user representation. In our framework, each user or action is mapped to an \( M \)-dimensional vector, and the vectors are trained to maximize the log probability, as defined in eqn. 1 across all users.

\[
\frac{1}{1 - 2K} \sum_{t=1}^{T-K} \log p(a^u_{t+K} | a^u_{t-K}, ..., a^u_t) 
\]

where \( T \) is the total number of actions from a given user, and \( K \) is the farthest action before/after the prediction target that is used as the context. In other words, the size of the sliding window is \( 2K + 1 \). Intuitively, the model optimized for the objective defined in eqn. 1 will be able to predict any action given the context of the user and the surrounding actions.

For the prediction, we use the softmax function to model the conditional probability \( p(a_t | a_{t-K}, ..., a_{t-K}) \) as follows (We omit the superscripts of \( a^u_t \) which are clear from context).

\[
p(a_t | a_{t-K}, ..., a_{t-K}) = \frac{e^{v_u}}{\sum e^{v_u}}
\]

where the vector \( p = b + Wh(u, a_{t-K}, ..., a_{t-K}) \); the bias vector \( b \) and weight matrix \( W \) are parameters of the model, and the columns of the matrices \( V \) and \( X \) store the user and action representations respectively, i.e., \( v_u = V \cdot [u] \) and \( x_t = X \cdot [t] \) in *numpy*-style notation. The parameters \( b, W, V, \) and \( X \) are learned during training. In the util2vec framework, we use a transfer function \( h \) that averages or concatenates a user representation with representations from 2\( K \) context actions, as Fig. 3 shows.

We use Stochastic Gradient Descent (SGD) to conduct the training. The model is trained with action histories from \( U \) unique Photoshop users (\( U = 3 \) million), and the user and action representations are updated concurrently. After the model is trained, we can infer a new user \( a \)'s representation \( v_u \) by fixing the parameters \( b, W, X \) and only fitting the vector \( v_u \) to user \( a \)'s action history.

4.2 Implementation details

Along with util2vec, we use negative sampling and additional action preprocessing steps to speed-up the training and reduce the noise, which will be discussed next.

4.2.1 Negative sampling

It is expensive to compute the softmax function in eqn. 2 since the denominator involves a sum over a large number of unique actions. To avoid this cost, we replace the softmax loss with a negative-sampling loss. This strategy has been successfully applied in the word2vec model [21]. Specifically, for each instance, we randomly sample \( S \) actions that are different from the target action \( a_t \) and approximate the log probability \( \log p(a_t | a_{t-K}, ..., a_{t-K}) \) as follows:

\[
\log \left( \sigma(y_n) \right) + \sum_{s \in S} \log \left( \sigma(-y_n) \right)
\]

where \( S \) is a set of randomly sampled actions such that \( a_t \not\in S \), and \( \sigma \) is the sigmoid function \( \sigma(x) \equiv \frac{1}{1+e^{-x}} \).

4.2.2 Preprocessing and parameter settings

**Preprocessing:** For each action, we keep it in the vocabulary only if it is used by at least 100 unique users, and the final size of the vocabulary is 1990. During the preprocessing, we also add a special separation token \( [E] \) between two sessions to indicate the boundary of action sequences.

**Parameter settings:** The hyper-parameters of our model are set as follows: (1) the dimensionality of the representations, \( M \), is set to 500; (2) sampling window size, \( 2K + 1 \), is set to 11, i.e., \( K = 5 \). During training, we use 0.025 as the initial learning rate and subtract it by 0.005 for each subsequent epoch (5 epochs in total). For inference, we use 0.1 as the initial learning rate and subtract it by 0.02 for each subsequent epoch (5 epochs in total). Our parameter settings are consistent with the previous work on word2vec [21], although further tuning might yield better performance.
4.3 User profiling performance

We evaluate the profiling performance of the util2vec model with a user fingerprinting task. We start by holding out the 200 most recent sessions from Photoshop users who have at least 400 sessions in the first 6 months of 2015 (in total, 15,369 unique users are selected). We then train the util2vec model over the rest of the action sequences from 3 million Photoshop users. For each of 15,369 users, her action history $H_i$ has been divided into training sub-sequence, i.e., $H_i^{\text{train}} = H_i[-200:]$ and validation sub-sequence, i.e., $H_i^{\text{val}} = H_i[-200:]$, and an ideal model should be able to link $H_i^{\text{val}}$ with $H_i^{\text{train}}$ based on generated profiles. We infer the user’s representation based on the two subsequences respectively, i.e., infer $v_i^{\text{train}}$ from $H_i^{\text{train}}$ and $v_i^{\text{val}}$ from $H_i^{\text{val}}$. For each user $i$ and her profile $v_i^{\text{train}}$, we predict which validation subsequence belongs to her using cosine similarities. More specifically, we sort all the validation subsequences $H_j^{\text{val}}$ by the similarities between $v_i^{\text{val}}$ and $v_j^{\text{train}}$ in a descending order, and the ranking of the user’s real validation subsequence $H_i^{\text{val}}$ is denoted as rank$_{i}$. Finally, Mean Reciprocal Rank (MRR), as defined in eqn. 4, is used to evaluate the overall fingerprinting accuracy across $N$ users ($N = 15369$).

\[
\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i}
\]  

(4)

We compare util2vec to the bag-of-actions model, which counts the frequency with which each action occurred. As shown in Table 1 our framework outperforms the baselines by 31.72% even when tf-idf is leveraged to down-weight the frequent actions. The experimental results demonstrate that our model is able to produce user vectors that are more representative and have stronger discriminative power. Generally speaking, for a given user, our representation is able to discriminate against 31.72% more distractors, and in practice, software service providers can use such representations to better fingerprint each user.

Along with the user representations, util2vec also learns an action embedding $X$ that encodes semantic similarities between actions. For example, we present the nearest neighbors of five Photoshop actions in Table 2 (the neighbors are ranked by the cosine similarities between action embeddings in descending order), and the retrieval results show that the actions are grouped by their functionalities and usage affinities. The action embeddings may also be useful for the service improvements as it tells the common software usage patterns among the population. Given the scope of this paper, we leave further investigation as future work.

5. APPLICATIONS

In this section, we build and present three applications that can benefit from the integration of such usage traces: software user tagging, cold-start art project recommendation and inspiration engine.
average recall rate, Recall@K, as defined below, to quantitatively
compare the tagging performance.

\[
\text{Recall@}K = \frac{\text{number of correct tags in top } K \text{ predictions}}{\text{total number of tags in the ground truth set}}
\]

The results in Table 3 show that the models leveraging software usage history significantly outperform the baseline that is agnostic to such information, and the improvements are particularly remarkable for top-ranked tags—the system achieves 31.0% and 35.0% improvements in terms of Recall@1 and Recall@2 respectively. This justifies that, practically, our system can not only predict tags that are popular, but the ones that are long-tailed. Ultimately, our tagging system can make accurate predictions for millions of Photoshop users, who may or may not be active on Behance, and it is valuable to enable customized business for the service provider.

Qualitatively, we show the outputs of two tagging approaches for 6 representative Photoshop users in Fig. 4. For each user, we present the ground truth tags, the tags predicted by our system and the popular tags. In addition, we include user’s Behance portfolio (uploaded projects) side-by-side for the illustration purpose. But this information is not available to the tagging algorithm under any circumstance. From Fig. 4 we find that our tagging model is especially advantageous in the following aspects.

- **Tag diversity.** We can accurately predict a diverse array of areas of focus based on the Photoshop usage traces, e.g., *Photography* (U1, U4), *Fine Arts* (U1), *Web Design* (U5), *Typography* (U2), *Cartooning* (U3), *Animation* (U4) and *Painting* (U6) etc. The tags can be popular on the platform, e.g. *Graphic Design and Photography*, or long-tailed (infrequent), e.g. *Motion Graphics, Cartooning* and *Painting* etc. The prediction results justify the robustness of our system when it is applied to diverse application usage patterns.

- **Generalization power.** Although there is a high correlation between user tags and appearances of uploaded art projects, as shown in Fig. 4, some users didn’t exhaustively select all of the tags that are related. This limitation is partially addressed by the generalization power of our linear classifier. For example, based on U1’s portfolio (images with same content but different coloring), there is a high chance that she is focusing on *retouching*, which is not selected by herself. Nevertheless, our system can still make reasonable predictions that include *retouching* in the top tags. This characteristic is further verified in the U6 example (tag *Drawing*).

Overall, we have shown that by modeling application usage traces, we are able to build an accurate and practical user tagging system for Photoshop with minimal human effort.

### 5.2 Cold-start art project recommendation

Cold-start is a well-known hard problem in the design of modern recommender systems. Specifically, *user-cold-start* [14] refers to the scenario where the recommendations are targeting new users, and *item-cold-start* [12] describes the case when a new item needs to be included in the recommendation pool. In *user-cold-start*, since we lack the information of her activities within a platform, a typical solution is to either recommend the most popular items, which is not personalized, or leverage side information, such as gender, age [22] and personal data traces [14]. However, in many cases, when a new user shows up in online social platforms, their application usage records are already available. If we could leverage these data traces and properly use them to inform the recommender, there is a great potential for the social platforms to improve their cold-start recommendations. For example, Behance might be able to generate better recommendations for 3 million Photoshop users, which is almost 4 times the current number of Behance users. In this section, we propose a two-phase recommendation framework that leverages Photoshop usage data in recommending artistic projects on Behance.

#### 5.2.1 Two-phase recommendation framework

Our recommendation framework is inspired by previous research on content-based music recommendation [25], that incorporates au-

### Table 3: User tagging performance in terms of Recall@K. We use bold font for the best performed approach and feature set. The percentage of improvements are the comparison between util2vec (bolded), and popular tags. Our tagging system outperforms popularity tags baseline by 31.0% and 35.0% regarding Recall@1 and Recall@2 respectively.

<table>
<thead>
<tr>
<th>Recall@K</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>tagging with software usage data</td>
<td>util2vec features (500 dim)</td>
<td>0.2320</td>
<td>0.3569</td>
<td>0.4466</td>
<td>0.5140</td>
</tr>
<tr>
<td></td>
<td>bag-of-actions+tf-idf features (1990 dim)</td>
<td>0.2246</td>
<td>0.3489</td>
<td>0.4352</td>
<td>0.5017</td>
</tr>
<tr>
<td>tagging without software usage data</td>
<td>popular tags (baseline)</td>
<td>0.1771</td>
<td>0.2644</td>
<td>0.3644</td>
<td>0.4309</td>
</tr>
<tr>
<td>% of improvements</td>
<td>31.0%</td>
<td>35.0%</td>
<td>22.6%</td>
<td>19.3%</td>
<td>19.0%</td>
</tr>
</tbody>
</table>

---

3The tagging model is mainly designed to classify Photoshop users who do not have Behance profile.
Figure 4: Six user tagging examples with two different approaches. For each user, we show her portfolio, top 5 tag predictions with util2vec feature, top 5 most popular tags and self-disclosed tags. The tags with orange color are the correct predictions, and the ones with green color are the ones that are inferable from the portfolio but not explicitly selected by the user.

Figure 5: Two-phase recommendation framework. In step 1, we derive users’ latent factors and items’ latent factors and bias from their implicit feedback (project views). In step 2, we learn a projection function / to map software usage features to the corresponding users’ latent factors.

dio features in solving the item-cold-start problem. We take advantage of the opportunity that a portion of Photoshop users are already active on Behance and have left a significant amount of implicit feedbacks, e.g., project views. Therefore, to build the recommendation pipeline, we first learn users’ and items’ latent factors from their project views and then build a function to map the application usage features extracted from util2vec or bag-of-actions, to the latent factors. In production, for any Photoshop user, the system conducts cold-start recommendations by first predicting user’s latent factors based on her application usage data, and then ranking the items accordingly in the latent space. Formally speaking, we build the cold-start recommendation system with the following two steps (Fig. 5).

**Step 1.** The goal of the first step is to learn each user $u$’s latent factors $l_u$, and each item $e$’s latent factors $l_e$ and bias $b_e$, such that the value of $r_{ue}$, which is defined as $r_{ue} = l_u^T l_e + b_e$, is proportional to user $u$’s preference level towards item $e$. We learn the parameters by leveraging users’ project views on the platform. Considering that such signals are implicit feedback, as suggested by [27], we propose to minimize the following Weighted Approximately
Table 4: Art project recommendation performance for cold-start users in terms of Recall@K and Areas Under Curve (AUC). We use bold font for the best performed approach and feature set. The percentage of improvements are the comparison between the approach with bold font, and baseline method (popular items).

<table>
<thead>
<tr>
<th>Recall@K</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>cold-start recommendation with software usage data</td>
<td>util2vec features (500 dim)</td>
<td>0.0143</td>
<td>0.0213</td>
<td>0.0261</td>
<td>0.0313</td>
<td>0.0356</td>
</tr>
<tr>
<td>bag-of-actions+tf-idf features (1990 dim)</td>
<td>0.0138</td>
<td>0.0209</td>
<td><strong>0.0269</strong></td>
<td>0.0309</td>
<td>0.0350</td>
<td>0.8166</td>
</tr>
<tr>
<td>cold-start recommendation without software usage data</td>
<td>popular items (baseline)</td>
<td>0.0118</td>
<td>0.0188</td>
<td>0.0218</td>
<td>0.0281</td>
<td>0.0356</td>
</tr>
</tbody>
</table>

| % of improvements | 21.2% | 13.3% | 23.4% | 11.4% | 19.9% | 6.8% |

Figure 6: Four image retrieval results of the inspiration engine using single action. The retrieval results reflect the context where each action is often used. For example, with **fade_smart_blur**, returned images have blurred background and fading effects, and with **rotate_canvas**, images tend to have repetitive patterns.

Figure 7: The algorithm framework for the inspiration engine. We learn a function $g$ to project image features to the util2vec embedding space such that true actions-image pairs are close to each other and false pairs are farther away.

$\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$

util2vec embedding

$g$

Step 2. In the second step, we learn a projection function $f$ that takes a user’s software usage feature $\psi_u$ as input and produces output $f(\psi_u)$ to approximate her latent factors $\mathbf{l}_u$. We propose to minimize the $l_2$ loss $\sum ||f(\psi_u) - \mathbf{l}_u||^2$ for regression.

In this paper, we use a linear function $f$, i.e., $f(\psi_u) = \mathbf{b} + W_\psi \psi_u$. However, any non-linear function should be directly applicable here, which we leave to future work. During training, for step 1, the parameters are learned with mini-batch Adagrad [5], the dimensionality of the latent factors and learning rate are set to 50 and 0.05 respectively. For step 2, we use l-bfgs to find the optimal solution since the optimization target is convex.

In practice, for any cold-start user $u$ and her Photoshop usage feature $\psi_u$, the items’ recommendation rankings are based on the value of $r_{auc} = f(\psi_u) f^T \mathbf{l}_c + b_c$ where the item with higher value of $r_{auc}$ will be recommended earlier.

5.2.2 Evaluation and analysis

We evaluate the performance of our cold-start recommendation system by holding out a validation set from the view histories of 67,805 users. We randomly sample 10,000 users among the people who have viewed at least one project after July 1st, 2015 and regard them as the cold-start users. The most recent viewed project $c_{pu}$ from each cold-start user $u$ is then held for validation, and the rest 57,805 users’ complete view histories are used for training. All the items appear in the training set are included in the items pool, which yields 5.8 million candidates for recommendation. The time
rest of the software usage data is collected from the first 6 months of 2015. During the validation, for each cold-start user, we only use her software usage data to make the preference prediction, without relying on any previous views. Therefore, our evaluation results can properly reflect the system performance when serving cold-start users in practice.

We compare our recommender to the baseline algorithm that ranks the items based on their popularity (total number of views received). This is shown to be a very strong baseline for the cold-start recommendations [8]. Similar to user tagging, we use Recall@K defined in eqn. (8) and Area Under ROC Curve (AUC) defined in eqn. (9) to evaluate the recommendation performance (N=10,000).

\[
\text{Recall@K} = \frac{1}{U} \sum_{u=1}^{U} \delta(e_{pu} \text{ in the top } K \text{ items for user } u) 
\]

\[
\text{AUC} = \frac{1}{U} \sum_{u=1}^{U} \frac{\delta(f(v_u)^T I_{e_{pu}} + b_{e_{pu}} > f(v_u)^T I_e + b_e)}{\text{size of the items pool}}
\]

The experimental results shown in Table. 4 demonstrate that all of the recommenders that leverage the Photoshop usage traces and two-phase recommendation framework perform significantly better than the baseline in terms of Recall@K and AUC. For top-ranked items (Recall@100), in particular, our recommender outperforms the popularity based recommendation by 21.2%, which means that the users will potentially appreciate 21.2% more items among which we recommend. Also, the performance improvement suggests that we are able to personalize item recommendations to creative professionals who are new to the Behance platform.

5.3 Inspiration engine

In this section, through a sample application named inspiration engine, we demonstrate that the data integration can also enable innovative user experiences. The goal of inspiration engine is to provide real-time and personalized inspirations for creative professionals when they are working in Photoshop, and the system is able to show the potential outcomes of the actions that have been or are likely to be performed. Such presentations are inspiring because the artists can explore a wider range of possibilities that are related but different from their current work.

Technically speaking, the core component of such an application is a search engine that returns art projects likely to be produced by a given sequence of Photoshop actions. We build the system by leveraging the weak correspondence between the pairs of users’ Photoshop usage traces and the projects that they uploaded to Behance. With such pairs, we can learn a heterogeneous joint embedding where the true actions-image pairs are close to each other, and the false pairs are further away. As shown in Fig. 7 for each actions-image pair \((a_1, a_2, \ldots, a_n, e), i = 1, 2, \ldots, n\), we first extract features for the action sequence and the image respectively, denoted as \(v_i\) and \(z_i\). In our prototype, we extract \(v_i\) from \(util2vec\) and \(z_i\) from the pooling layer (2048 dim) of pre-trained ResNet [11], the state-of-the-art image feature extractor. Then we learn a function \(g\) to project image features \(z_i\) to the \(util2vec\) embedding space such that the objective \(\sum ||v_i - g(z_i)||^2\) is minimized.

To prototype the system, we train a linear projection function \(g\) with 353,205 actions-image pairs from 43,441 users and validate it over 20,000 held-out pairs from 20,000 users, i.e., each user contributes exactly one pair in the validation. There is no user overlap in the training and validation set, and the training is conducted using l-bfgs algorithm. Quantitatively, we use Recall@K and AUC as defined in Section 5.2 to evaluate the system performance, and the results are shown in Table. 5. The improvements over the random guess baseline justify that there is a close relationship between the Photoshop usage pattern and visual appearance of art project. In addition, in Fig. 8, we show four qualitative retrieval results of the inspiration engine (we only show retrieval with a single action, but our technique is applicable to action sequence as well). The nearest neighbors of each action reflect the scenarios where it is often used. For example, the action \(\text{drag_path}\) is heavily used in web design, and the \(\text{rotate_canvas}\) is typically leveraged to create repetitive patterns. We will conduct an end-to-end further user study in the future to evaluate the engine.

Through three applications, we observe that the amount of improvements brought by \(util2vec\), compared to the intuitive \(bag-of-actions+tfidf\) model, are contingent on the context of end applications. Nevertheless, the performance improvements are significant under most of the metrics except Recall@300 in the cold-start recommendation task, so we can safely conclude that \(util2vec\) is beneficial in modeling unstructured application usage traces, and we may get further gains in the future by tuning the model parameters and training methods.

6. CONCLUSIONS AND FUTURE WORK

In this work, we personalize software and web applications for creative professionals by leveraging Photoshop usage traces. These systems can enhance existing services provided by Photoshop (i.e., accurate prediction of users’ areas of focus, Section 5.1), and Behance (i.e., personalized recommendation for cold-start users, Section 5.2), and enable new experiences (i.e., inspiration engine, Section 5.3) for millions of users.

Although we focus on platforms for creative professionals, our results suggest that such an integration may be fruitful for personalization research more generally. For example, personalized applications can be built for programmers based on their Github usage records, and for journalists based on the usage of document editing tools. As people’s work and leisure lives are increasingly accompanied by applications, understanding and integrating digital breadcrumbs that they leave behind can lead to truly user-centric personalization.

7. ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by Adobe Research gift funding and The AOL-funded Connected Experiences Lab [http://cx.jacobs.cornell.edu]. The first author conducted part of this research at Adobe Research as a summer intern and is further supported by the small data lab at Cornell Tech, which receives funding from NSF, NIH, RWJF, MacArthur Foundation, Google, and UnitedHealth Group.
8. REFERENCES


