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.

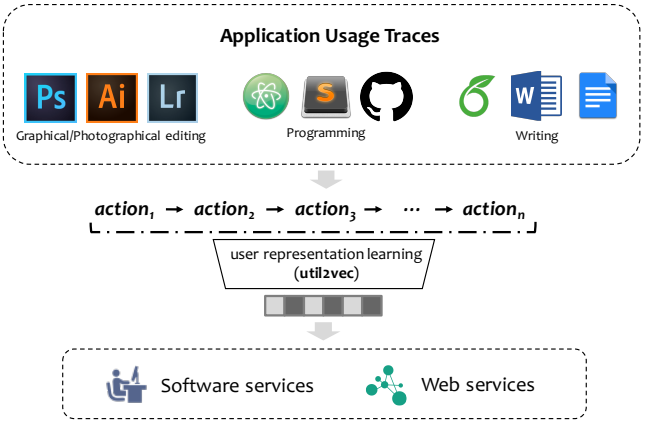


Figure 1: Services that can benefit from the integration of application usage traces. By leveraging user representations learned from sequences of actions, we explore how software service providers and social platforms can improve their personalized services and empower new user experiences.

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[1], 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.

- We develop and evaluate an approach,**util2vec**, based on distributed representation learning, that produces high-quality representations of Photoshop users. This model encodes the sequence patterns of the actions each user has performed, and significantly outperforms the *bag-of-actions* representation by 31.72% Mean Reciprocal Rank (MRR) on the *user fingerprinting* task (Section 4).
- Based on this model, we present three sample applications:
  1. We develop and evaluate a practical tagging system for Photoshop users. The system, for the first time, is able to accurately predict areas of focus for millions of Photoshop users, who may or may not be active on Behance. Our model significantly outperforms popularity-based tagging by 31% (Re-

[*]Work done while working at Adobe Research

[1]https://www.behance.net/

WWW'17 Companion, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4914-7/17/04.
http://dx.doi.org/10.1145/3041021.3054183

call@1), and is able to accurately predict long-tailed tags that are important but unpopular among the broader population (Section 5.1).

2. 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-occurences 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. 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 [8]. 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 concepts. Our **util2vec** framework is inspired by the previous research efforts mentioned above, and to the best of our knowledge, this is the first work to design a distributed representation learning algorithm in the domain of software user modeling.

2.2 Intra- and cross- platform user modeling in online social platforms

For online social platforms, personalization and user modeling are important tasks, since appropriately matching customers and products is a key to satisfactory user experiences [16]. Often, the goal of such modeling is to derive a real-valued vector for each user that summarizes his/her preferences, habits, and traits in online social platforms. Previous work constructs user vectors by leveraging intra-platform interactions [24], e.g., ratings [16, 3], purchases [13], content consumption [2, 25, 15], reviews [29], and social networks [10, 9], or cross-platform interactions, e.g., personal data streams across email, Twitter, and Facebook [14], and follower-followee connections across YouTube and Twitter [28]. Learned user representations have been shown to be effective in many application domains, such as movie [3], television [15], article [14, 26], e-commerce [13] and social network [9, 10, 2].

2.3 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.

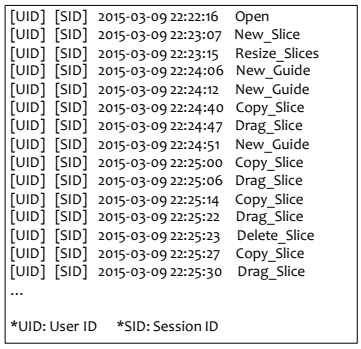
Another line of related work around application usage records attempts to understand the semantic meanings of software actions [1, 7]. By training a word2vec model [21] on online documents, previous work [1] discovered correspondences and relationships between natural language and software actions, which 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.

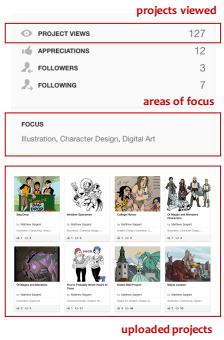
3. DATASET

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 million** 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-



(a) An example of action sequence in Photoshop



(b) An example of the collected user's social interactions on Behance

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*.

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 $\mathbf{H}_u = (a_1^u, a_2^u, ..., a_n^u)$ from a software user u, our goal is to learn a fixed-length real-valued vector $\boldsymbol{v}_u$ 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.

$$\frac{1}{T-2K}\sum_{u}\sum_{t=K}^{T-K}\log p(a_t^u|a_{t-K}^u,...,a_{t+K}^u \setminus a_t^u) \tag{1}$$

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 $p(a_t|a_{t-K},...,a_{t+K} \setminus a_t)$ as follows (We omit the superscripts of a_t^u where they are clear from context).

$$p(a_t|a_{t-K},...,a_{t+K} \setminus a_t) = \frac{e^{y_{a_t}}}{\sum_i e^{y_i}} \tag{2}$$

where the vector $\boldsymbol{y} = \boldsymbol{b} + Wh(u, a_{t-K},...,a_{t+K} \setminus a_t; V, X)$; the bias vector $\boldsymbol{b}$ 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., $\boldsymbol{v}_u = V[:,u]$ and $\boldsymbol{x}_i = X[:,i]$ in *numpy*-style notation. The parameters $\boldsymbol{b}, W, V$, 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 $2K$ context actions, as Fig. 3 shows.

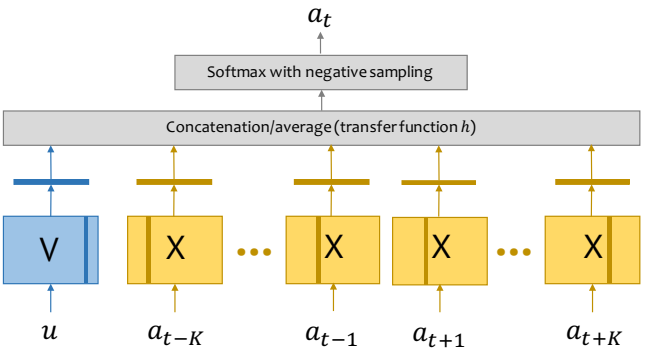


Figure 3: The architecture of util2vec model. The columns of V and X store the user representations and action representations respectively. While the action embedding X is shared across different users, user embedding V is user-specific.

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 u's representation $\boldsymbol{v}_u$ by fixing the parameters $\boldsymbol{b}, W, X$ and only fitting the vector $\boldsymbol{v}_u$ to user u'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 $\log p(a_t|a_{t-K},...,a_{t+K} \setminus a_t)$ as follows:

$$\log(\sigma(y_{a_t})) + \sum_{s\in\mathcal{S}}\log(\sigma(-y_s)) \tag{3}$$

where $\mathcal{S}$ is a set of randomly sampled actions such that $a_t \notin \mathcal{S}$, and σ 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 follow: (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.

Table 1: User fingerprinting (hold-out session retrieval) performance regarding Mean Reciprocal Rank (MRR). The improvement is respect to the *bag-of-actions+tf-idf*.

Modeling framework	MRR ($\pm$ standard error of mean)
util2vec	**0.8238$\pm$ 0.0029**
bag-of-actions + tf-idf (baseline)	0.6037$\pm$ 0.0037
bag-of-actions	0.5944$\pm$ 0.0037
% of improvement	**31.72%**

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 $\mathbf{H}_i$ has been divided into training sub-sequence, i.e., $\mathbf{H}_i^{\text{train}} = \mathbf{H}_i[:-200]$ and validation sub-sequence, i.e., $\mathbf{H}_i^{\text{val}} = \mathbf{H}_i[-200:]$, and an ideal model should be able to link $\mathbf{H}_i^{\text{val}}$ with $\mathbf{H}_i^{\text{train}}$ based on generated profiles. We infer the user's representation based on the two subsequences respectively, i.e., infer $\boldsymbol{v}_i^{\text{train}}$ from $\mathbf{H}_i^{\text{train}}$ and $\boldsymbol{v}_i^{\text{val}}$ from $\mathbf{H}_i^{\text{val}}$. For each user i and her profile $\boldsymbol{v}_i^{\text{train}}$, we predict which validation subsequence belongs to her using cosine similarities. More specifically, we sort all the validation subsequences $\mathbf{H}_j^{\text{val}}$ by the similarities between $\boldsymbol{v}_j^{\text{val}}$ and $\boldsymbol{v}_i^{\text{train}}$ in a descending order, and the ranking of the user's real validation subsequence $\mathbf{H}_i^{\text{val}}$ 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 ($N = 15369$).

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i} \tag{4}$$

We compare **util2vec** to the *bag-of-actions* model, which counts the frequency with which each action ocurred. 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*.

5.1 Software user tagging

User tagging is an important task for software service providers, as accurate tags are fundamental to effective business and ads targeting, personalization and recommendation. Essentially, the goal of user tagging is to assign a set of relevant tags to users based on her behavior in the platform. For Photoshop, in particular, the tagging task is to predict users' areas of focus based on the software usage patterns. For example, an ideal tagging system should be able to predict whether an user is focusing on *web design*, *UI/UX* or *architecture* based on the tools that she uses. Traditionally, building such a user tagging system requires a significant amount of domain knowledge and human labor to bootstrap the training labels. The expert software developers need to manually examine the raw usage histories and come up with the tags for a large number users. As imagined, such a human labeling process subjects to diverse human expertise in recognizing the patterns and is inevitably error-prone and incomprehensive.

We build an accurate Photoshop user tagging system with minimal human efforts by leveraging interactions on Behance. It is based on the observation that nowadays, people self-disclose their expertise and areas of focus in many social platforms for socializing and job hunting. By leveraging the self-disclosed tags from Behance and the accurate user representations derived from **util2vec**, we are able to build a user tagging system that is more accurate and robust than other approachs.

5.1.1 User tagging model

Formally speaking, given U users, along with their self-disclosed tags from Behance, $\boldsymbol{t}_u$ and representations $\boldsymbol{v}_u$ derived from Photoshop usage traces ($\boldsymbol{t}_u$ is a D-dimensional one-hot encoded vector, where D is the size of the tag set), we learn a user tagging model f that takes $\boldsymbol{v}_u$ as input and produces an output to approximate $\boldsymbol{t}_u$. As suggested in the image tagging tasks, we train the user tagging model by minimizing the following sigmoid cross-entropy loss:

$$-\frac{1}{U}\sum_u \boldsymbol{t}_u \log(\sigma(f(\boldsymbol{v}_u))) + (1-\boldsymbol{t}_u)\log(1-\sigma(f(\boldsymbol{v}_u))), \tag{5}$$

where the value of the j-th element in $\boldsymbol{t}_u$ is 1 if the j-th tag is selected by the user u, 0 otherwise. Theoretically, model f can be any linear or non-linear function. In this paper, we use the linear projection, i.e., $f(\boldsymbol{v}_u) = \boldsymbol{b} + W\boldsymbol{v}_u$, though adding non-linear components such as multi-layer perceptron (a.k.a. deep neural networks) might potentially improve the performance.

We train the model using limited-memory BFGS (l-bfgs) [19] over the areas that are indicated by at least 100 active users on Behance. With such a filtering criteria, we finally keep 67 labels in the tag pool, which includes, to name but a few, *Graphic Design*, *Motion Graphics*, *Character Design*, *Cinematography*, *Icon Design* and *Computer Animation*. Then for any Photoshop user (without any requirement to be on Behance), we can assign tags by running the classifier over her application usage history.

5.1.2 Evaluation and analysis

To demonstrate the effectiveness of our tagging system, we conduct an evaluation against 65,331 users who have labeled themselves with at least one of the 67 tags. In the final dataset, each user is associated with 1 to 5 tags. We randomly divide the users into a training set and a validation set, which consists of 45,331 and 20,000 samples respectively. The baseline approach that we compare to ranks the tags purely based on their number of appearances on the Behance platform. While simple, such a comparison directly reflects the feasibility and reliability of the tagging system, and it is the best we can achieve without our tagging model. We use the

Table 2: 5-nearest neighbors of the selected actions in action embedding X. The actions in the first row are the index, and the five actions below are the corresponding nearest neighbors. (From left to right, the actions are related to *video editing*, *font awesome icons*, *blur filters*, *path manipulations* and *shadow effects*, respectively.)

modify_video_clip	fa_times	delete_smart_filter_blur	drag_path	inner_shadow
modify_video_clip_audio	fa_user	edit_filter_effect_blur	duplicate_paths	inner_glow
set_work_area_start	fa_bars	edit_filter_blending_options_blur	nudge_paths	bevel_emboss
add_audio_clips	fa_home	delete_smart_filter_blur_more	scale_paths	gradient_overlay
mute_audio_track	fa_map_marker	delete_smart_filter_gaussian_blur	drag_anchor_points	clear_effects
modify_audio_clip	fa_plus	enable_filter_effect_blur	distribute_horizontal_centers	drop_shadow

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.

Recall@K		1	2	3	4	5
tagging with software usage data	util2vec features (500 dim)	**0.2320**	**0.3569**	**0.4466**	**0.5140**	**0.5691**
	bag-of-actions+tf-idf features (1990 dim)	0.2246	0.3489	0.4352	0.5017	0.5559
tagging without software usage data	popular tags (baseline)	0.1771	0.2644	0.3644	0.4309	0.4781
% of improvements		**31.0%**	**35.0%**	**22.6%**	**19.3%**	**19.0%**

average recall rate, Recall@K, as defined below, to quantitatively compare the tagging performance.

$$\text{Recall@}K = \frac{\text{number of correct tags in top } K \text{ predictions}}{\text{total number of tags in the ground truth set}} \quad (6)$$

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[2]. From Fig. 4, we find that our tagging model is especially advantagous 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 apperances 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*).

[2]The tagging model is mainly designed to classify Photoshop users who do not have Behance profile.

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 recommeder, 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-

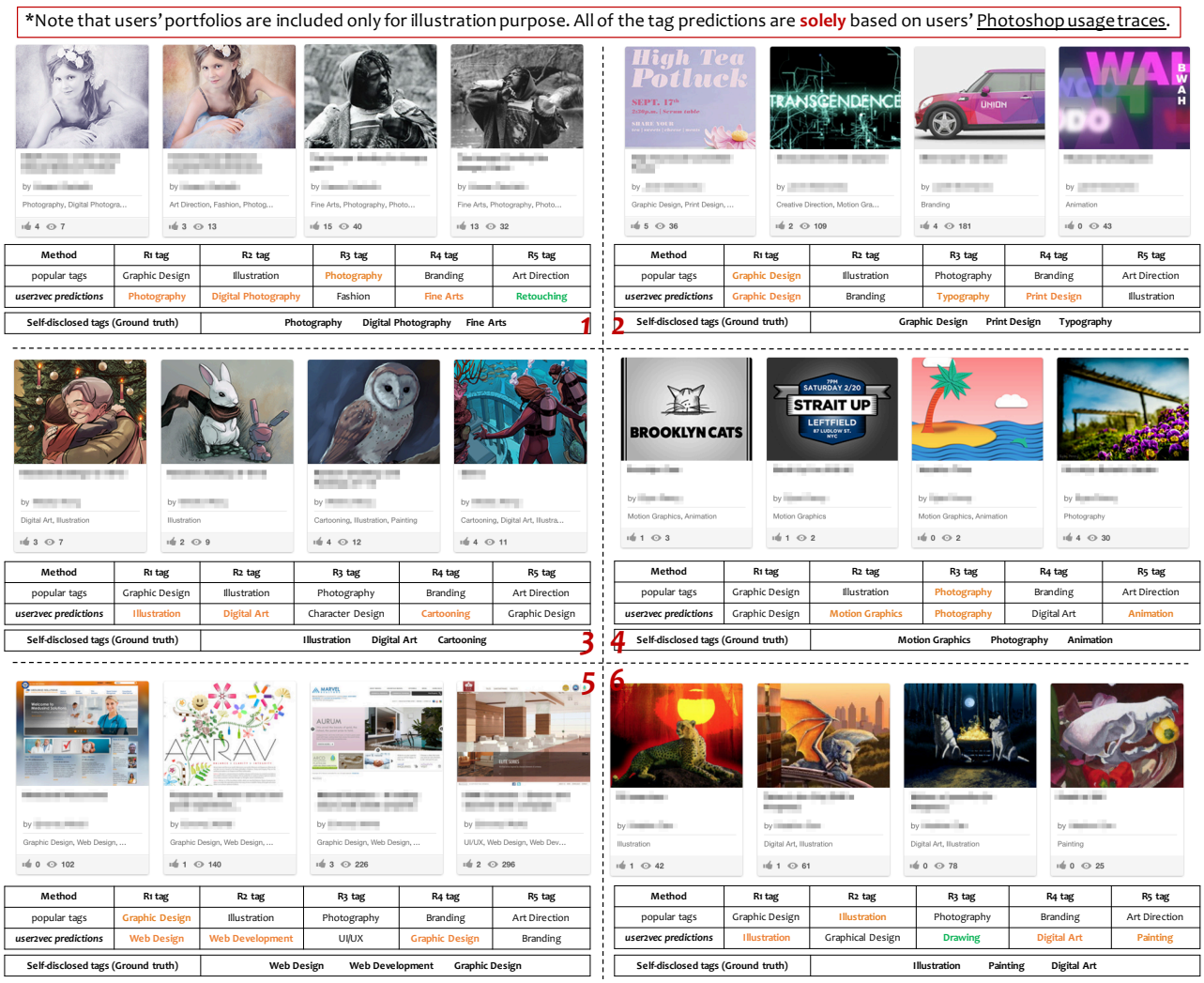


Figure 4: Six user tagging examples with two different approaches. For each user, we show her portfolio, top 5 tag predictions with util2vec feature, top 5 most popular tags and self-disclosed tags. The tags with orange color are the correct predictions, and the ones with green color are the ones that are inferrable from the portfolio but not explicitly selected by the user.

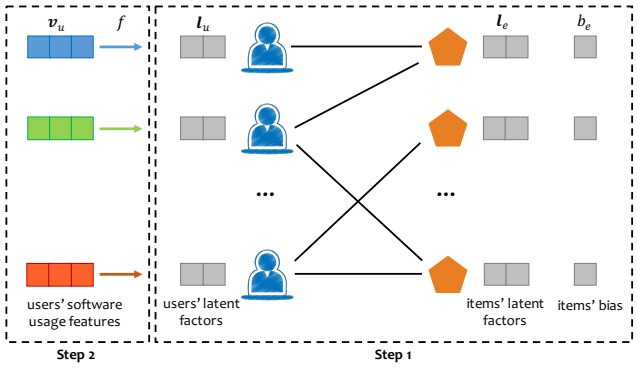


Figure 5: Two-phase recommendation framework. In step 1, we derive users' latent factors and items' latent factors and bias from their implicit feedback (project views). In step 2, we learn a projection function f to map software usage features to the corresponding users' latent factors.

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 $\boldsymbol{l}_u$, and each item e's latent factors $\boldsymbol{l}_e$ and bias b_e, such that the value of r_{ue}, which is defined as $r_{ue} = \boldsymbol{l}_u^T \boldsymbol{l}_e + b_e$, 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).

Recall@K		100	200	300	400	500	AUC
cold-start recommendation with software usage data	util2vec features (500 dim)	**0.0143**	**0.0213**	0.0261	**0.0313**	**0.0356**	**0.8202**
	bag-of-actions+tf-idf features (1990 dim)	0.0138	0.0209	**0.0269**	0.0309	0.0350	0.8166
cold-start recommendation without software usage data	popular items (baseline)	0.0118	0.0188	0.0218	0.0281	0.0297	0.7683
% of improvements		**21.2%**	**13.3%**	**23.4%**	**11.4%**	**19.9%**	**6.8%**

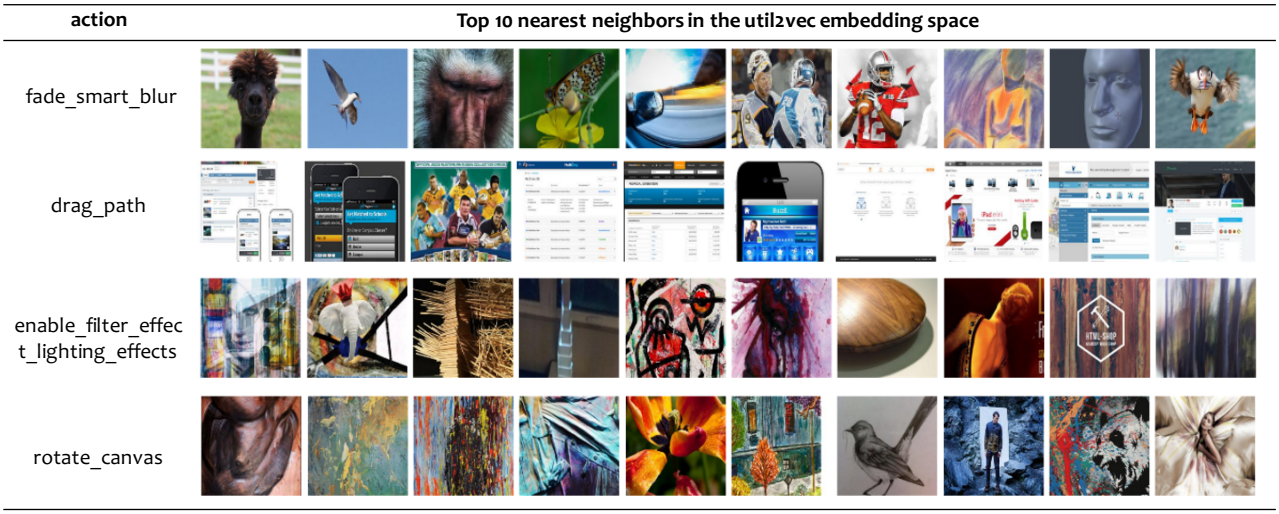


Figure 6: Four image retrieval results of the inspiration engine using single action. The retrieval results reflect the context where each action is often used. For example, with *fade_smart_blur*, returned images have blurred background and fading effects, and with *rotate_canvas*, images tend to have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

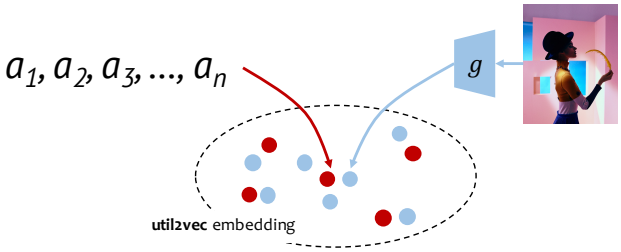


Figure 7: The algorithm framework for the inspiration engine. We learn a function g to project image features to the util2vec embedding space such that true actions-image pairs are close to each other and false pairs are father away.

Ranked Pairwise (WARP) loss:

$$\sum_{u,e\in P_u,e'\in S\setminus P_u} \ln\left(\frac{Y}{M_{e'}}\right)\left|1-\boldsymbol{l}_u^T\boldsymbol{l}_e-b_e+\boldsymbol{l}_u^T\boldsymbol{l}_{e'}+b_{e'}\right|_+, \qquad (7)$$

where S denotes the set of all items, P_u denotes the set of items viewed by user u, Y denotes the total number of items, and $M_{e'}$ denotes the number of negative sampling conducted before encountering an item e' that produces non-zero loss. In other words, during training, for each item that the user viewed, we keep sampling negative items e' until $1+\boldsymbol{l}_u^T\boldsymbol{l}_{e'}+b_{e'}>\boldsymbol{l}_u^T\boldsymbol{l}_e+b_e$ is satisfied.

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

In this paper, we use a linear function f, i.e., $f(\boldsymbol{v}_u)=\boldsymbol{b}+W\boldsymbol{v}_u$. 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 $\boldsymbol{v}_u$, the items' recommendation rankings are based on the value of $r_{ue}=f(\boldsymbol{v}_u)^T\boldsymbol{l}_e+b_e$ where the item with higher value of r_{ue} 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 e_{p_u} 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

restriction is used to guarantee the causality of recommendation as the Photoshop 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 [14]. 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).

$$\text{Recall@}K = \frac{1}{U}\sum_{u=1}^{U}\delta(e_{p_u} \text{ in the top } K \text{ items for } u) \quad (8)$$

$$\text{AUC} = \frac{1}{U}\sum_{u=1}^{U}\frac{\sum_{e'}\delta(f(\boldsymbol{v}_u)^T\boldsymbol{l}_{e_{p_u}} + b_{e_{p_u}} > f(\boldsymbol{v}_u)^T\boldsymbol{l}_{e'} + b_{e'})}{\text{size of the items pool}} \quad (9)$$

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 $((a_1^i, a_2^i, ..., a_n^i), c_i), i = 1, 2, ..., n$, we first extract features for the action sequence and the image respectively, denoted as $\boldsymbol{v}_i$ and $\boldsymbol{z}_i$. In our prototype, we extract $\boldsymbol{v}_i$ from **util2vec** and $\boldsymbol{z}_i$ 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 $\boldsymbol{z}_i$ to the **util2vec** embedding space such that the objective $\sum_i \|\boldsymbol{v}_i - g(\boldsymbol{z}_i)\|^2$ 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. 6, 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 *drag_path* is heavily used in web design, and the *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.

Table 5: Action-image retrieval performance in terms of Recall@K. We use bold font for the best performed approach.

Recall@K	100	300	500	AUC
inspiration engine (util2vec features)	**0.0244**	**0.0603**	**0.0884**	**0.6646**
inspiration engine (bag-of-actions+tfidf)	0.0181	0.0488	0.0741	0.6357
random guess	0.005	0.015	0.025	0.5

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

[1] E. Adar, M. Dontcheva, and G. Laput. Commandspace: modeling the relationships between tasks, descriptions and features. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 167–176. ACM, 2014.

[2] D. Agarwal, B.-C. Chen, Q. He, Z. Hua, G. Lebanon, Y. Ma, P. Shivaswamy, H.-P. Tseng, J. Yang, and L. Zhang. Personalizing linkedin feed. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1651–1660. ACM, 2015.

[3] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[4] E. Choi, M. T. Bahadori, E. Searles, C. Coffey, and J. Sun. Multi-layer representation learning for medical concepts. *arXiv preprint arXiv:1602.05568*, 2016.

[5] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[6] M. Ekstrand, W. Li, T. Grossman, J. Matejka, and G. Fitzmaurice. Searching for software learning resources using application context. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 195–204. ACM, 2011.

[7] C. A. Fraser, M. Dontcheva, H. Winnemoeller, and S. Klemmer. Discoveryspace: Crowdsourced suggestions onboard novices in complex software. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, pages 29–32. ACM, 2016.

[8] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1809–1818. ACM, 2015.

[9] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *Proceedings of the third ACM conference on Recommender systems*, pages 53–60. ACM, 2009.

[10] I. Guy, N. Zwerdling, I. Ronen, D. Carmel, and E. Uziel. Social media recommendation based on people and tags. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 194–201. ACM, 2010.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[12] R. He and J. McAuley. Vbpr: visual bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1510.01784*, 2015.

[13] R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 507–517. International World Wide Web Conferences Steering Committee, 2016.

[14] C.-K. Hsieh, L. Yang, H. Wei, M. Naaman, and D. Estrin. Immersive recommendation: News and event recommendations using personal digital traces. In *Proceedings of the 25th International Conference on World Wide Web*, pages 51–62. International World Wide Web Conferences Steering Committee, 2016.

[15] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[16] Y. Koren, R. Bell, C. Volinsky, et al. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[17] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.

[18] W. Li, J. Matejka, T. Grossman, J. A. Konstan, and G. Fitzmaurice. Design and evaluation of a command recommendation system for software applications. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 18(2):6, 2011.

[19] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[20] J. Matejka, W. Li, T. Grossman, and G. Fitzmaurice. Communitycommands: command recommendations for software applications. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 193–202. ACM, 2009.

[21] T. Mikolov and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013.

[22] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pages 21–28. ACM, 2009.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[24] L. Tang, B.-C. Chen, D. Agarwal, and B. Long. An empirical study on recommendation with multiple types of feedback. In *Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016.

[25] A. Van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651, 2013.

[26] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM, 2011.

[27] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. 2011.

[28] M. Yan, J. Sang, and C. Xu. Mining cross-network association for youtube video promotion. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 557–566. ACM, 2014.

[29] F. Zhang, N. J. Yuan, K. Zheng, D. Lian, X. Xie, and Y. Rui. Exploiting dining preference for restaurant recommendation. In *Proceedings of the 25th International Conference on World Wide Web*, pages 725–735. International World Wide Web Conferences Steering Committee, 2016.