Characterizing User Skills from Application Usage Traces with Hierarchical Attention Recurrent Networks

LONGQI YANG, Cornell Tech, Cornell University
CHEN FANG and HAILIN JIN, Adobe Research
MATTHEW D. HOFFMAN*, Google
DEBORAH ESTRIN, Cornell Tech, Cornell University

Predicting users' proficiencies is a critical component of AI-powered personal assistants. This article introduces a novel approach for the prediction based on users' diverse, noisy, and passively generated application usage histories. We propose a novel bi-directional recurrent neural network with hierarchical attention mechanism to extract sequential patterns and distinguish informative traces from noise. Our model is able to attend to the most discriminative actions and sessions to make more accurate and directly interpretable predictions while requiring 50× less training data than the state-of-the-art sequential learning approach. We evaluate our model with two large scale datasets collected from 68K Photoshop users: a digital design skill dataset where the user skill is determined by the quality of the end products and a software skill dataset where users self-disclose their software usage skill levels. The empirical results demonstrate our model's superior performance compared to existing user representation learning techniques that leverage action frequencies and sequential patterns. In addition, we qualitatively illustrate the model's significant interpretative power. The proposed approach is broadly relevant to applications that generate user time-series analytics.

CCS Concepts: • **Information systems** → **Personalization**;

Additional Key Words and Phrases: User skill, recurrent neural network, hierarchical attention, user modeling

ACM Reference format:
Longqi Yang, Chen Fang, Hailin Jin, Matthew D. Hoffman, and Deborah Estrin. 2018. Characterizing User Skills from Application Usage Traces with Hierarchical Attention Recurrent Networks. *ACM Trans. Intell. Syst. Technol.* 9, 6, Article 68 (October 2018), 18 pages.
https://doi.org/10.1145/3232231

1 INTRODUCTION

People increasingly rely on professional applications to accomplish complex tasks at work. For example, artists use Photoshop and Illustrator for creative artwork, and engineers design 3D models with Autodesk. The user's proficiency in manipulating these professional tools has become an

*Work done while working at Adobe Research.

This work is supported by the Adobe Research gift funding and AOL-Program for Connected Experiences, and is further supported by the small data lab at Cornell Tech, which receives funding from UnitedHealth Group, Google, Pfizer, RWJF, NIH and NSF.

Authors' addresses: L. Yang and D. Estrin, 2 W Loop Rd, New York, NY 10044, Cornell Tech, Cornell University; emails: ylongqi@cs.cornell.edu, destrin@cornell.edu; C. Fang and H. Jin, 345 Park Avenue, San Jose, CA 95110, Adobe Research; emails: {cfang, hljin}@adobe.com; M. D. Hoffman, 345 Spear St, San Francisco, CA 94105, Google; email: matt@matthewdhoffman.com.

https://doi.org/10.1145/3232231

important component in application personalization (e.g., customization of the user interface [34], personal assistants [17]), tutorial recommendation [35], and talent search [2]. Therefore, developing effective models and systems that predict users' skill levels has captured much attention from industry and academia [11, 12, 36].

In this article, we leverage passively generated application usage traces for skill level prediction, which is far more scalable and immersive than traditional user testing and surveying. We utilize traces that contain a series of actions or commands[1] performed by users in the application, e.g., *open*, *gaussian filter,* and *crop* in Photoshop. These traces are routinely recorded to generate application usage statistics.

Previous user representation learning techniques commonly model users with bag-of-action features, i.e., the frequency of each action performed in the history [11, 28]. This approach captures the associations between users' skill levels and their action choices (e.g., the beginners may solely perform basic actions while experts tend to use advanced features) but has limitations in its prediction of expertise:

(1) *Sequential patterns.* Apart from the frequency of each action performed, how the actions are sequentially combined to accomplish tasks is relevant to distinguish experts from general users. For example, the trial-and-error of beginners may yield many repetitive sequences in his or her application usage history. It is difficult for the bag-of-actions approach to capture such patterns because it collapses the sequence data into unigram collections.
(2) *Identifying informative actions and sessions.*[2] Intuitively, some common sessions may not be discriminative in predicting users' skill levels. For example, a session used to crop an image, such as [open]-[crop]-[save], is pervasive for all users. In addition, for a given session, some actions, such as [open], are not indicative of the user's proficiency. As bag-of-actions does not differentiate informative actions and sessions from uninformative ones, the predictions are sub-optimal.

Recently, Yang et al. [43] proposed a util2vec framework based on word2vec [33] to learn a software user representation from the sequential action history, and demonstrated its power in predicting users' preferences and areas of focus. Nevertheless, although this approach takes into consideration sequential patterns, it overlooks the discriminative power differences of action sequences (i.e., some sub-sequences may be more informative than others), which makes it suboptimal for skill predictions (see Session 2.4 for algorithm details and Sections 5 and 6 for experimental results).

In this article, we propose a novel hierarchical attention mechanism with a bi-directional recurrent neural network (h-ATT-BiRNN) (Section 3) to tackle the limitations mentioned above. Our model takes as input a list of sessions from a user and produces a score proportional to the user's level of expertness. It conducts a soft-search for the most informative actions in the user's usage history using our attention modules and BiRNN and then makes the predictions accordingly. Compared to previous approaches, our model advances the state-of-the-art in three ways:

(1) h-ATT-BiRNN makes more accurate predictions by leveraging an action-level and a session-level attention mechanism to search for, and attend to, the most informative sequences.

[1]We use the terms "actions" and "commands" interchangeably. Each action is usually a valid click in the application.
[2]In software applications, a session contains a sequence of actions performed from the launch of the application to the point when it is closed.

(2) The predictions from h-ATT-BiRNN are directly interpretable. We are able to discover sequential patterns that are most predictive of the user's skill level, and the patterns cannot be captured by the prior art [11, 28, 43]. Such interpretations are useful for service providers to optimize their services and for the users to improve their skills.
(3) Compared to the previous util2vec model that needs to be trained on traces from 3 million users [43], h-ATT-BiRNN requires 50× less training data (less than 60k) and achieves significantly better performance (Sections 5 and 6). Therefore, it is much easier for the service providers to adapt and update.

We evaluate h-ATT-BiRNN model and baselines on two large-scale datasets that are passively and actively collected from 68,456 Photoshop users, respectively. They reflect different dimensions of users' skill levels.

(1) *Digital design skill dataset.* Models are trained to rank [30] the quality of the artwork created by different users, which is a direct indicator of the user's digital design skill level, defined as the abilities to produce high-quality digital media outcomes.
(2) *Software skill dataset.* We use this dataset to evaluate models' performances in predicting users' software skill levels, i.e., proficiencies in manipulating professional tools. We train regression models to predict skill levels ranging from novice to expert.

The experimental results demonstrate that h-ATT-BiRNN significantly outperforms the existing user modeling approaches in terms of area under curve (AUC) and mean square error (MSE) in both datasets, and it provides sequential pattern interpretations for the prediction. This work represents an early step in advancing deep learning techniques for the mining and modeling of users' action sequences. Given the increasing pervasiveness of user time-series analytics, the model proposed in this article has broad relevance. Our code can be found at https://github.com/ylongqi/characterizing-user-skills.

2 RELATED WORK

Our work benefits from previous research on user skill modeling in related domains, software user modeling, recurrent neural networks (RNNs), and attention mechanisms.

2.1 User Skill Modeling

User knowledge level and skill modeling has been extensively studied in the domain of web search [5–7, 21, 22, 29, 31, 41, 46–48], recommendation [32], social media [49], and gaming [18, 19]. Previous work designed domain-specific features to model and capture user skills. For example, in web search, features were hand-crafted for documents [6, 7, 47], queries [21, 22, 41, 47], and sessions [7, 41]; in gaming, previous work analyzed the intensity and break [18, 19]; and in recommendation, McAuley et al. [32] leveraged a latent model to predict users' experiences.

Compared to previous work on user skill modeling, our research concerns the domain of professional software, which is less-explored and has much richer functionality than search, gaming, and recommendation. People accomplish a wide range of complex tasks using such software. Therefore, hand-crafting features to characterize user skills becomes practically infeasible, and this work automatically learns feature representations through end supervision.

2.2 Sequential User Behavior Modeling

Modeling users' sequential behavior has been recently studied in the context of recommendation [14, 15, 42]. Previous work demonstrated the power of embedding methods [14] and recurrent neural networks [42] in encoding sequential patterns, which inspired us to leverage similar structures in the skill prediction.

The main goal of recommendation is to learn users' preferences and rank items accordingly, which is fundamentally different from skill prediction where the focus is to rank time-series usage traces. Therefore, our work complements previous research in terms of developing sequential models to characterize users' other important aspects.

2.3 Software User Modeling

Learning representations of software users has captured much attention from many research communities [9, 11, 24, 28, 43]. Such representations have been shown to inform effective skill level predictors [11], command recommenders [9, 24, 28], creative content recommenders [43] and user tagging systems [43]. The framework of predicting users' skill levels using interaction records is also widely studied in other domains, such as crowdsourcing [36] and education [12]. Previous work used a bag-of-actions approach to represent each software user [11, 28], which overlooks the sequential patterns of the action sequences. To address this issue, Yang et al. [43] proposed the util2vec framework to learn user representations in a distributed manner. A major limitation of this framework in skill level modeling is that it fails to consider the condition where some sessions or actions may not be predictive of the user's skill level, and assigning equal weights to all data points may deprive the predictor of making effective judgements based on noisy data streams. As the strongest baseline that we compare to in the experiments, the util2vec framework is briefly reviewed in the next subsection (Section 2.4). Compared to the previous work in this domain, our proposed model jointly learns to encode the sequential patterns of action histories and to attend to subsequences that are predictive of the user's skill level. The empirical results demonstrate that our model achieves better ranking and prediction performance and builds connections between sequences and the predictive results.

2.4 util2vec Framework

The util2vec framework [43] jointly learns user representations V and action representations X across a large number of users' software usage histories (e.g., 3 million users are used in Reference [43]), where a row in each of the matrices V and X store the representation for a user and an action, respectively. The representations are trained to maximize the log probability of the correct action predictions (Equation (1)) across users, i.e., $\sum_u \mathcal{L}_u$, with the $\mathcal{L}_u$ defined as below:

$$\mathcal{L}_u = \frac{1}{N - 2K} \sum_{t=K}^{T-K} \log p(a_t | a_{t-K}, \ldots, a_{t+K} \setminus a_t), \tag{1}$$

where $[a_1, \ldots, a_N]$ denotes the user's action history, and K denotes the farthest action before/from the context. In the optimization function $\mathcal{L}_u$, the conditional probability p is defined as in Equation (2):

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

where $\boldsymbol{y} = \boldsymbol{b} + Wh(u, a_{t-K}, \ldots, a_{t+K} \setminus a_t; V, X)$. As suggested by Refernce [43], util2vec leverages a transfer function that averages a user representation with the representations from $2K$ actions. After such a model is trained, for each new user u, the user representation $\boldsymbol{v}_u$ is obtained by fixing $\boldsymbol{b}, W, X$ and only fitting the user vector $\boldsymbol{v}_u$ to the user u's action history.

2.5 Recurrent Neural Network

Recurrent neural networks (RNNs) have been shown to be superior in modeling sequential data traces such as natural language [3] and speech [13]. It successfully enables a wide range of real

world applications, e.g., machine translation [3, 39], speech recognition [13], and image captioning [40]. Among a number of RNN structures, long short term memory (LSTM) [16] is widely adopted because of its ability to encode long sequences. In this article, we also build our model based on the structure of LSTM. To the best of our knowledge, this is the first work that advances deep sequential models in modeling software users.

2.6 Attention Mechanism

Our idea of hierarchical attention mechanism is inspired by previous work on the alignment of the encoder and the decoder in neural machine translation (NMT) [3]. The basic idea is to allow the decoder to soft-search the most relevant words in the source language to make more confident and accurate translations. Our work is focusing on a different domain where there is no explicit encoder-decoder structure [39], and the model needs to handle hierarchical aggregation both on the action-level and session-level. Fundamentally, the attention module in NMT is driven by the current translation status, while our attention module is guided by a single prediction target. Therefore, our attention modules are suited for static classification and regression tasks while the attention mechanism in NMT is specifically designed for sequence generation. Recently, the hierarchical attention network was successfully used in document classification [45]. Our model is similar to the structure proposed by Yang et al. [45] but focuses on a significantly different and important application scenarios.

3 H-ATT-BIRNN: HIERARCHICAL ATTENTION

In this section, we describe the proposed h-ATT-BiRNN in the single-user case. A model that predicts skill levels takes as input a list of sessions $i = 1, 2, \ldots, L$ from a user's usage history, and each session i contains a list of actions that the user performed, denoted as $[a_1^i, a_2^i, \ldots, a_{T_i}^i]$ (As the length of each session may be different, we use T_i to represent the number of actions in session i). The goal of such a model is to predict a score k that is proportional to the user's skill level.

We build the h-ATT-BiRNN model as Figure 1 shows. First, we represent each action a_j^i using a vector $\boldsymbol{x}_j^i$, which is derived from an *embedding module* e (Equation (3)):

$$\boldsymbol{x}_j^i = e(a_j^i). \tag{3}$$

In our model, module e is a simple dictionary look-up that takes a_j^i as the index and returns the a_j^ith row of an $N \times M$ matrix X, where N denotes the number of unique actions and M denotes the dimensionality of $\boldsymbol{x}_j^i$.

We then design a non-linear *action encoder* p to derive a representation $\boldsymbol{s}_i$ for each session i from a list of action representations, i.e., $[\boldsymbol{x}_1^i, \boldsymbol{x}_2^i, \ldots, \boldsymbol{x}_{T_i}^i]$, as Equation (4) shows:

$$\boldsymbol{s}_i = p([\boldsymbol{x}_1^i, \boldsymbol{x}_2^i, \ldots, \boldsymbol{x}_{T_i}^i]). \tag{4}$$

To capture the sequential patterns and attend to the most discriminative actions, we design an action-level attention mechanism along with a bi-directional RNN to conduct the representation learning. We describe the details of the action encoder in Section 3.1.

Afterward, with L session representations, we further derive the user representation $\boldsymbol{v}$ using a non-linear *session encoder* q, as Equation (5) shows:

$$\boldsymbol{v} = q([\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_L]). \tag{5}$$

Our session encoder contains a session-level attention module and is able to discover sessions that are most predictive of the user's skill level. The details of the session encoder is presented in Section 3.2.

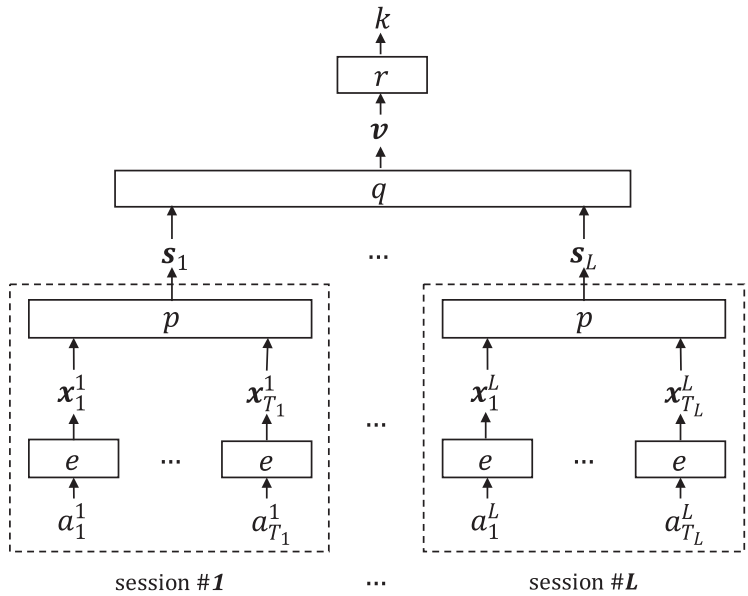

Fig. 1. The high-level structure of the h-ATT-BiRNN model. The rectangles are used to denote modules, each of which will be discussed in detail in the article. The h-ATT-BiRNN model takes a list of sessions $i \in \{1, \ldots, L\}$ as inputs and outputs a score k, representing a user's skill level. Each session i contains a list of actions $a_1^i, \ldots, a_{T_i}^i$. Figure notations: e: action embedding module; p: action encoder; q: session encoder; r: regression function; $\boldsymbol{x}_{T_i}^j$: representation for action $a_{T_i}^j$; $\boldsymbol{s}_i$: representation for session i; $\boldsymbol{v}$: user representation.

Finally, the prediction is made based on a regression function r. In our model, we use a linear regression, along with a rectified linear unit (ReLU), as shown in Equation (6), to make the prediction, because we believe that the non-linear action encoder p and session encoder q already provide enough learning capacity:

$$k = r(\boldsymbol{v}) = W_r \max(\boldsymbol{v}, 0) + b_r. \tag{6}$$

In the rest of this section, we present the design of the action encoder p (Section 3.1) and the session encoder q (Section 3.2).

3.1 Action Encoder p

The action encoder reads a sequence of action vectors $[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T]$ and then produces a session representation $\boldsymbol{s}$. (We omit the superscripts of $\boldsymbol{x}_j^i$ and the subscript of $\boldsymbol{s}_i$ where they are clear from context). As is illustrated in Figure 2, we use an RNN [16] as the building block for the encoder because of its superior performance in modeling sequential traces. With the basic RNN structure, a non-linear function g is used to read an input vector $\boldsymbol{x}_j$ for each step j and generate a memory state $\boldsymbol{c}_j$ and a hidden state $\boldsymbol{h}_j$, which are then fed into the same function g until reaching the end of the sequence (Equation (7)), and the hidden state $\boldsymbol{h}_j$ is then treated as the contextual representation of each position j:

$$\boldsymbol{c}_j, \boldsymbol{h}_j = g(\boldsymbol{c}_{j-1}, \boldsymbol{h}_{j-1}, \boldsymbol{x}_j). \tag{7}$$

However, the representation $\boldsymbol{h}_j$ derived from traditional RNN does not encode information from subsequent or future steps. To achieve more comprehensive contextual representations, we add backward connections to traditional RNN, i.e., we use bi-directional RNN [37] in our model.

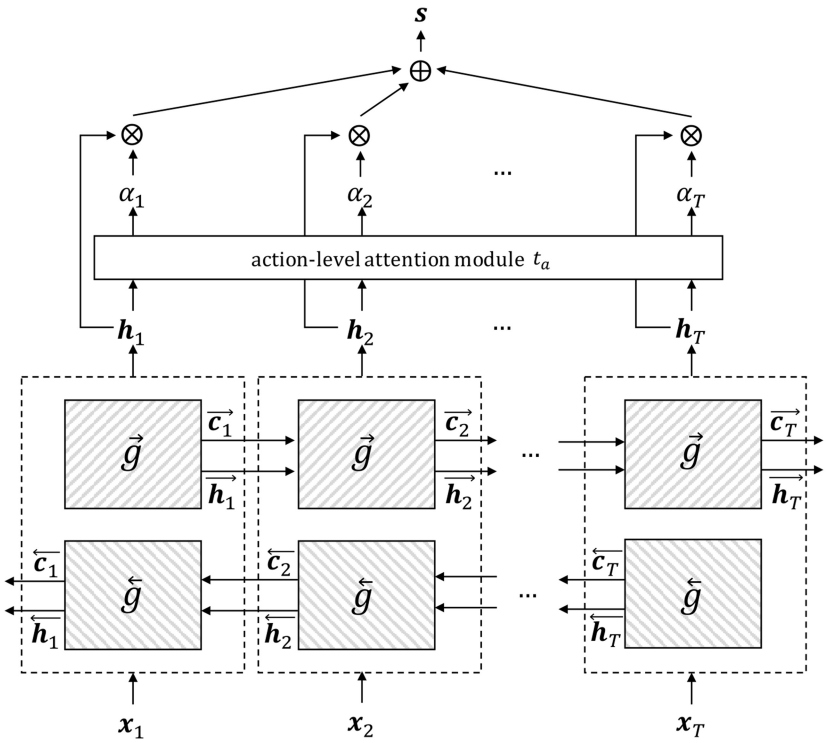

Fig. 2. The structure of the action encoder p. The rectangles are used to denote modules, and the modules that share parameters are represented with the same shading. Figure notations: x_i: representation for action i; $\overrightarrow{g}$, $\overleftarrow{g}$: the forward and backword RNNs; $\overrightarrow{c_i}$, $\overleftarrow{c_i}$: the forward and backward states of the step i; $\overrightarrow{h_i}$, $\overleftarrow{h_i}$: the forward and backward outputs of the step i; h_i: the output of the step i; α_i: the attention value for the step i; s: the session representation.

Compared to the basic RNN structure, bi-directional RNN contains a forward RNN $\overrightarrow{g}$ and a backward RNN $\overleftarrow{g}$, as shown in Figure 2. The forward RNN $\overrightarrow{g}$ reads the action sequence from $\boldsymbol{x}_1$ to $\boldsymbol{x}_T$ and generates forward contextual representations $[\overrightarrow{\boldsymbol{h}_1}, \overrightarrow{\boldsymbol{h}_2}, \ldots, \overrightarrow{\boldsymbol{h}_T}]$, while the backward RNN $\overleftarrow{g}$ reads the sequence reversely, i.e., from $\boldsymbol{x}_T$ to $\boldsymbol{x}_1$, and generates backward contextual representations $[\overleftarrow{\boldsymbol{h}_T}, \overleftarrow{\boldsymbol{h}_{T-1}}, \ldots, \overleftarrow{\boldsymbol{h}_1}]$. The final contextual representation of each position j is then the concatenation of $\overrightarrow{\boldsymbol{h}_j}$ and $\overleftarrow{\boldsymbol{h}_j}$, as shown in Equation (8):

$$\boldsymbol{h}_j = [\overrightarrow{\boldsymbol{h}_j}^\top; \overleftarrow{\boldsymbol{h}_j}^\top]^\top. \tag{8}$$

We use LSTM for the non-linear functions $\overleftarrow{g}$ and $\overrightarrow{g}$, and we refer readers to the original article [16] for details.

With contextual representations $\boldsymbol{h}_j$ encoding action context, we then weighted-sum $\boldsymbol{h}_j$ to derive the corresponding session representation $\boldsymbol{s}$, as shown in Figure 2 and Equation (9):

$$\boldsymbol{s} = \sum_{j=1}^{T} \alpha_j \boldsymbol{h}_j, \tag{9}$$

such that higher weights are assigned to the positions or actions with stronger discriminative power. Intuitively, the derived sequence representation attends to the positions that are most predictive of the user's skill level, and thus the model is able to make more accurate predictions.

We propose an attention module t_a (Figure 2), as defined below,

$$[\alpha_1, \alpha_2, \ldots, \alpha_T] = t_a([\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_T]), \tag{10}$$

where $\alpha_1 + \alpha_2 + \cdots + \alpha_T = 1$, to implement such an action-level attention mechanism. A multi-layer neural network is designed to calculate the weights. Specifically, each weight α_j is derived as Equation (11) shows:

$$\alpha_j = \frac{\exp(e_j^\alpha)}{\sum_{i=1}^{T} \exp(e_i^\alpha)}, \tag{11}$$

where $e_j^\alpha = \boldsymbol{v}_{t_a}^\top \tanh(W_{t_a}\boldsymbol{h}_j + \boldsymbol{b}_{t_a})$. As demonstrated in the experiments (Section 5), for a given position, the module t_a learns to predict its power for the end task based on the corresponding contextual representation, and such a module not only improves the prediction accuracy, but also enables direct interpretations.

3.2 Session Encoder q

With the action encoder p, each sequence of actions (i.e., a session) is represented by a vector $\boldsymbol{s}$. The task of the session encoder q is to aggregate the representations across the user's sessions and derive the user representation $\boldsymbol{v}$. Similar to the design of the action encoder, we want our session encoder to also possess the ability to distinguish the most discriminative sessions from the noisy ones, so that the predictions can mostly attend to the informative sequences. We propose a session-level attention module t_s, as defined below, to achieve the goal,

$$[\beta_1, \beta_2, \ldots, \beta_L] = t_s([\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_L]), \tag{12}$$

where $\beta_1 + \beta_2 + \cdots + \beta_L = 1$.

The weights $\beta_i (i = 1, \ldots, L)$ like the ones shown in Equation (9) are used to weighted-sum the session representations to derive the user representation (Equation (13)):

$$\boldsymbol{v} = \sum_{i=1}^{L} \beta_i \boldsymbol{s}_i, \tag{13}$$

where higher value of β is placed on the sessions with higher predictive power. We propose to use another multi-layer neural network to derive such weights, as presented below:

$$\beta_i = \frac{\exp(e_i^\beta)}{\sum_{j=1}^{L} \exp(e_j^\beta)}, \tag{14}$$

where $e_i^\beta = \boldsymbol{v}_{t_s}^\top \tanh(W_{t_s}\boldsymbol{s}_i + \boldsymbol{b}_{t_s})$.

Through extensive experiments, our session encoder is shown to enable the session-level interpretations of the prediction results, and simultaneously improve the prediction performance.

4 END-TO-END TRAINING OF H-ATT-BIRNN

Our proposed h-ATT-BiRNN model learns a non-linear function that predicts a score proportional to the user's skill level. Similar to other representation learning approaches [44], h-ATT-BiRNN can be trained using various kinds of supervisions. Without loss of generality, in this section, we describe the training procedure for pairwise and pointwise labels.

4.1 Pairwise Supervision

With pairwise supervision, each training instance contains the action histories from a *positive user* u^+ (higher skill level) and a *negative user* u^- (lower skill level). We first use a shared h-ATT-BiRNN model, i.e., with the same parameter values, to predict the scores for both users, denoted as k_{u^+} and k_{u^-}, respectively. Then we calculate the training loss using a hinge loss function (Equation (15)),

$$\begin{aligned}\mathcal{L} &= \sum_{(u^+,u^-)\in\mathbb{S}_{\text{train}}} [1 + k_{u^-} - k_{u^+}]_+ \\ &= \sum_{(u^+,u^-)\in\mathbb{S}_{\text{train}}} [1 + W_r(\max(\boldsymbol{v}_{u^-}, 0) - \max(\boldsymbol{v}_{u^+}, 0))]_+,\end{aligned} \tag{15}$$

where $\boldsymbol{v}_{u^+}$ and $\boldsymbol{v}_{u^-}$ represent user representations for u^+ and u^-, respectively, and $\mathbb{S}_{\text{train}}$ denotes the training set. Regularization terms will be discussed in Section 5.2.

Intuitively, the loss function $\mathcal{L}$ enforces a margin between k_{u^+} and k_{u^-}. The loss $\mathcal{L}$ is equal to zero only when $k_{u^+} - k_{u^-} \geq 1$. After $\mathcal{L}$ is calculated, it is then back-propagated to update each trainable parameter in the model using a gradient descent algorithm.

4.2 Pointwise Supervision

Pointwise supervision directly provides a skill label for each user. With such supervision, we train the h-ATT-BiRNN model through regression. Specifically, the model is trained to output a score approximating the true label, and the training loss is calculated as follows:

$$\begin{aligned}\mathcal{L} &= \sum_{u\in\mathbb{S}_{\text{train}}} \|k_u - \hat{k}_u\|^2 \\ &= \sum_{u\in\mathbb{S}_{\text{train}}} \|W_r \max(\boldsymbol{v}_u, 0) + b_r - \hat{k}_u\|^2,\end{aligned} \tag{16}$$

where k_u and $\hat{k}_u$ denote predicted and ground truth skill levels for user u, respectively.

5 DESIGN SKILL PREDICTION

In this section, we evaluate our model in the context of predicting digital design skill levels (the ability to produce high quality art) of Photoshop users and extensively compare it with other baseline features and algorithms.

5.1 Digital Design Skill Dataset

We use implicit supervision from the Behance platform,[3] where millions of creative professionals share their work and socialize with each other to train and evaluate the models. Specifically, we regard the users who have at least one project featured as the *positive users*, and otherwise as the *negative users* (artists uploaded their creative work to Behance, each of which was then evaluated by the editorial group, and those with high quality were picked as *featured projects*). The performance of each model is then evaluated using binary classification (against the extent to which each model ranks *positive users* higher than *negative users*), which is a widely adopted practice in evaluating regression models [8]. It is worth mentioning that the labels used in our current dataset are not perfect as the users are only divided into two groups, and the intra-group user differences are not taken into consideration. Collecting more fine-grained labels is left as future work.

We randomly select 68,456 Photoshop users who uploaded at least one project on Behance by November 2015 and have at least one session recorded from 01/01/15 to 06/30/15, then the

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

Table 1. Model Size of h-ATT-BiRNN

variables	dimensionality
$\boldsymbol{x}$, $\boldsymbol{h}$, $\boldsymbol{v}$, $\boldsymbol{s}$ (the representations)	500
$\boldsymbol{v}_{t_a}$, $\boldsymbol{v}_{t_s}$ (hidden layers)	100
$\overrightarrow{\boldsymbol{h}}$, $\overleftarrow{\boldsymbol{h}}$ (LSTM)	250

6-month action histories are included in our dataset. Among all users, 6,043 of them are identified as *positive*. The dataset is divided into a training set that includes 57,136 users (5,043 positive and 52,093 negative), and a testing set that includes 11,320 users (1,000 positive and 10,320 negative). In the training set, 500 positive users and 5,000 negative users are further held-out as a validation set for the parameter selection. Note that, in this article, we hold a static view of users' skill levels, i.e., the time series evolution of the skills is not taken into account. We will investigate the time effects in future work.

5.2 Model and Training Configurations

Given the nature of the collected dataset, we train h-ATT-BiRNN model with pairwise supervision, as discussed in Section 4.1. In our experiments, the size of the h-ATT-BiRNN model, i.e., the dimensionality of each input/output, is shown in Table 1. All of the free parameters, if not specified explicitly, are trainable, and their sizes can be easily inferred from the information presented. During training, we leverage two strategies to alleviate overfitting problems. (1) Dropout [38]. The dropout operation randomly drops a set of units during training [38], and it is shown to effectively address the overfitting problem in deep neural networks [38]. In our model, we add a dropout operation to each representation, i.e., $\boldsymbol{x}$, $\boldsymbol{s}$, and $\boldsymbol{v}$, and the outputs of the first hidden layer in the attention modules, i.e., $W_{t_a}\boldsymbol{h} + \boldsymbol{b}_{t_a}$ and $W_{t_s}\boldsymbol{h} + \boldsymbol{b}_{t_s}$. We use 0.5 as the dropout rate. (2) L_2 regularization. We add the L_2 loss of each parameter as an addition to the original loss (Equations (15) and (16)), and the ultimate training loss is calculated as follows:

$$\mathcal{L}_{\text{all}} = \mathcal{L} + \gamma \sum_{\omega} \omega^2, \tag{17}$$

where ω represents any trainable variable in our model. We select the best value of γ among $1e{-}6$, $1e{-}5$, $1e{-}4$, and $1e{-}3$ with the validation set. Apart from the afore-mentioned mechanisms, the optimal number of training iterations (within a maximum limit = 4,000) is also determined through validation.

We implement our model using the Tensorflow framework [1] and conduct the training in a GPU machine equipped with NVIDIA Titan X cards. The network is optimized via the Adam Optimization algorithm [25] with the initial learning rate set to 0.001. To leverage the parallel computing power of modern hardware, we train the network with the batch size of 100, i.e., 100 pairs of (*positive user*, *negative user*). We sample at most 20 sessions for each user and 100 consecutive actions for each session in real time to augment training data [26]. During the validation and testing, such a sampling is independently executed five times for a given user, and the final score is the average of the five predictions, which potentially captures the length effect.

5.3 Baseline Algorithms

Different from the h-ATT-BiRNN model that learns a feature extractor and a classifier end-to-end, previous approaches separate the process of the representation learning and the classifier training. In general, the feature extractor is either hand-crafted or learned in an unsupervised manner, and the classifier is trained with the features fixed. In this section, we present the baseline features and

classifiers in Sections 5.3.1 and 5.3.2, respectively. In addition, to test the effectiveness of different levels of the attention mechanisms, we compare h-ATT-BiRNN to three of its variants, which will be discussed in Section 5.3.3.

5.3.1 Baseline Features. We consider three baseline features in this article as follows.

— *Bag-of-n-grams* [11, 28]. Bag-of-n-grams counts the number of times that each n-gram action sequence is performed by the user, and the feature vector is normalized with L_1 norm. This is the standard user modeling approach employed by the service providers. Specifically, we consider uni-gram (uni), 1,2-gram (1,2) and 1,2,3-gram (1,2,3) features.
— *bag-of-n-grams with tf-idf.* One of the limitations of the standard n-gram feature is the inherently imbalanced usage frequency of different actions. For example, the actions *open* and *close* may appear much more frequently than the action *gaussian_blur*. To compensate for the frequency differences, we use *tf-idf* to re-weight the raw counts of n-grams.
— *utilization-to-vector (util2vec).* util2vec [43] is the state-of-the-art representation learning model for the software users. As is described in Section 2.4, it models the sequential patterns of actions using a distributed representation learning framework and is shown to accurately capture users' traits and preferences [43].

To make the features directly comparable to our model, we set the dimensionality of the util2vec representation to 500, which is the same as the representation learned from h-ATT-BiRNN, i.e., $\boldsymbol{v}$. The feature dimensionality for uni-gram, 1,2-gram and 1,2,3-gram are 1990, 55,431, and 111,583, respectively.[4]

5.3.2 Baseline Classifiers. Under the pairwise learning-to-rank framework [20], we choose three state-of-the-art pairwise ranking classifiers as the baselines.

— *Rank Support Vector Machine (SVM)* [20]. Given pairwise training data $(\boldsymbol{d}_i, \boldsymbol{d}_j)$ where $\boldsymbol{d}_i > \boldsymbol{d}_j$, RankSVM aims to learn a weight vector $\boldsymbol{\omega}$ such that

$$\boldsymbol{\omega}^\top \Phi(\boldsymbol{d}_i) > \boldsymbol{\omega}^\top \Phi(\boldsymbol{d}_j), \tag{18}$$

where Φ is a kernel function. The model training is driven by the optimization problem defined as follows:

$$\begin{aligned} \min_{\boldsymbol{\omega}, \xi} \quad & \frac{1}{2}\boldsymbol{\omega}^\top \boldsymbol{\omega} + C \sum_{(i,j)} \xi_{i,j} \\ \text{subject to.} \quad & \boldsymbol{\omega}^\top (\Phi(\boldsymbol{d}_i) - \Phi(\boldsymbol{d}_j)) \geq 1 - \xi_{i,j}, \forall i, j \\ & \xi_{i,j} \geq 0, \forall i, j \end{aligned}, \tag{19}$$

where C is the regularization parameter, and we select its optimal value among 0.001, 0.01, 0.1, 1, 10, 100, 1000 using the validation set. In the real world settings, we have millions of training pairs, and the non-linear kernels become intractable under such scenarios. Therefore, we choose linear RankSVM in our experiments.
— *Rank Logistic Regression (LR).* Instead of using SVM that learns a maximum margin for the ranking problem, RankLR leverages LR that maximizes the conditional likelihood of the training pairs. Specifically, Given the training pairs presented above, the weight vector $\boldsymbol{\omega}$ is trained by minimizing the following objective function:

$$\min_{\boldsymbol{\omega}} \quad \frac{1}{2}\boldsymbol{\omega}^\top \boldsymbol{\omega} + C \sum_{i,j} \log(\exp(-\boldsymbol{\omega}^\top (\boldsymbol{d}_i - \boldsymbol{d}_j)) + 1), \tag{20}$$

[4]We only retain frequent bi-grams and tri-grams so that the derived feature vectors are memory tractable.

where the regularization parameter C is also selected from the same pool of candidate values through validation.

— *Rank Gradient Boosting Decision Tree (GBDT).* Similar to SVM and LR, we additionally leverage the model-ensemble based GBDT [10] to learn the ranking function (with pairwise features $\boldsymbol{d}_i - \boldsymbol{d}_j$). For each GBDT model, we incrementally learn 100 estimators with number of leaves set to 31.

We implement the classifiers using libsvm [4], sklearn and LightBGM [23] libraries and conduct the training until they converge, i.e., the maximum value of the gradients is less than $1e{-}4$. To construct the training pairs, we experiment different sampling ratio, i.e., [number of positive users]:[number of negative users], from 1:1 to 1:50 (for 1,2-gram and 1,2,3-gram features, sampling ratios beyond 1:5 are memory-intractable).

5.3.3 Variants of h-ATT-BiRNN. We also compare h-ATT-BiRNN to the following three variants to show the effectiveness of different attention modules:

— *BiRNN.* To investigate whether the attention mechanism is effective at all, the BiRNN variant does not include any attention module, i.e., the session representation takes the average of the contextual representations ($\boldsymbol{s} = \frac{1}{T}\sum_i \boldsymbol{h}_i$), and the user representation takes the average of the session representations ($\boldsymbol{v} = \frac{1}{L}\sum_i \boldsymbol{s}_i$).

— *a-ATT-BiRNN.* To show the effectiveness of the action-level attention module t_a, a-ATT-BiRNN maintains the attentions to actions but removes module t_s from the proposed model.

— *s-ATT-BiRNN.* Similar to the variants presented above, s-ATT-BiRNN maintains the session-level attention module t_s but removes the action-level attentions, i.e., t_a.

5.4 Quantitative Results

For each method, including h-ATT-BiRNN and the baseline algorithms introduced above, we find the best performed model in the validation set, i.e., the best regularization setting, and then report its performance in the testing set. In our experiments, we measure the ranking performance of each method using Area Under ROC Curve (AUC), as defined in Equation (21):

$$\text{AUC} = \frac{1}{U}\sum_{u^+=1}^{U} \frac{\sum_{u^-} \delta(k_{u^+} > k_{u^-})}{\text{number of negative users in the testing set}}, \tag{21}$$

where U denotes the total number of positive users in the testing set. Inspired by previous research [27] showing that initializing the input embedding, i.e., X, with the distributed representations learned from a larger corpus may improve the performance (e.g., initialing word embeddings with word2vec outputs in natural language processing [27]), we conduct controlled experiments for h-ATT-BiRNN and its variants. Specifically, we derive action representations (**a2v**) using the word2vec framework [33] over usage histories from 3 million users (22 billion actions) and then experiment each model in the following three settings: (1) initialize the action embeddings with a2v and fix such embeddings during training; (2) initialize action embeddings with a2v but fine-tune them, i.e., update the parameters, during training; and (3) randomly initialize the embeddings and directly learn the representations from the pairwise supervision.

The comprehensive quantitative results are presented in Table 2, where, for previous approaches, we test the ranking performance for each combination of *feature*, *classifier*, and *sampling ratio*. The results demonstrate that our model significantly outperforms the state-of-the-art and its variants, and the hierarchical attention mechanism is effective in capturing signals that are significant for predicting digital design skill levels. Specifically, we want to highlight the following findings from Table 2.

Table 2. Models' Ranking Performances Evaluated in the Digital Design Skill Testing Set

P-N ratio	1:1	1:2	1:3	1:4	1:5	1:6	1:7	1:8	1:9	1:10	1:50
uni-LR	.669	.680	.678	.680	.679	.682	**.683**	**.683**	.682	.679	.682
uni-tfidf-LR	.672	.675	.682	.684	.683	.682	**.687**	.682	.684	.683	.685
uni-SVM	.672	.680	.679	.678	.679	.679	.677	.679	.679	.679	**.681**
uni-tfidf-SVM	.674	.676	.682	.683	.683	.684	.683	.685	.684	.683	**.686**
uni-GBDT	.702	.707	.706	.708	.708	.706	**.715**	.709	.707	.712	.708
uni-tfidf-GBDT	.698	.703	**.710**	.698	**.710**	.700	.700	.703	.705	.704	.705
util2vec-LR	.697	.698	.701	.701	.698	**.702**	**.702**	.700	.699	.701	.697
util2vec-SVM	.691	.692	**.698**	.694	.696	.697	.692	.696	.694	.694	.694
util2vec-GBDT	.670	.673	.684	.679	**.693**	.685	.688	.685	.688	.688	.686

P-N ratio	1:1	1:2	1:3	1:4	1:5
1,2-gram-LR	.683	.693	.689	.689	**.694**
1,2-gram-tfidf-LR	.696	.692	.694	.689	**.696**
1,2-gram-SVM	.682	.688	.689	**.694**	.690
1,2-gram-tfidf-SVM	.689	.688	.694	.694	**.695**
1,2-gram-GBDT	**.720**	.712	.709	.712	.716
1,2-gram-tfidf-GBDT	.711	**.716**	.711	**.716**	.713
1,2,3-gram-LR	.685	.688	.692	.693	**.698**
1,2,3-gram-tfidf-LR	.683	.692	**.698**	**.698**	.691
1,2,3-gram-SVM	.684	.687	.693	.693	**.695**
1,2,3-gram-tfidf-SVM	.694	.687	.697	**.701**	.695
1,2,3-gram-GBDT	.705	.712	.713	.712	**.716**
1,2,3-gram-tfidf-GBDT	.707	.709	.710	**.712**	.707

action embedding	a2v, fixed	a2v, fine-tuned	random init
BiRNN	.701	**.703**	.695
a-ATT-BiRNN	.698	**.702**	.698
s-ATT-BiRNN	.715	**.719**	.707
h-ATT-BiRNN	.734	.733	**.736**

We present the area under curve (AUC) values, as defined in Equation (21). The standard mean of error (SEM) of all methods are 0.008, which demonstrates the significance of the improvements (h-ATT-BiRNN over baselines). For unigram and util2vec features, we present model performance with different positive-to-negative feature sampling ratios from 1:1 to 1:50; for 1,2-gram and 1,2,3-gram features, because of the high dimensionality, we only experiment ratios from 1:1 to 1:5. The best performed setting of each row is marked with bold font. Note that GBDT is an ensemble-based method, and the performance of h-ATT-BiRNN may be further improved by combining multiple model instances with different initializations.

— *Simply scaling up the training data does not help with the ranking.* As shown in Table 2, the performances of previous methods do not improve even with orders of magnitudes more training data, i.e., the AUC value stays stable with the increasing sampling ratio. This phenomenon reveals that previous methods are not limited by the amount of data for training, but by their intrinsic capacity and expressive power in modeling noisy time-series traces.

— *It takes two to tango, the interplay between two levels of attention mechanisms are essential to an effective model.* The experimental results demonstrate that compared to baseline approaches, BiRNN and a-ATT-BiRNN do not bring benefits to the prediction, and h-ATT-BiRNN achieves the best performance. Intuitively, without session-level attention, trivial sessions such as [*open*]-[*crop*]-[*close*] are likely to contaminate the user representation, and without action-level attention, the session representations may be influenced by uninformative actions for common operations.

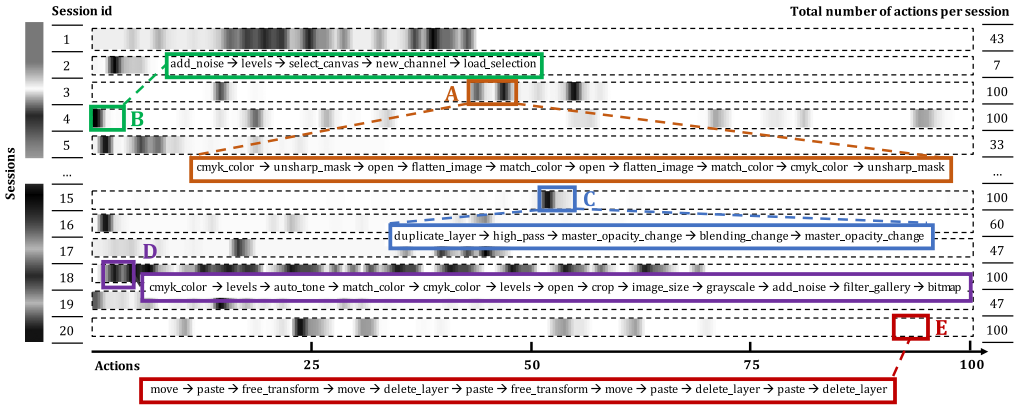

Fig. 3. The values of attention variables (h-ATT-BiRNN) for a representative positive user in the digital design skill testing set. The grey-scale of each block is proportional to the value of the corresponding attention variable. (The attentions to sessions are presented vertically on the left, and the attentions to the actions are presented horizontally).

— *The best performance can be achieved without pretraining action embeddings.* The results presented in Table 2 show that the randomly initialized model achieves the same, if not better performance with the ones initialized using a2v weights. Compared to util2vec that requires training data from millions of users [43], our model achieves significantly better performance with 50× less training data.

5.5 Qualitative Results

In addition to achieving a better ranking performance, our model also facilitates the interpretation of the predictions. Specifically, our interpretations discover sequential patterns that are predictive to the user's digital design skill level. As described in Sections 3.1 and 3.2, attention weights α_i and β_i are positive numbers, and $\sum_i \alpha_i = 1, \sum_i \beta_i = 1$. Therefore, the weight assigned to each position or session in the attention module reflects the fraction of attention that the prediction pays to the corresponding component.

We present a sample interpretation of a representative positive user in our testing set, as shown in Figure 3. The grey-scale of each block is proportional to the value of the corresponding attention variable in the h-ATT-BiRNN model. In Figure 3, we show the model's attentions to sessions and actions, and we discuss detailed interpretations as follows.

— *Only a small fraction of action sequences contribute to the prediction.* As demonstrated in the visualization, the attended sequences are sparse compared to the potentially large volume of the user's application usage history. In other words, one's skill level is, to a large extent, only determined by a small number of effective sequences.
— *The usage contexts of actions are important for the prediction.* As shown in Figure 3, although the action [*cmyk_color*] from sequences **A** and **D** gains a significant action-level weight, the session that **A** belongs to, i.e., #3, receives the minimal attention, while the other session, i.e., #18, captures much. Therefore, even for the same action, e.g., [*cmyk_color*], the context in which it is used determines the role it plays in the final prediction.
— *Action sequences accomplishing preliminary tasks may not be predictive.* Intuitively, for software applications, certain operations are preliminary and mastered by users across different skill levels. Therefore, they are not indicative of the user's expertise. Such a phenomenon is

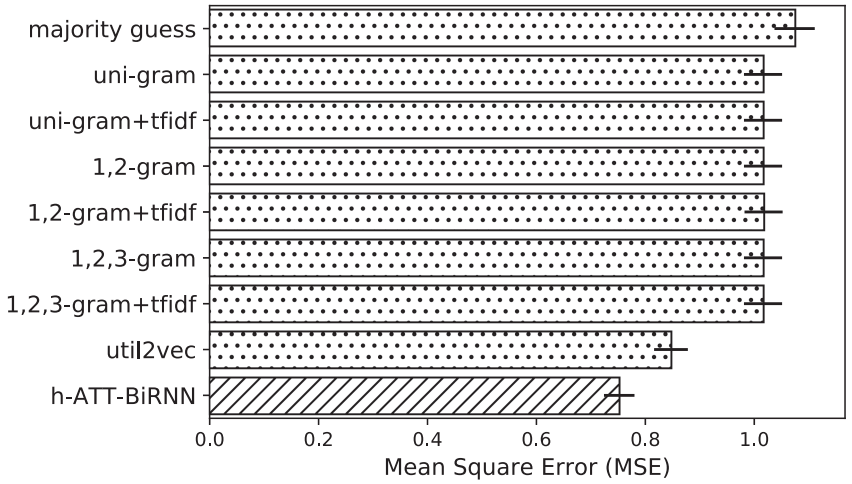

Fig. 4. Models' performances (mean square error) in predicting users' software skill levels. Baseline algorithms are shaded with dots, and our method is shaded with slashes. The error bar represents the standard error of mean, which demonstrates the significance of the improvements (h-ATT-BiRNN over baselines).

reflected in our experiment. As shown in Figure 3(**E**), the action sequence such as [*move*]-[*paste*]-[*delete_layer*] captures fairly limited attention because the task that it accomplishes is simple and basic.

— *Sequences of advanced operations usually capture significant attentions.* In contrast to those for preliminary tasks, the actions leveraging advanced software features and performing fine-grained operations usually capture a significant amount of attention in both levels. For example, sequences **B** and **C** in Figure 3 contain advanced operations for image processing, such as the noise manipulation (i.e., [*add_noise*]-[*levels*]-[*select_canvas*]) and blending (i.e., [*master_opacity_change*]-[*blending_change*]-[*master_opacity_change*]).

6 SOFTWARE SKILL PREDICTION

In addition to the digital design skill, we further evaluate our model's performance in predicting users' software skill levels, i.e., proficiencies in manipulating professional tools. To conduct this experiment, we collected a software skill dataset by surveying 8675 Photoshop users. Each user was asked to self-disclose their expertise in Photoshop usage, from 1 (novice) to 5 (expert). We split the dataset into 6175/1000/1500 for training, validation, and testing, respectively. As users' ratings are inherently ordinal, as opposed to categorical, we trained the h-ATT-BiRNN model via regression (i.e., with pointwise supervision in Section 4.2) and the same model configurations (Section 5). For comparison, we train linear regression models for *uni-gram*, *uni-gram+tfidf*, *1,2-gram*, *1,2-gram+tfidf*, *1,2,3-gram*, *1,2,3-gram+tfidf* and *util2vec* features and use them as baselines. The training of baseline models follows the same procedure as h-ATT-BiRNN.

The models' prediction performances are measured by Mean Square Error (MSE), and we present the results in Figure 4. The empirical evidence demonstrates that our model significantly and consistently outperforms baseline methods in predicting users' software skills. Compared to the digital design skill prediction task where the embedding based method (util2vec) does not bring significant improvments, the software skill prediction benefits from the embedding approach, and the util2vec feature significantly outperforms n-gram based features.

Furthermore, the results in Figure 4 justify that the application usage traces are promising data sources for user skill level predictions as h-ATT-BiRNN has much lower MSE than the majority guess. Overall, the experiments on both datasets justify the merits of h-ATT-BiRNN in assessing diverse user skills. We believe that our results have revealed the utilities and potentials of effective attention mechanisms for accurate user modeling and understanding.

7 DISCUSSIONS

In this section, we discuss the limitations of our work and the generalizability of the proposed h-ATT-BiRNN model.

7.1 Limitations

First, in the digital design skill dataset, we treat featured projects as positive signals for a user's skill level, which may overlook other factors that potentially contribute to the ground truth labels, such as the fame of the artists and the general fashion trend. Second, users' skill level may evolve over time. Because of the limitation of available datasets, such a factor is not taken into consideration in this article. Third, we do not incorporate the timestamp, action interval, action count, and session length into the model, which may provide extra information regarding the user's response time, proficiency, and activity level. We leave these as future work.

7.2 Generalizability of h-ATT-BiRNN

Although the h-ATT-BiRNN is evaluated using application usage traces from Photoshop users, the model is potentially applicable to other software and web services by redefining user actions, e.g., a command in Autodesk [28] or a query on the web [41]. This is possible because the inputs to the h-ATT-BiRNN model are abstractly defined tokens and are not restricted to the Photoshop actions. In addition, we also demonstrate two ways of collecting training labels, i.e., through expert judgments (digital design skills) and self-report (software skills), which can be easily adopted by the industry.

8 CONCLUSION AND FUTURE WORK

In this article, we propose a bi-directional RNN model with hierarchical attention mechanism that assesses software users' skill levels by leveraging the sequential patterns and the heterogeneous predictive power of the action series. The experimental results demonstrate its superior performance and interpretative power in characterizing diverse user skills. Although we only conduct evaluations with Photoshop users, we have reasons to believe that the technique studied has a great potential to benefit other domains with time-series user interaction traces, e.g., web browsing records (i.e., clicks, scrolling, and typing), Github pushes, gaming actions, and so on. In these domains, we face similar scenarios where only some fractions of data traces are informative and predictive to users' traits. Therefore, customized algorithms can be built upon our attention model.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*.

[2] Fabian Abel. 2015. We know where you should work next summer: Job recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 230–230.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

[4] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2, 3 (2011), 27.

[5] Michael J. Cole, Jacek Gwizdka, Nicholas J. Belkin, and Chang Liu. 2011. User domain knowledge and eye movement patterns during search. In *Proceedings of the 5th Workshop on Human–Computer Interaction and Information Retrieval (HCIR)*.

[6] Michael J. Cole, Xiangmin Zhang, Chang Liu, Nicholas J. Belkin, and Jacek Gwizdka. 2011. Knowledge effects on document selection in search results pages. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1219–1220.

[7] Kevyn Collins-Thompson, Paul N. Bennett, Ryen W. White, Sebastian De La Chica, and David Sontag. 2011. Personalizing web search results by reading level. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. ACM, 403–412.

[8] Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. *arXiv:1306.6078*.

[9] Michael Ekstrand, Wei Li, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2011. Searching for software learning resources using application context. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. ACM, 195–204.

[10] Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* (2001), 1189–1232.

[11] Arin Ghazarian and S. Majid Noorhosseini. 2010. Automatic detection of users' skill levels using high-frequency user interface events. *User Modeling and User-Adapted Interaction* 20, 2 (2010), 109–146.

[12] Janice D. Gobert, Michael A. Sao Pedro, Ryan S. J. D. Baker, Ermal Toto, and Orlando Montalvo. 2012. Leveraging educational data mining for real-time performance assessment of scientific inquiry skills within microworlds. *JEDM-Journal of Educational Data Mining* 4, 1 (2012), 111–143.

[13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'13)*. IEEE, 6645–6649.

[14] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1809–1818.

[15] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems*. ACM, 161–169.

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[17] Eric Horvitz, Jack Breese, David Heckerman, David Hovel, and Koos Rommelse. 1998. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 256–265.

[18] Jeff Huang, Eddie Yan, Gifford Cheung, Nachiappan Nagappan, and Thomas Zimmermann. 2017. Master maker: Understanding gaming skill through practice and habit from gameplay behavior. *Topics in Cognitive Science* 9, 2 (2017), 437–466.

[19] Jeff Huang, Thomas Zimmermann, Nachiappan Nagapan, Charles Harrison, and Bruce C Phillips. 2013. Mastering the art of war: How patterns of gameplay influence skill in Halo. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 695–704.

[20] Thorsten Joachims. 2002. Optimizing search engines using click-through data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 133–142.

[21] Ruogu Kang and Wai-Tat Fu. 2010. Exploratory information search by domain experts and novices. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*. ACM, 329–332.

[22] Sarvnaz Karimi, Falk Scholer, Adam Clark, and Sadegh Kharazmi. 2011. Domain expert topic familiarity and search behavior. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1135–1136.

[23] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*. 3149–3157.

[24] Md Adnan Alam Khan, Volodymyr Dziubak, and Andrea Bunt. 2015. Exploring personalized command recommendations based on information found in web documentation. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*. ACM, 225–235.

[25] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1097–1105.

[27] Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Katerina Tymoshenko, Alessandro Moschitti, and Lluis Marquez. 2015. Semi-supervised question retrieval with gated convolutions. *arXiv:1512.05726*.

[28] Wei Li, Justin Matejka, Tovi Grossman, Joseph A. Konstan, and George Fitzmaurice. 2011. Design and evaluation of a command recommendation system for software applications. *ACM Transactions on Computer-Human Interaction* 18, 2 (2011), 6.

[29] Chang Liu, Jingjing Liu, Michael Cole, Nicholas J Belkin, and Xiangmin Zhang. 2012. Task difficulty and domain knowledge effects on information search behaviors. *Proceedings of the Association for Information Science and Technology* 49, 1 (2012), 1–10.

[30] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.

[31] Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2016. How does domain expertise affect users' search processes in exploratory searches? In *SAL@ SIGIR*.

[32] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 897–908.

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. 3111–3119.

[34] Stanley R. Page, Todd J. Johnsgard, Uhl Albert, and C. Dennis Allen. 1996. User customization of a word processor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 340–346.

[35] Guangyuan Piao and John G. Breslin. 2016. Analyzing MOOC entries of professionals on linkedin for user modeling and personalized MOOC recommendations. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*. ACM, 291–292.

[36] Jeffrey M. Rzeszotarski and Aniket Kittur. 2011. Instrumenting the crowd: Using implicit behavioral measures to predict task performance. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. ACM, 13–22.

[37] Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[38] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[39] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. 3104–3112.

[40] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3156–3164.

[41] Ryen W. White, Susan T. Dumais, and Jaime Teevan. 2009. Characterizing the influence of domain expertise on web search behavior. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining*. ACM, 132–141.

[42] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. ACM, 495–503.

[43] Longqi Yang, Chen Fang, Hailin Jin, Matthew Hoffman, and Deborah Estrin. 2017. Personalizing software and web services by integrating unstructured application usage traces. In *Proceedings of the 26th International Conference on World Wide Web*. ACM, 485–493.

[44] Longqi Yang, Cheng-Kang Hsieh, Hongjian Yang, John P. Pollak, Nicola Dell, Serge Belongie, Curtis Cole, and Deborah Estrin. 2017. Yum-me: A personalized nutrient-based meal recommender system. *ACM Transactions on Information Systems* 36, 1 (2017), 7.

[45] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1480–1489.

[46] Xiaojun Yuan and Ryen White. 2012. Building the trail best traveled: Effects of domain knowledge on web search trailblazing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1795–1804.

[47] Xiangmin Zhang, Michael Cole, and Nicholas Belkin. 2011. Predicting users' domain knowledge from search behaviors. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1225–1226.

[48] Xiangmin Zhang, Jingjing Liu, and Michael Cole. 2013. Task topic knowledge vs. background domain knowledge: Impact of two types of knowledge on user search performance. In *Advances in Information Systems and Technologies*. Springer, 179–191.

[49] Wayne Xin Zhao, Jing Liu, Yulan He, Chin-Yew Lin, and Ji-Rong Wen. 2016. A computational approach to measuring the correlation between expertise and social media influence for celebrities on microblogs. *World Wide Web* 19, 5 (2016), 865–886.

Received December 2017; revised May 2018; accepted June 2018