




An exploratory analysis of the latent structure of process data via action sequence autoencoders

Xueying Tang^{1*} , Zhi Wang², Jingchen Liu² and Zhiliang Ying²

¹Department of Mathematics, University of Arizona, Tucson, Arizona, USA

²Department of Statistics, Columbia University, New York, New York, USA

Computer simulations have become a popular tool for assessing complex skills such as problem-solving. Log files of computer-based items record the human–computer interactive processes for each respondent in full. The response processes are very diverse, noisy, and of non-standard formats. Few generic methods have been developed to exploit the information contained in process data. In this paper we propose a method to extract latent variables from process data. The method utilizes a sequence-to-sequence autoencoder to compress response processes into standard numerical vectors. It does not require prior knowledge of the specific items and human–computer interaction patterns. The proposed method is applied to both simulated and real process data to demonstrate that the resulting latent variables extract useful information from the response processes.

1. Introduction

Problem-solving is one of the key skills for people in a world characterized by rapid changes (OECD, 2017). Computer-based items have recently become popular for assessing problem-solving skills. In such items, problem-solving scenarios can be conveniently simulated through human–computer interfaces and the problem-solving processes can be easily recorded for analysis.

In 2012, several computer-based items were designed and deployed in the Programme for the International Assessment of Adult Competencies (PIAAC) to measure adults' competency in problem-solving in technology-rich environments (PSTRE). Screenshots¹ of the interface for a released PSTRE item are shown in Figures 1–3. The opening page for the item, displayed in Figure 1, consists of two panels. The left-hand panel contains item instructions and navigation buttons, while the right-hand panel is the main medium for interaction. In this example, the right-hand panel is a web browser showing a job searching website. The task is to find all job listings that meet the criteria described in the instructions. The drop-down menus and radio buttons can be used to narrow down the search range. Once the 'Find Jobs' button is clicked, jobs that meet the selected criteria are listed on the web page as shown in Figure 2. Participants can read the detailed information about a listing by clicking 'More about this position'. Figure 3 is the detailed information page for the first listing in Figure 2. If a listing is considered to meet all the

*Correspondence should be addressed to Xueying Tang, Department of Mathematics, University of Arizona, 617 N. Santa Rita Ave, Tucson, AZ 85721, USA (email: xytang@math.arizona.edu).

¹ Retrieved from <https://piaac-logdata.tba-hosting.de/>

requirements, it can be saved by clicking the 'SAVE this listing' button. When a participant works on a problem, the entire response process is recorded in the log files in addition to the final response outcome (correct/incorrect). For example, if a participant selected 'Photography' and '7 days' in the two drop-down menus, clicked the 'Part-time' radio button, then clicked 'Find Jobs', read the detailed information for the first listing and saved it, then a sequence of actions, 'Start, Dropdown1_Photography, Dropdown2_7, Part-time, Find_Jobs, Click_W1, Save, Next', is recorded in the log files.² The entire action sequence constitutes a single observation of process data. It tracks all the major actions the participants took when they interacted with the browsing environment.

The process responses contain substantially more comprehensive information from respondents than traditional item responses, which are often dichotomous (correct/incorrect) or polytomous (partial credits). On the other hand, to what extent this information is useful for educational and cognitive assessments and how to systematically make full use of such information are largely unknown. One of the difficulties in analysing process data is to cope with its non-standard format. Each process is a sequence of categorical variables (mouse clicks and keystrokes) and its length varies across observations. As a result, existing models for traditional item responses such as item response theory (IRT) models are not directly applicable to process data. Although some

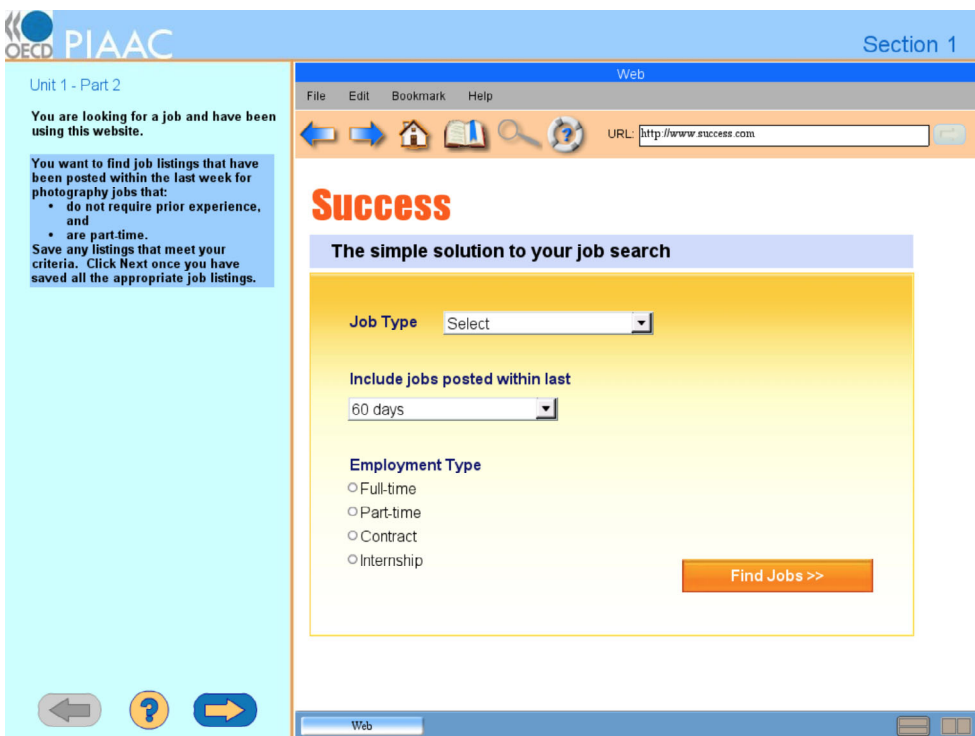


Figure 1. Main page of the sample item. [Colour figure can be viewed at wileyonlinelibrary.com]

² This example item is not used in real practice. The coding of the actions and the action sequence described above were created for illustration purposes.

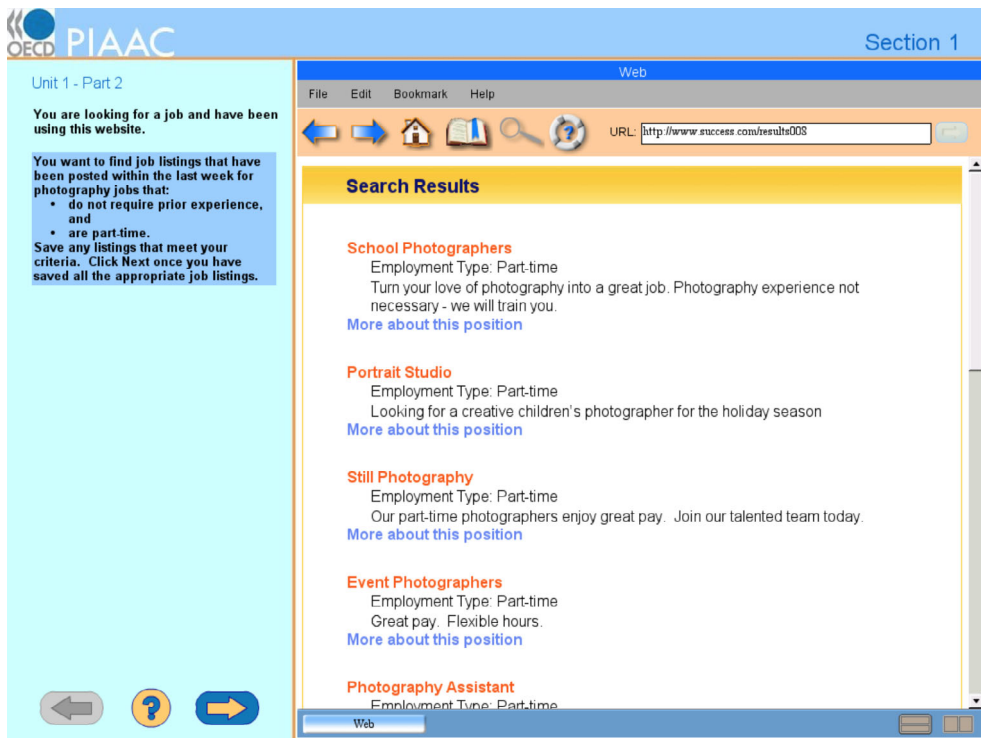


Figure 2. Web page after clicking 'Find Jobs' in Figure 1. [Colour figure can be viewed at wileyonlinelibrary.com]

models have been extended to incorporate item response time (Klein Entink, Fox, & van der Linden, 2009; S. Wang, Zhang, Douglas, & Culpepper, 2018; Zhan, Jiao, & Liao, 2018), similar extensions for response processes are difficult.

Another challenge for analysing process data comes from the wide diversity of human behaviours. Signals from behavioural patterns in response processes are often attenuated by a large number of noisy actions. Figure 4 presents the one-step and two-step lagged correlation of the response processes from one of the PSTRE items in PIAAC 2012. The lagged correlation matrix is analogous to the autocorrelation function of time series (Brockwell and Davis, 2016). A positive (negative) element of the one-step lagged correlation matrix indicates that the action in the corresponding column tends to (does not tend to) appear one step after the action in the corresponding row. The elements in the two-step lagged correlation matrix can be interpreted similarly. As most of the elements in the two matrices are close to zero, there is no clear correlation between two actions appearing consecutively or one step away. Thus if we use models only taking into account short-term dependence to analyse process data, for instance, autoregressive or moving average models, we will find that the response process is largely random noise with no structure.

The rich variety of computer-based items also adds to the difficulty in developing general methods for process data. The computer interface involved in the PSTRE items in PIAAC 2012 includes web browsers, mail clients and spreadsheets. The tasks required in these items also vary greatly. In some recent developments of process data analysis such as Greiff, Niepel, Scherer, and Martin (2016) and Kroehne and Goldhammer (2018), process

data are first summarized into several variables according to domain knowledge and then their relationship with other variables of interest is investigated by conventional statistical methods. The design of the summary variables is usually item-specific and requires a thorough understanding of respondents' cognitive process during human-computer

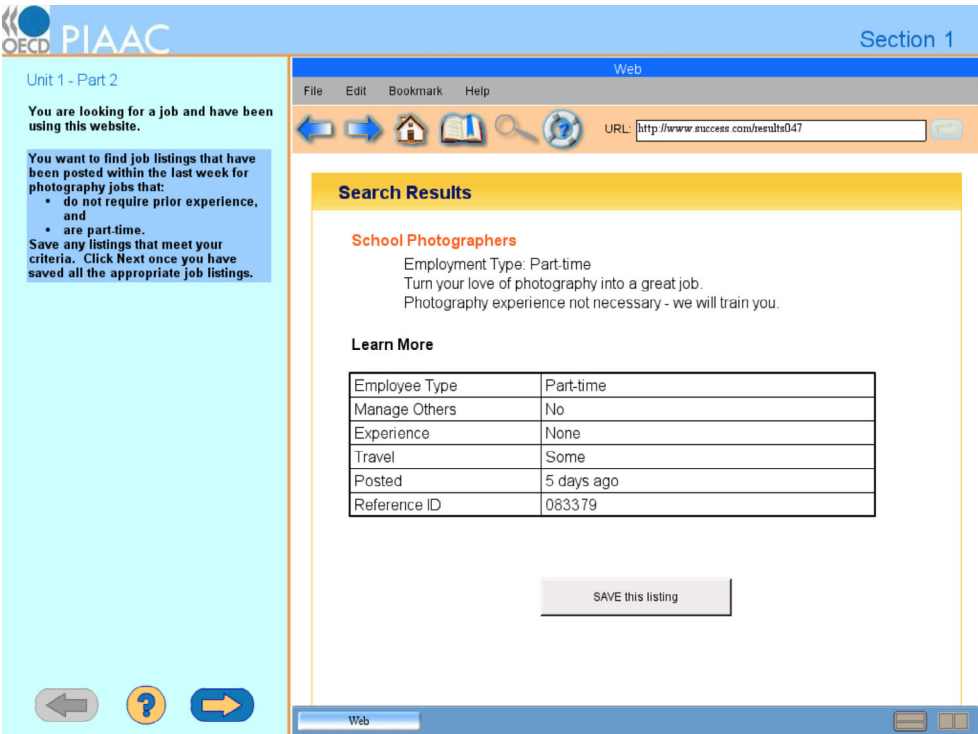


Figure 3. Detailed information page for the first job listing in Figure 2. [Colour figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com)]

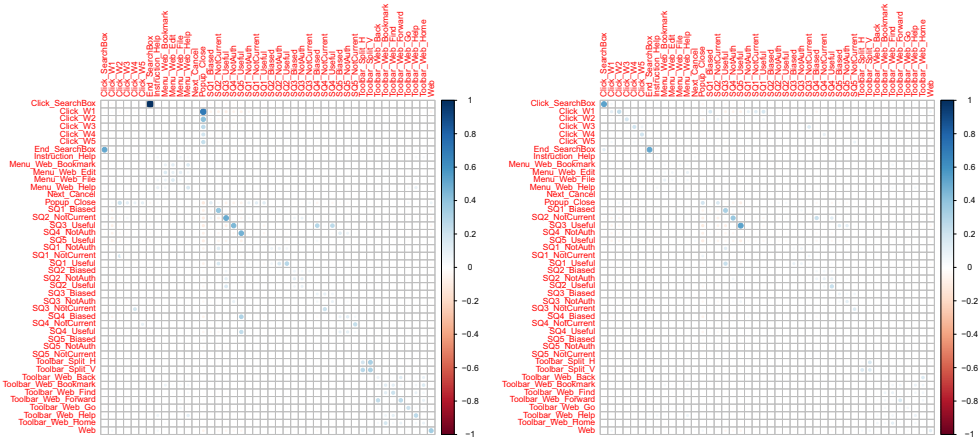


Figure 4. One-step (left) and two-step (right) lagged correlation of the response processes from item U06b of PIAAC 2012. [Colour figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com)]

interaction. Thus these approaches are too 'expensive' to apply to even a moderate number of diverse items such as the PSTRE items in PIAAC 2012. He and von Davier (2016) adopted the concept of n -grams from natural language processing to explore the association between action sequence patterns and traditional item responses. The sequence patterns extracted from their procedure depend on the coding of log files and are often of limited capacity since it only considers consecutive actions.

In this paper we propose a generic method to extract features from process data. The extracted features play a similar role to the latent variables in item response theory (Lord, 1980; Lord & Novick, 1968). The proposed method does not rely on prior knowledge of the items and coding of the log files. Therefore, it is applicable to a wide range of process data with little item-specific processing effort. In the case study, we applied the proposed method to 14 PSTRE items in PIAAC 2012. These items vary widely in many aspects, including the content of the problem-solving task and their overall difficulty levels.

The main component of the proposed feature extraction method is an autoencoder (Goodfellow, Bengio, & Courville, 2016, Chapter 14). It is a class of artificial neural networks that tries to reproduce the input in its output. Autoencoders are often used for dimension reduction (Hinton & Salakhutdinov, 2006) and data denoising (Vincent, Larochelle, Bengio, & Manzagol, 2008) in pattern recognition, computer vision, and many other machine learning applications (Deng *et al.*, 2010; Li, Luong, & Jurafsky, 2015; Lu, Tsao, Matsuda, & Hori, 2013; Yousefi-Azar, Varadharajan, Hamey, & Tupakula, 2017). They first map the input to a low-dimensional vector, from which they then try to reconstruct the input. Once a good autoencoder is found for a data set, the low-dimensional vector contains comprehensive information of original data and thus can be used to summarize the response processes.

With the proposed method, we extract features from each of the PSTRE items in PIAAC 2012 and explore the extracted feature space of process data. We show that the extracted features from response processes contain more information than the traditional item responses. We find that the prediction of many variables, including literacy and numeracy scores and a variety of background variables, can be considerably improved once the process features are incorporated.

Neural networks have recently been used to analyse educational data. Piech *et al.* (2015) and Wang, Sy, Liu, and Piech (2017) applied recurrent neural networks to knowledge tracing and showed that their deep knowledge tracing models can predict students' performance on the next exercise from their exercise trajectories more accurately than other traditional methods. Bosch and Paquette (2017) discussed several neural network architectures that can be used to analyse interaction log data. They extracted features for detecting student boredom through modelling the relations of student behaviours in two time intervals. The log file data used there were aggregated into a more regular form. Ding, Yang, Yeung, and Pong (2019) also studied the problem of extracting features from students' learning processes using autoencoders. The learning processes considered there have a fixed number of steps and the data in each step were preprocessed into fixed-dimension raw features.

The rest of this paper is organized as follows. In Section 2 we introduce the action sequence autoencoder and the feature extraction procedure for process data. The proposed procedure is applied to simulated processes in Section 3 to demonstrate how extracted features reveal the latent structure in response processes. Section 4 presents a case study of process data from PIAAC PSTRE items to show that response processes contain more information than traditional responses. Some concluding remarks are made in Section 5.

2. Feature extraction by action sequence autoencoder

We adopt the following setting throughout this paper. Let $\mathcal{A} = \{a_1, \dots, a_N\}$ denote the set of possible actions for an item, where N is the total number of distinct actions and each element in \mathcal{A} is a unique action. A response process can be represented as a sequence of actions, $\mathbf{s} = (s_1, \dots, s_T)$, where $s_t \in \mathcal{A}$ for $t = 1, \dots, T$ and T denotes the length of the process, that is, the total number of actions that a respondent took to solve the problem. An action sequence \mathbf{s} can equivalently be represented as a $T \times N$ binary matrix $\mathbf{S} = (\mathbf{S}_{ij})$ whose t th row gives the dummy variable representation of the action at time-step t . More specifically, $S_{ij} = 1$ indicates that the t th action of the sequence is action a_j . There is one and only one element equal to 1 in each row. All other elements are 0. In the rest of this paper, \mathbf{S} is used interchangeably with \mathbf{s} to refer to an action sequence.

The length of a response process is likely to vary widely across respondents. As a result, the matrix representation of response processes from different respondents will have different numbers of rows. For a set of n processes, $\mathbf{s}_1, \dots, \mathbf{s}_n$ (equivalently, $\mathbf{S}_1, \dots, \mathbf{S}_n$), the length of \mathbf{s}_i (the number of rows in \mathbf{S}_i) is denoted by T_i , for $i = 1, \dots, n$. The main motivation for developing a feature extraction method for process data is to compress the non-standard data with varying dimension into homogeneous dimension vectors to facilitate subsequent standard statistical analysis. In the remainder of this section, we describe a feature extraction method built on a type of neural network called an autoencoder. We start the description with a brief introduction to the structure and the working mechanism of a generic autoencoder in Section 2.1. In Section 2.2 we introduce recurrent neural networks, a type of neural network that is often used to process sequential information in data. In Section 2.3 we put the pieces together to construct an autoencoder that is suitable for extracting features from action sequences in process data.

2.1. Autoencoder

The main component of our feature extraction method is an autoencoder (Goodfellow *et al.*, 2016, Chapter 14). It is a type of artificial neural network whose output tries to reproduce the input. A trivial solution to this task is to link the input and the output through an identity function, but this provides little insight into the data. Autoencoders employ special structures in the mapping from the input to the output so that non-trivial reconstructions are formed to reveal the underlying low-dimensional structure. As illustrated in Figure 5, an autoencoder consists of two components, an encoder ϕ and a decoder ψ . The encoder ϕ transforms a complex and high-dimensional input \mathbf{s} into a low-dimensional vector $\boldsymbol{\theta}$. Then the decoder ψ reconstructs the input from $\boldsymbol{\theta}$. Since the low-dimensional vector is in a standard and simpler format and contains adequate information to restore the original data, autoencoders are often used for dimension reduction and feature extraction.

The encoder and the decoder are often specified as a family of functions, $\phi_{\boldsymbol{\eta}}$ and $\psi_{\boldsymbol{\xi}}$, respectively, where $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ are parameters to be estimated by minimizing the discrepancy

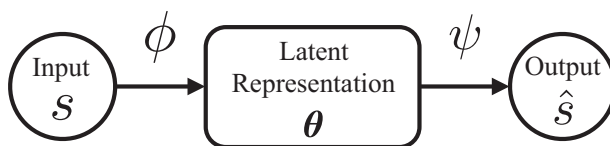


Figure 5. Structure of an autoencoder.

between the inputs and the outputs of the autoencoder. To be more specific, letting $\hat{\mathbf{s}}_i = \psi_{\xi}(\phi_{\eta}(\mathbf{s}_i))$ denote the output for input \mathbf{s}_i , $i = 1, \dots, n$, the parameters η and ξ are estimated by minimizing

$$F(\eta, \xi) = \sum_{i=1}^n L(\mathbf{s}_i, \hat{\mathbf{s}}_i), \quad (1)$$

where L is a loss function measuring the difference between the reconstructed data $\hat{\mathbf{s}}_i$ and the original data \mathbf{s}_i . Once estimates $\hat{\eta}$ and $\hat{\xi}$ are obtained, the latent representation or the features of an action sequence \mathbf{s} can be computed by $\boldsymbol{\theta} = \phi_{\hat{\eta}}(\mathbf{s})$.

To draw an analogy to IRT or other latent variable models, one may consider $\boldsymbol{\theta}$, the output of the encoder ϕ , to be an estimator of the latent variables based on the responses, and the decoder ψ to be the item response function that specifies the response distribution corresponding to a latent vector. For the IRT model, the estimator and the item response function are often coherent in the sense that the estimator is determined by the item response function. For an autoencoder, both ϕ and ψ are parameterized and estimated based on the data. There is no coherence guarantee between them. This is one of the theoretical drawbacks of autoencoders. Nonetheless, we hope that the parametric families for ϕ and ψ are flexible enough so that they can be consistently estimated with large samples and thus approximate coherence is automatically achieved.

Based on the above discussion, a crucial step in the application of autoencoders is to specify an encoder and a decoder that are suitable for the data to be compressed. In the remainder of this section, we will describe an autoencoder that performs well for response processes.

2.2. Recurrent neural network

To facilitate the presentation, we first provide a brief introduction to recurrent neural networks (RNNs), a pivotal component of the encoder and the decoder of the action sequence autoencoder.

RNNs form a class of artificial neural networks that deal with sequences. Unlike traditional artificial neural networks such as multi-layer feed-forward networks (FFNs; Patterson & Gibson, 2017, Chapter 2) that treat an input as a simple vector, RNNs have a special structure to utilize the sequential information in the data. As depicted in Figure 6, the basic structure of RNNs has three components: inputs, hidden states, and outputs, each of which is a multivariate time series. The inputs $\mathbf{x}_1, \dots, \mathbf{x}_T$ are K -dimensional vectors. The hidden states $\mathbf{m}_1, \dots, \mathbf{m}_T$ are also K -dimensional and can be viewed as the memory that helps process the input information sequentially. The hidden state evolves as the input evolves. Each \mathbf{m}_t summarizes what has happened up to time t by integrating the current information \mathbf{x}_t with the previous memory \mathbf{m}_{t-1} , that is, \mathbf{m}_t is a function of \mathbf{x}_t and \mathbf{m}_{t-1} ,

$$\mathbf{m}_t = f(\mathbf{x}_t, \mathbf{m}_{t-1}), \quad (2)$$

for $t = 1, \dots, T$. The initial hidden state \mathbf{m}_0 is often set to be the zero vector. To extract from memory the information that is useful for subsequent tasks, a K -dimensional output vector \mathbf{y}_t is produced as a function of the hidden state \mathbf{m}_t at each time-step t ,

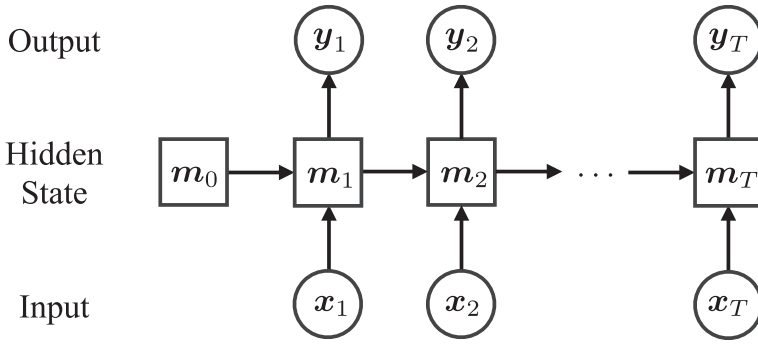


Figure 6. Structure of RNNs.

$$y_t = g(m_t). \quad (3)$$

Both f and g are often specified as a parametric family of functions with parameters to be estimated from data.

To summarize, an RNN makes use of the current input x_t and a summary of previous information m_{t-1} to produce updated summary information m_t , which in turn produces an output y_t at each time-step t . An RNN is not a probabilistic model. It does not specify the probability distribution of the input x_t or the output y_t given the hidden state m_t . It is essentially a deterministic nonlinear function that takes a sequence of vectors and outputs another sequence of vectors. Each output vector summarizes the useful information in the input vectors up to the current time-step. We will write the function induced by an RNN as $\mathcal{R}(\cdot; \gamma)$, where γ collects the parameters in f and g . Letting $\mathbf{X} = (x_1, \dots, x_T)^T$ and $\mathbf{Y} = (y_1, \dots, y_T)^T$ respectively denote the inputs and the outputs of the RNN, we have $\mathbf{Y} = \mathcal{R}(\mathbf{X}; \gamma)$. We use a subscript t on \mathcal{R} to denote the output vector at time-step t , that is, $y_t = \mathcal{R}_t(\mathbf{X}; \gamma)$.

RNNs can process sequences of different lengths. Note that the functions f and g in (2) and (3) are the same across time-steps. Therefore, the total number of parameters for an RNN does not depend on the number of time-steps.

Various choices of f and g have been proposed to compute the hidden states and outputs. Two of the most widely used are the long-short-term-memory (LSTM) unit (Hochreiter & Schmidhuber, 1997) and the gate recurrent unit (GRU; Cho *et al.*, 2014). These are designed to mitigate the vanishing or exploding gradient problem of a basic RNN (Bengio, Simard, & Frasconi, 1994). We will also use the two designs in the RNN component of our action sequence autoencoder. Detailed expressions for the LSTM unit and GRU are given in Appendix.

2.3. Action sequence autoencoder

The action sequence autoencoder used for extracting features from process data takes a sequence of actions as the input process and outputs a reconstructed sequence. Figure 7 illustrates the structure of the action sequence autoencoder. In what follows, we elaborate the encoding and the decoding mechanism.

2.3.1. Encoder

The encoder of the action sequence autoencoder takes a sequence of actions and outputs a K -dimensional vector as a compressed summary of the input action sequence. Working

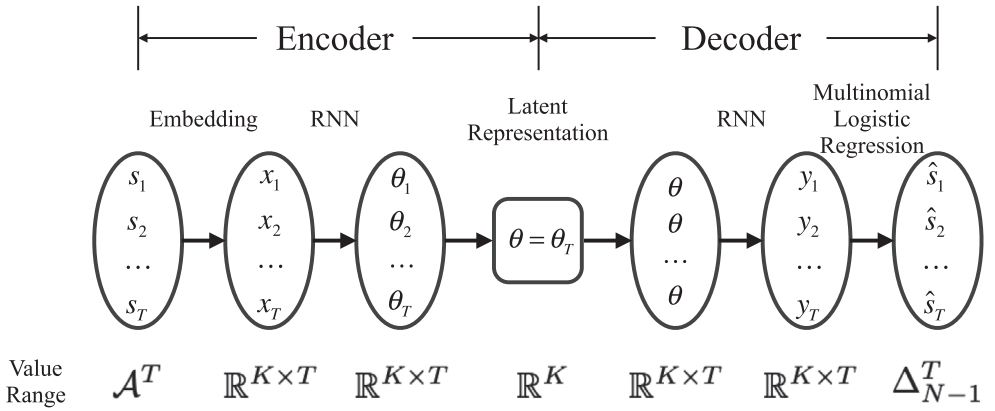


Figure 7. Structure of action sequence autoencoders. The bottom row gives the value range of each component, where $\Delta_{N-1} = \{(p_1, \dots, p_N) : \sum_{i=1}^N p_i = 1, p_i \geq 0, i = 1, \dots, N\}$.

with action sequences directly is often challenging because of the categorical nature of the actions. To overcome this obstacle, we associate each action a_i in the action pool \mathcal{A} with a K -dimensional latent vector \mathbf{e}_i that will be estimated based on the data. These latent vectors describe the attributes of actions and will be used to summarize the information contained in the sequence. The method of mapping categorical variables to continuous latent attributes is often called the embedding method. It is widely used in machine learning applications such as neural machine translation and knowledge graph completion (Bengio, Ducharme, Vincent, & Jauvin, 2003; Kraft, Jain, & Rush, 2016; Mikolov, Sutskever, Chen, Corrado, & Dean, 2013).

The first operation of our encoder is to transform the input sequence $\mathbf{s} = (s_1, \dots, s_T)$ into a corresponding sequence of latent vectors $(\mathbf{e}_{i_1}, \dots, \mathbf{e}_{i_T})$, where i_t is the index of the action in \mathcal{A} at time-step t , that is, $s_t = a_{i_t}$ for $t = 1, \dots, T$. With the binary matrix representation \mathbf{S} of action sequence \mathbf{s} , the embedding step of the encoder is simply a matrix multiplication $\mathbf{X} = \mathbf{SE}$, where $\mathbf{E} = (\mathbf{e}_1, \dots, \mathbf{e}_N)^T$ is an $N \times K$ matrix whose i th row is the latent vector for action a_i and the rows of $\mathbf{X} = (\mathbf{e}_{i_1}, \dots, \mathbf{e}_{i_T})^T$ form the latent vector sequence corresponding to the original action sequence \mathbf{s} .

Given the latent vector sequence, the encoder uses an RNN to summarize the information. Since our goal is to compress the entire response process into a single K -dimensional vector, only the last output vector of the RNN is kept to serve as a summary of information. Therefore, the output of the encoder, that is, the latent representation of the input sequence, is $\boldsymbol{\theta} = \mathcal{R}_T(\mathbf{X}; \boldsymbol{\gamma}_E)$.

To summarize, the encoder of our action sequence autoencoder is

$$\phi_{\boldsymbol{\eta}}(\mathbf{S}) = \mathcal{R}_T(\mathbf{SE}; \boldsymbol{\gamma}_E), \quad (4)$$

where $\boldsymbol{\eta}$ represents all the parameters, including the embedding matrix \mathbf{E} and the parameter vector $\boldsymbol{\gamma}_E$ of the encoder RNN. The encoding procedure consists of the following three steps.

1. An observed action sequence is transformed into a sequence of latent vectors by the embedding method: $\mathbf{X} = \mathbf{SE}$.
2. The latent vector sequence is processed by the encoder RNN to obtain another sequence of vectors $\mathcal{R}(\mathbf{X}; \boldsymbol{\gamma}_E) = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_T)^T$, where $\boldsymbol{\theta}_t = \mathcal{R}_t(\mathbf{X}; \boldsymbol{\gamma}_E)$ for $t = 1, \dots, T$.

3. The last output of the RNN is kept as the latent representation, namely, $\theta = \theta_T$.

Each of the three steps corresponds to an arrow in the encoder part of Figure 7.

2.3.2. Decoder

The decoder of the action sequence autoencoder reconstructs an action sequence \mathbf{s} , or equivalently its binary matrix representation \mathbf{S} , from θ . First, a different RNN is used to expand the latent representation θ into a sequence of vectors, each of which contains the information on the action at the corresponding time-step. As θ is the only information available for the reconstruction, the input of the decoder RNN is the same θ for each of the T time-steps. Written in a matrix form, the input of the decoder RNN is $1_T \theta^T$, where 1_T is a T -dimensional vector of ones. After the decoder RNN's processing, we obtain a sequence of K -dimensional vectors $\mathbf{Y} = (y_1, \dots, y_T)^T = \mathcal{R}(1_T \theta^T; \gamma_D)$. Each y_t contains the information for the action taken at time-step t .

Recall that each row of \mathbf{S} is the dummy variable representation of the action at a particular time. Each row essentially specifies a degenerate categorical distribution on \mathcal{A} , with the action that is actually taken having probability 1 and all the other actions having probability 0. With this observation, the task of restoring the action at step t becomes one of constructing the probability distribution of the action taken at step t from y_t . The multinomial logit model (MLM) can be used in the decoder to achieve this. To be more specific, the probability of taking action a_j at time t is

$$\hat{S}_{tj} = \begin{cases} \frac{\exp(b_j + y_t^T \beta_j)}{1 + \sum_{k=1}^{N-1} \exp(b_k + y_t^T \beta_k)} & \text{if } j = 1, \dots, N-1, \\ \frac{1}{1 + \sum_{k=1}^{N-1} \exp(b_k + y_t^T \beta_k)} & \text{if } j = N, \end{cases} \quad (5)$$

where b_j and β_j are parameters to be estimated from the data. Note that the parameters in (5) do not depend on t . That is, the encoder uses the same MLM to compute the probability distribution of s_t from y_t for $t = 1, \dots, T$. As a result, the reconstructed sequence is $\hat{\mathbf{S}} = (\hat{S}_{tj})$ and the decoder can be written as

$$\psi_{\xi}(\theta) = MLM(\mathcal{R}(1_T \theta^T; \gamma_D)), \quad (6)$$

where the parameter vector ξ consists of the parameter vector γ_D in the decoder RNN and $b_j, \beta_j, j = 1, \dots, N-1$.

If we have an ideal autoencoder that reconstructs the input perfectly, the probability distribution specified by $(\hat{S}_{t1}, \dots, \hat{S}_{tN})$ will concentrate all its probability mass on the action that is actually taken. In practice, such a perfect autoencoder is unlikely to be constructed. Usually, every action in the action set \mathcal{A} will be assigned a positive probability in the reconstructed probability distribution. For a given set of response processes, we want to manipulate the parameters in the encoder and the decoder so that the reconstructed probability distribution concentrates as much probability mass on the actual action as possible.

To summarize, as depicted in the decoder part of Figure 7, the decoding procedure of the action sequence autoencoder consists of the following three steps.

1. The latent representation θ is replicated T times to form the $T \times K$ matrix $1_T \theta^T$.
2. The decoder RNN takes $1_T \theta^T$ and outputs a sequence of vectors (y_1, \dots, y_T) , each element of which contains the information on the action at the corresponding step.
3. The probability distribution of s_t is computed according to the MLM from y_t at each time-step t .

2.3.3. Loss function

In order to extract good features for a given set of response processes, we need to construct an action sequence autoencoder that reconstructs the response processes as well as possible. The discrepancy between an action sequence \mathbf{S} and its reconstructed version $\hat{\mathbf{S}}$ can be measured by the loss function

$$L(\mathbf{S}, \hat{\mathbf{S}}) = -\frac{1}{T} \sum_{t=1}^T \sum_{j=1}^N S_{tj} \log(\hat{S}_{tj}). \quad (7)$$

Note that, for a given t , only one of (S_{t1}, \dots, S_{tN}) is non-zero. The loss function is smaller if the distribution specified by $(\hat{S}_{t1}, \dots, \hat{S}_{tN})$ is more concentrated on the action that is actually taken at step t . The best action sequence autoencoder for describing a given set of response processes is the one that minimizes the total reconstruction loss defined in (1).

Notice that (7) is in the same form as the log-likelihood function of categorical distributions. By using this loss function, we implicitly define a probabilistic model for the response processes. That is, given the latent representation θ , s_t follows a categorical distribution on \mathcal{A} with probability vector $(\hat{S}_{t1}, \dots, \hat{S}_{tN})$. The decoder of the action sequence autoencoder specifies the functional form of the probability vector in terms of θ and ξ .

2.4. Procedure

Based on the above discussion, we extract K features from n response processes $\mathbf{S}_1, \dots, \mathbf{S}_n$ through the following procedure.

Procedure 1 (Feature extraction for process data).

1. Find a minimizer, $(\hat{\eta}, \hat{\xi})$, of the objective function $F(\eta, \xi) = \sum_{i=1}^n L(\mathbf{S}_i, \hat{\mathbf{S}}_i)$ by stochastic gradient descent through the following steps.
 - a. Initialize the parameters η and ξ .
 - b. Randomly generate i from $\{1, \dots, n\}$ and update η and ξ with $\eta - \alpha \frac{\partial L(\mathbf{S}_i, \hat{\mathbf{S}}_i)}{\partial \eta}$ and $\xi - \alpha \frac{\partial L(\mathbf{S}_i, \hat{\mathbf{S}}_i)}{\partial \xi}$, respectively, where α is a predetermined small positive number.
 - c. Repeat step (b) until convergence.
2. Calculate $\tilde{\theta}_i = \phi_{\hat{\eta}}(\mathbf{S}_i)$, for $i = 1, \dots, n$. Each column of $\tilde{\Theta} = (\tilde{\theta}_1, \dots, \tilde{\theta}_n)^T$ is a raw feature of the response processes.
3. Perform principal component analysis on $\tilde{\Theta}$. The principal components are the K principal features of the response processes.

In step 1, the optimization problem is solved by stochastic gradient descent (SGD; Robbins & Monro 1951). In step 1(b), a fixed step size α is used to update the

parameters. Data-dependent step sizes such as those proposed in Duchi, Hazan, and Singer (2011), Zeiler (2012) and Kingma and Ba (2015) can easily be adapted for the optimization problem.

Neural networks are often overparameterized. To prevent overfitting, validation-based early stopping (Prechelt, 2012) is often used when estimating parameters of complicated neural networks such as our action sequence autoencoder. With this technique, the optimization algorithm (in our case, SGD) is not run until convergence. A parameter value that is obtained before convergence with good performance on the validation set is used as an estimate of the minimizer. To perform early stopping, a data set is split into a training set and a validation set. A chosen optimization algorithm is performed only on the training set for a number of epochs. An epoch consists of n_T iterations, where n_T is the size of the training set. At the end of each epoch, the objective function is evaluated on the validation set. The value of the parameters producing the lowest validation loss is used as an estimate of the minimizer. We adopt this technique when constructing the action sequence autoencoder. The feature extraction procedure with validation-based early stopping is summarized in the following procedure.

Procedure 2 (Feature extraction with validation-based early stopping).

1. Find a minimizer, $(\hat{\eta}, \hat{\xi})$, of the objective function $F(\eta, \xi) = \sum_{i=1}^n L(\mathbf{S}_i, \hat{\mathbf{S}}_i)$ by stochastic gradient descent with validation-based early stopping through the following steps.
 - a. Randomly split $\{1, \dots, n\}$ into a training index set Ω_T of size n_T and a validation index set Ω_V of size n_V .
 - b. Initialize the parameters η and ξ and calculate $F_{V1} = \sum_{i \in \Omega_V} L(\mathbf{S}_i, \hat{\mathbf{S}}_i)$.
 - c. Randomly permute the indices in Ω_T and denote the result by (i_1, \dots, i_{n_T}) .
 - d. For $k = 1, \dots, n_T$, update η and ξ with $\eta - \alpha \frac{\partial L(\mathbf{S}_{i_k}, \hat{\mathbf{S}}_{i_k})}{\partial \eta}$ and $\xi - \alpha \frac{\partial L(\mathbf{S}_{i_k}, \hat{\mathbf{S}}_{i_k})}{\partial \xi}$, respectively.
 - e. Calculate $F_{V2} = \sum_{i \in \Omega_V} L(\mathbf{S}_i, \hat{\mathbf{S}}_i)$. If F_{V2} is smaller than F_{V1} , let $\hat{\eta} = \eta$ and $\hat{\xi} = \xi$ and update F_{V1} with F_{V2} .
 - f. Repeat steps (c), (d), and (e) sufficiently many times.
2. Calculate $\hat{\theta}_i = \phi_{\hat{\eta}}(\mathbf{S}_i)$, for $i = 1, \dots, n$. Each column of $\tilde{\Theta} = (\tilde{\theta}_1, \dots, \tilde{\theta}_n)^T$ is a raw feature of the response processes.
3. Perform principal component analysis on $\tilde{\Theta}$. The principal components are the K principal features of the response processes.

The proposed feature extraction procedure requires the number of features to be extracted, K , as an input. In general, if K is too small, the action sequence autoencoder will not have enough flexibility to capture the structure of the response processes. On the other hand, if K is too big, the extracted features will contain too much redundant information, causing overfitting and instability in downstream analyses. We adopt the k -fold cross-validation procedure (Stone, 1974) to choose a suitable K in the analyses presented in Sections 4.4 and 4.5.

We perform principal component analysis on the raw features in the last step of the proposed feature extraction procedure to seek interpretations. No dimension reduction is achieved in this step and we keep all K principal components of the K raw features $\hat{\theta}$ obtained in step 2. As we will show in the case study in Section 4, the first several principal features usually have clear interpretations even if interpretability is not taken into account in the feature extraction procedure.

Since the extracted features have a standard format, they can be easily incorporated into (generalized) linear models and many other well-developed statistical procedures. As we will show in the sequel, the extracted features contain a substantial amount of information about the action sequences. They can be used as surrogates for the action sequences to study how response processes are related to the respondents' latent traits and other quantities of interest.

3. Simulations

3.1. Experiment settings

In this section we apply the proposed feature extraction method to simulated response processes of an item with 26 possible actions. Each action in the item is denoted by an upper-case English letter. In other words, we define $\mathcal{A} = \{A, B, \dots, Z\}$. All the sequences used in the study start with A and end with Z, meaning that A and Z represent the start and end of an item, respectively.

In our simulation study, action sequences are generated from Markov chains. That is, given the first t actions in a response process, s_1, \dots, s_t , the distribution from which s_{t+1} is generated depends only on s_t . A Markov chain is determined by its probability transition matrix $\mathbf{P} = (p_{ij})_{1 \leq i, j \leq N}$, where $p_{ij} = P(s_{t+1} = a_j | s_t = a_i)$. Because of the special meaning of actions A and Z, there should not be transitions from other actions to A and from Z to other actions. As a result, the probability transition matrices used in our simulation study have the constraints that $p_{i1} = 0$ for $i = 1, \dots, N$, and $p_{Nj} = 0$ for $j = 1, \dots, N-1$. To construct a probability transition matrix, we only need to specify the elements in its upper right $(N-1) \times (N-1)$ submatrix. Given a transition matrix \mathbf{P} , we start a sequence with A and generate all subsequent actions according to \mathbf{P} until Z appears.

Two simulation scenarios are devised in our experiments to impose latent class structures in generated response processes. In scenario I, two latent groups are formed by generating action sequences from two different Markov chains. Let \mathbf{P}_1 and \mathbf{P}_2 denote the probability transition matrices of the two chains. A set of n sequences is obtained by generating $n/2$ sequences according to \mathbf{P}_1 and the remaining $n/2$ sequences according to \mathbf{P}_2 . Both \mathbf{P}_1 and \mathbf{P}_2 are randomly generated and then fixed to generate all sets of response processes. To generate $\mathbf{P}_1 = (p_{ij}^{(1)})_{1 \leq i, j \leq N}$, we first construct an $(N-1) \times (N-1)$ matrix \mathbf{U} whose elements are independent samples from a uniform distribution on the interval $[-10, 10]$. Then the upper right $(N-1) \times (N-1)$ submatrix of \mathbf{P}_1 is computed from \mathbf{U} by

$$p_{i+1,j}^{(1)} = \frac{\exp(u_{ij})}{\sum_{l=1}^{N-1} \exp(u_{il})}, \quad i, j = 1, \dots, N-1. \quad (8)$$

The transition matrix \mathbf{P}_2 is obtained similarly.

In scenario II, half of the n action sequences in a set are generated from \mathbf{P}_1 as in scenario I. The other half are obtained by reversing the actions between A and Z in each of the generated sequences. For example, if (A, B, C, Z) is a generated sequence, then the corresponding reversed sequence is (A, C, B, Z). The two latent groups formed in this scenario are more subtle than those in scenario I, as a sequence and its reversed version cannot be distinguished by marginal counts of actions in \mathcal{A} .

We consider three choices of n : 500, 1,000, and 2,000. One hundred sets of action sequences are generated for each simulation scenario and each choice of n . Both LSTM

and GRU are considered for the recurrent unit in the autoencoder. For each choice of the recurrent unit, the number of features to be extracted is chosen from $\{10, 20, 30, 40, 50\}$ by five-fold cross-validation. Once K is selected, Procedure 2 is applied to each data set. The (Kingma & Ba, 2015) optimizer optimizer is used to train action sequence autoencoders with initial step size 10^{-3} . The training algorithm is run for 50 epochs with validation-based early stopping and 10% of the processes are randomly sampled as the validation set.

We investigate the ability of the extracted features to preserve the information in action sequences by examining their performance in reconstructing variables derived from action sequences. The variables to be reconstructed are indicators of the appearance of an action or an action pair in a sequence. Rare actions and action pairs that appear fewer than $0.05n$ times in a data set are not taken into consideration. We model the relationship between the indicators and the extracted features through logistic regression. For each data set, n sequences are split into training and test sets in the ratio of 4:1. A logistic regression model is estimated for each indicator on the training set and its prediction performance is evaluated on the test set by the proportion of correct predictions, that is, prediction accuracy. The average prediction accuracy over all the indicators considered is recorded for each data set and each choice of the recurrent unit.

To study how well the extracted features reveal the latent group structures in response processes, we build a logistic regression model to classify the action sequences according to the extracted features. The training and test sets are split similarly as before and the prediction accuracy on the test set is recorded for evaluation.

3.2. Results

Table 1 reports the results of our simulation study. A few observations can be made from Table 1. First, the accuracy for reconstructing the appearance of actions and action pairs is high in both simulation scenarios, indicating that the extracted features preserve a significant amount of information in the original action sequences. The reconstruction accuracy is slightly improved as n increases. Including more action sequences can provide more information for estimating the autoencoder in step 1 of Procedure 2, thus producing better features. A larger sample size can also lead to a better fit of the logistic models that relate features to derived variables. Both effects contribute to the improvement of action and action pair reconstruction.

Second, in both simulation scenarios, the extracted features can distinguish the two latent groups well. In scenario I the two groups can be separated almost perfectly. Since the group difference in scenario II is more subtle, the accuracy in classifying the two

Table 1. Mean (standard deviation) of prediction accuracy in the simulation study

Scenario	n	Reconstruction accuracy		Group accuracy	
		LSTM	GRU	LSTM	GRU
I	500	0.88 (0.005)	0.87 (0.006)	0.99 (0.010)	1.00 (0.007)
	1,000	0.90 (0.003)	0.90 (0.004)	0.99 (0.005)	0.99 (0.006)
	2,000	0.91 (0.002)	0.91 (0.003)	0.99 (0.005)	0.99 (0.005)
II	500	0.88 (0.006)	0.88 (0.006)	0.86 (0.033)	0.87 (0.031)
	1,000	0.90 (0.004)	0.91 (0.005)	0.86 (0.021)	0.86 (0.021)
	2,000	0.91 (0.002)	0.92 (0.003)	0.87 (0.027)	0.87 (0.016)

groups is lower than that in scenario I, but still more than 85% of the sequences can be classified correctly. To further look at how the extracted features reveal the latent structure of action sequences, we plot two principal features for one of the data sets with 2,000 sequences for each scenario in Figure 8. The left-hand panel presents the first two principal features for scenario I. The group structure is clearly shown and the two groups can be roughly separated by a horizontal line at 0. The right-hand panel of Figure 8 displays the plot of the first and fourth principal features for scenario II. Again the two groups can be clearly separated.

Finally, the extracted features for the two choices of the recurrent unit in the action sequence autoencoder are comparable in terms of both reconstruction and group structure identification. A GRU has a simpler structure and fewer parameters than an LSTM unit with the same latent dimension. In this sense, the GRU is more efficient for our action sequence modelling.

4. Case study

4.1. Data

The process data set used in this study contains 11,464 respondents' response processes for the PSTRE items in PIAAC 2012. There are 14 PSTRE items in total. In our data, 7,620 respondents answered seven items and 3,645 respondents answered all 14 items. For each of the 14 items, there are around 7,500 respondents. For each respondent–item pair, both the response process (action sequence) and the final response outcome were recorded. The original final outcomes for some items are polytomous. We simplify them into binary outcomes with the fully corrected responses labelled as 1 and all others as 0.

The 14 PSTRE items in PIAAC 2012 vary in content, task complexity and difficulty. Some basic descriptive statistics for the items are summarized in Table 2, where n denotes the number of respondents, N is the number of possible actions, \bar{T} stands for the average sequence length and Correct % is the percentage of correct responses. There are three types of interaction environments: email client, spreadsheet, and web browser. Some items such as U01a and U01b have a single environment, while some items such as

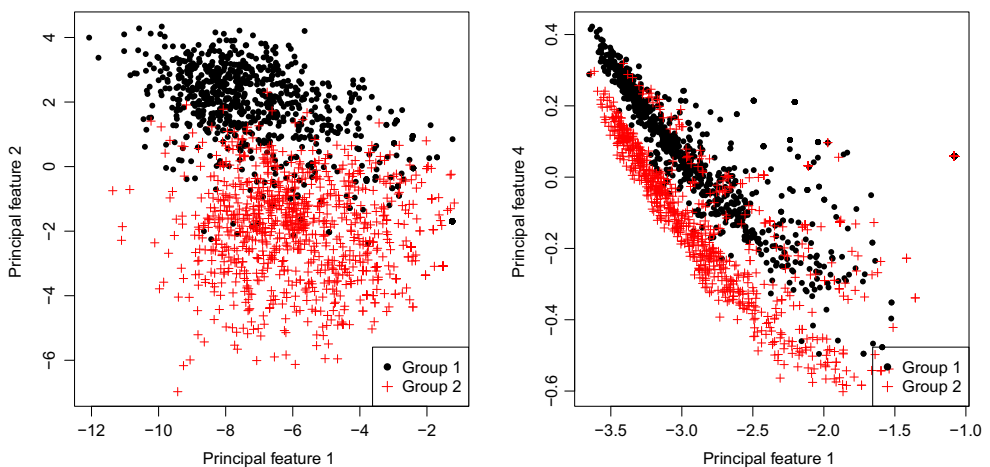


Figure 8. Left: scatterplot of the first two principal features for one data set of 2,000 sequences generated under scenario I. Right: scatterplot of principal features 1 and 4 for one data set of 2,000 sequences generated under scenario II. [Colour figure can be viewed at wileyonlinelibrary.com]

U02 and U23 involve multiple environments. U06a is the simplest item in terms of number of possible actions and average response length, but only about a quarter of the participants answered it correctly. Items U02 and U04a are the most difficult items, and only around 10% of the respondents correctly completed the given tasks. The tasks in these two items are relatively complicated: there are a few hundred possible actions and more than 40 actions are needed to finish the task. With the wide item variety, manually extracting important features of process data based on experts' understanding of the items is time-consuming, while the proposed automatic method can be easily applied to all these items.

4.2. Features and their interpretations

We extract features from the response processes for each of the 14 items using the proposed procedure. The number of features is chosen from {10, 20, . . . , 100} by five-fold cross-validation. The Adam algorithm is used to optimize the object function in step 1 of Procedure 2. The initial step size is set to 10^{-4} . The algorithm is run for 100 epochs with validation-based early stopping, where 10% of the processes are randomly sampled to form the validation set for each item.

Although the proposed method does not utilize the meaning of the actions for feature extraction, many of the principal features, especially the first several components, often have practical meaning. To search for the interpretation of a feature, we first sort the action sequences according to the value of the feature. Next, we examine the action sequences with low and high feature values respectively and then construct a variable that distinguishes the high-feature-value and low-feature-value groups. Finally, we quantitatively confirm the interpretation by computing the correlation between the constructed variable and the feature. A higher correlation indicates a more confident interpretation. We use the first feature of U01a as an example to demonstrate how the procedure works. After sorting the action sequences according to the feature value, we find that the action sequences with the lowest feature values are often very short and that the action

Table 2. Descriptive statistics for PIAAC PSTRE items

ID	Description	<i>n</i>	<i>N</i>	\bar{T}	Correct %
U01a	Party invitations – can/cannot come	7,620	207	24.8	54.5
U01b	Party invitations – accommodations	7,670	249	52.9	49.3
U02	Meeting rooms	7,537	328	54.1	12.8
U03a	CD tally	7,613	280	13.7	37.9
U04a	Class attendance	7,617	986	44.3	11.9
U06a	Sprained ankle – site evaluation table	7,622	47	10.8	26.4
U06b	Sprained ankle – reliable/trustworthy site	7,612	98	16.0	52.3
U07	Digital photography book purchase	7,549	125	18.6	46.0
U11b	Locate e-mail – file 3 e-mails	7,528	236	30.9	20.1
U16	Reply all	7,531	257	96.9	57.0
U19a	Club membership – member ID	7,556	373	26.9	69.4
U19b	Club membership – eligibility for club president	7,558	458	21.3	46.3
U21	Tickets	7,606	252	23.4	38.2
U23	Lamp return	7,540	303	28.6	34.3

Note. *n* = number of respondents; *N* = number of possible actions; \bar{T} = average sequence length; Correct % = percentage of correct responses.

sequences with the highest feature values are often long. This observation suggests that the first feature of U01a is related to the length of a response process. Next, we compute the logarithm of the sequence length for each process and find that its correlation with the feature is .90, which confirms our conjecture. Using this procedure, we examined the first five principal features of each item. A partial list of the feature interpretations we found is given in Table 3.

The first or second principal feature of each item usually corresponds to respondents' attentiveness. An inattentive respondent tends to move to the next item without meaningful interactions with the computer environment. In contrast, an attentive respondent typically tries to understand and to complete the task by exploring the environment. Thus attentiveness in the response process may be reflected in the length of the process. We call the principal feature that has the largest absolute correlation with the logarithm of the process length the attentiveness feature. In our case, the attentiveness feature is the second principal feature for item U06a and the first for all other items. For all the items, the absolute correlation between the attentiveness feature and the logarithm of sequence length is greater than .85. To make a higher attentiveness feature correspond to a more attentive respondent, we modify the attentiveness features by multiplying each of them by the sign of their correlation with the logarithm of process length. For a given pair of items, we select respondents who responded to both items and calculate the correlation between the two modified attentiveness features. These correlations are all positive and range from .30 to .70, implying that the respondents who are inattentive on one item tend to be inattentive on the other item.

The feature space of the respondents with correct responses is usually very different from that of the respondents with incorrect responses. As an illustration, the first two principal features of U01b are plotted in Figure 9 for the two groups of respondents separately. Note that we perform principal component analysis before dividing the respondents into correct and incorrect groups. Thus the feature spaces presented in the two panels can be compared. As the figure shows, the distributions of the principal components are very different for these two groups. The non-oval shape of the clouds suggests that the feature space is highly nonlinear. Furthermore, the variance of the

Table 3. A partial list of feature interpretations by interface type. The last column gives an example of the features with the interpretation. The correlation between the feature and the constructed variable is included in parentheses

Interface type	Interpretation	Example
Email client	Viewing emails and folders	U01a-2 (.71)
	Moving emails	U01a-3 (.72)
	Creating new folders	U01b-2 (.81)
Spreadsheet	Using sort	U03a-2 (.68)
	Using search	U19a-2 (.72)
Web browser	Clicking relevant links	U06b-2 (.84)
	Clicking irrelevant links	U23-2 (.77)
All interfaces	Sequence length	Often the first feature (.85 to .97)
	Using actions related to the task	U23-4 (.75)
	Switching working environments	U04a-2 (.78)
	Selecting answers	U06b-4 (.65)

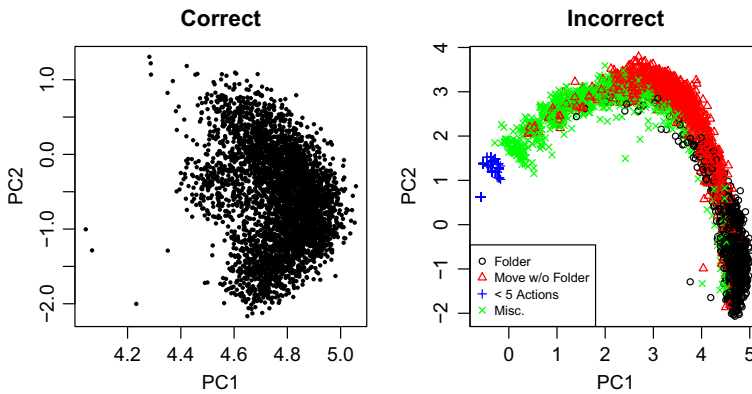


Figure 9. Scatterplots of the first two principal features of U01b stratified by response outcome [Colour figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com)]

incorrect group is much larger than that of the correct group suggesting that the incorrect group is much more heterogeneous. The main reason is that there are more ways to solve the problem incorrectly than correctly. Item U01b requires the respondents to create a new folder and to move some emails to the new folder. Among the incorrect respondents, some moved emails but did not create a new folder while some created a new folder but did not move the emails correctly. There are also some respondents who did not respond seriously: they took fewer than five actions before moving to the next item. As shown in the right-hand panel of Figure 9, respondents with similar behaviours are located close to each other in the feature space.

4.3. Reconstruction of derived variables

We demonstrate in this subsection that the features extracted from the proposed procedure retain a substantial amount of information on the response processes. To be more specific, we show that various variables directly derived from the processes can be reconstructed by the extracted features.

We define a derived variable as a binary variable indicating whether an action or a combination of actions appears in the process. For example, whether the first dropdown menu is set to ‘Photography’ is a derived variable of the item described in the introduction. The binary response outcome is also a derived variable since it is entirely determined by the response process. In our data, 93 derived variables, including 14 item response outcomes, are considered.

Similarly to the simulation study, we examine how well the derived variables can be reconstructed through a prediction procedure. We use logistic regression to model the relation between a derived variable and the principal features of the corresponding item. For each derived variable, 80% of the respondents are randomly sampled to form the training set and the remaining 20% form the test set. We fit the model on the training set and predict the derived variable for each respondent in the test set. Specifically, the derived variable is predicted as 1 if the fitted probability is greater than .5, and 0 otherwise. Prediction accuracy on the entire test set is calculated for evaluation.

As shown in Table 4, for all the derived variables, the prediction accuracy is higher than 0.80. For 75 out of 93 variables, the accuracy is higher than .90. Thirty-five variables are predicted nearly perfectly (prediction accuracy greater than .975). These results

Table 4. Distribution of the out-of-sample prediction accuracy for 93 derived variables

Accuracy	(.80, .85]	(.85, .90]	(.90, .95]	(.95, .975]	(.975, 1.00]
Counts	5	13	28	12	35

manifest that the extracted features carry a significant amount of information in the action sequences. We demonstrate in the remaining subsections that the extracted features are useful for assessing respondents' competency and behaviours.

4.4. Variable prediction based on a single item

The item responses (both outcome and process) for an item reflect respondents' latent traits, which affect their overall performance in a test. Therefore, each item response should have some power to predict the responses on other items and overall competency. Process data contain more detailed information on respondents' behaviours than a single binary outcome. We expect that prediction based on the response process is more accurate than that solely based on the final outcome. In this subsection we assess information in the response processes of a single item via the prediction of the binary response outcomes of other items as well as numeracy and literacy scores.

Given the final outcome and the response process for an item, say item j , we model their relation with the predicted variable by a generalized linear model

$$g(\mu) = \boldsymbol{\eta}_j^T \boldsymbol{\beta}, \quad (9)$$

where μ is the expectation of the predicted variable, g is the link function, $\boldsymbol{\eta}_j$ is a vector of covariates related to item j , which will be specified later, and $\boldsymbol{\beta}$ is the coefficient vector. If the predicted variable is the binary outcome of item j' , $g(\mu) = \log(\mu/(1 - \mu))$ is the logit link and μ is the probability of answering the item correctly. If the predicted variable is the literacy or numeracy score, g is the identity link and (9) becomes linear regression.

Let z_j denote the binary outcome and let $\boldsymbol{\theta}_j$ denote the features extracted from the response process of item j . We consider two choices of $\boldsymbol{\eta}_j$ for a given predicted variable, $\boldsymbol{\eta}_j = (1, z_j)^T$ and $\boldsymbol{\eta}_j = (1, z_j, \boldsymbol{\theta}_j^T, z_j \boldsymbol{\theta}_j^T)^T$. The first choice only uses the binary outcome for prediction. The second uses both the outcome and the response process. We call the models with these two choices of covariates the baseline model and the process model, respectively. It turns out that the information in the baseline model is very limited, especially when the correct rate of item j is close to 0 or 1.

For a given predicted variable, two thirds of the available respondents are randomly sampled to form the training set. The remaining one third are evenly split to form the validation and the test set. Both the baseline model and the process model are fitted on the training set. We add L_2 penalties on the coefficient vector $\boldsymbol{\beta}$ in the process model to avoid overfitting. The penalty parameter is chosen by examining the prediction performance of the resulting model on the validation set. Specifically, a process model is fitted for each candidate value of the penalty parameter. The one that produces the best prediction performance on the validation set is chosen to obtain the final process model for comparison with the baseline model. The evaluation criterion is prediction accuracy for outcome prediction and out-of-sample R^2 (OSR²) for score prediction. OSR² is defined to

be the square of the Pearson correlation between the predicted and true values. A higher OSR^2 indicates better prediction performance.

4.4.1. Outcome prediction results

Figure 10 presents the results of outcome prediction. The plot in the left-hand panel gives the improvement in the out-of-sample prediction accuracy of the process model over that of the baseline model for all item pairs. The entry in the i th row and the j th column gives the result for predicting item j by item i . For many item pairs, adding the features extracted from process data improves the prediction. To further examine the improvements, for the task of predicting the outcome of item j' by item j , we calculate the prediction accuracy separately for the respondents who answered item j correctly and for those who answered incorrectly. The improvements for these two groups are plotted respectively in the middle and right-hand panels of Figure 10. The improvement is more significant for the incorrect group in both the number of item pairs that have improvement and the magnitude of the improvement. As mentioned previously, the incorrect response processes are more diverse than the correct ones, thus providing more information about the respondents. Misunderstanding the item requirements and lack of basic computer skills often lead to an incorrect response. Carelessness and inattentiveness are also possible causes of incorrect answers. These differences can be reflected in the extracted features as illustrated in Figure 9. Therefore, including these features in the model helps the prediction more for the incorrect group than for the correct group.

4.4.2. Numeracy and literacy prediction results

Numeracy and literacy score prediction results are displayed in Figure 11. In the left-hand panel, we plot the OSR^2 of the process model against that of the baseline model. For both literacy and numeracy, regardless of the item used for prediction, the process model produces a higher OSR^2 than the baseline model. Although the PSTRE items are not designed to measure these two competencies, the response processes are helpful for predicting the scores. To further examine the results, for each item-score pair, we again group the respondents according to their item response outcome and calculate the OSR^2 of the process model for the two groups separately. The OSR^2 for the incorrect group is plotted against that for the correct group in the right-hand panel of Figure 11. Similar to

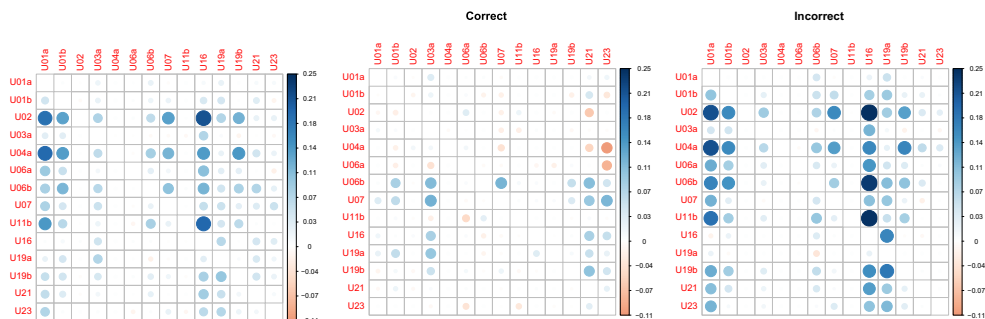


Figure 10. Difference of the cross-item outcome prediction accuracy for the process model and the baseline model. [Colour figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com)]

the outcome prediction, the prediction performance for the incorrect group is usually much better than that for the correct group since action sequences corresponding to incorrect answers are often more diverse and informative than those corresponding to correct answers.

4.5. Prediction based on multiple items

In this subsection we examine how the improvement in prediction performance from process data aggregates as more items are incorporated in the prediction. The variables of interest are age, gender, and literacy and numeracy scores.

We only consider the 3,645 respondents who responded to all 14 PSTRE items in this experiment. The respondents are randomly split into training, validation, and test sets. The sizes of the three sets are 2,645, 500, and 500, respectively. The split is fixed for estimating and evaluating all models in this experiment.

We still consider model (9) for prediction. The logit link (i.e. logistic regression) is used for gender prediction, and linear regression for other variables. In this experiment, the covariate vector η incorporates information from multiple items. Given a predicted variable and a set of available items, a baseline model and a process model are considered for each variable. For the baseline model, η consists of only the final outcomes, while for the process model it also includes the first 20 principal features for each available item. Let $S_m = \{j_1, \dots, j_m\}$ denote the set of the indices for available items. The predictor for the baseline model is $\eta = (1, z_{j_1}, \dots, z_{j_m})^T$ and that of the process model is $\eta = (1, z_{j_1}, \dots, z_{j_m}, \theta_{j_1}, \dots, \theta_{j_m})^T$, where $\theta_j \in \mathbb{R}^{50}$ is the first 50 principal features for item j . We start with an empty item set and add one item to the set at a time. That is, for a given predicted variable, a sequence of 14 baseline models and 14 process models are fitted. The order of items being added to the model is determined by forward Akaike information criterion (AIC) selection for the 14 outcomes on the training set. Specifically, for a given m , S_m contains the items whose outcomes are the first m variables selected by forward AIC selection among all 14 outcomes. We use prediction accuracy as the evaluation criterion for gender prediction and OSR^2 for other variables.

Recently, n -grams, which are contiguous sequences of n actions appearing in a set of action sequences, have been used to look for action patterns that are closely related to

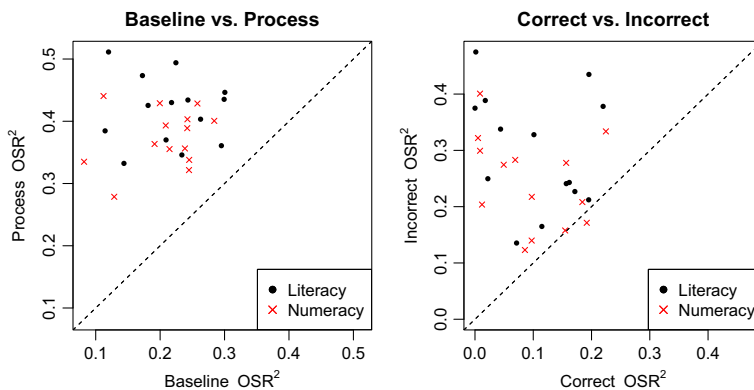


Figure 11. Left: OSR^2 of the baseline and process model on the test set. Right: OSR^2 of the process model for the correct and incorrect groups. [Colour figure can be viewed at wileyonlinelibrary.com]

item responses (He & von Davier, 2016). We extract all unigrams (actions) and bigrams (action pairs) that appear in more than 1% of the response processes and screen them by the method proposed in (He & von Davier, 2016) for each item. The frequencies of the selected n -grams in an action sequence can also be seen as features of response processes. To compare the prediction performance of the autoencoder features and n -gram features, we repeat the above experiment of predicting literacy and numeracy scores with θ_j replaced by the n -gram features. Other settings are the same as before.

4.5.1. Numeracy and literacy prediction results

In Figure 12, the OSR^2 for predicting literacy and numeracy scores is plotted against the number of available items. For both the process model and the baseline model, the prediction of numeracy and literacy improves as responses are available for more items. Regardless of the number of available items, the process model outperforms the baseline model in both literacy and numeracy score predictions, although the difference becomes smaller as the number of available items increases. Notice that the OSR^2 of the process model based on only two items roughly equals the OSR^2 of the baseline model based on four items. These results imply that properly incorporating process data in data analysis can exploit the information in items more efficiently and that the incorporation is especially beneficial when a small number of items are available. Figure 12 also shows that autoencoder features can achieve higher OSR^2 than n -gram features, suggesting that more information related to the two scores is contained in the autoencoder features.

The PSTRE item responses have some power to predict literacy and numeracy. This is not surprising as literacy and numeracy are related to the understanding of the PSTRE item description and material. In our case study, PSTRE items are more related to literacy than numeracy: the OSR^2 for literacy score models is usually higher than that for the corresponding numeracy score model. The number of items needed in the process model to achieve a similar OSR^2 to that obtained in the baseline model with all 14 items is five for literacy and eight for numeracy.

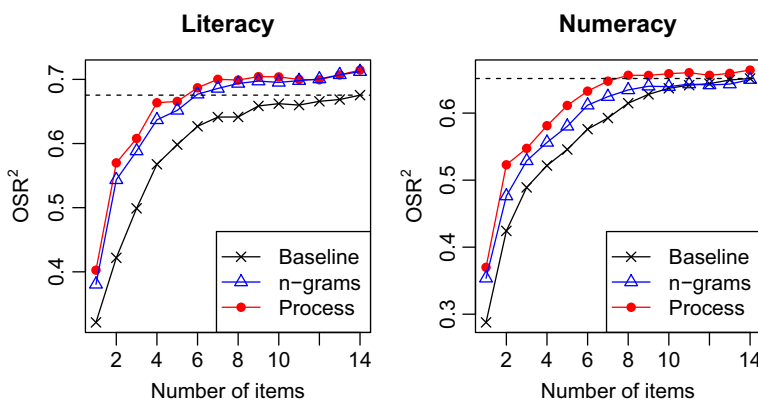


Figure 12. OSR^2 of the baseline and process model with various numbers of items. [Colour figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com)]

4.5.2. Background variable prediction results

Figure 13 presents the results for predicting age and gender. Adding more items to the baseline model barely improves the OSR^2 for predicting age, while in the process model the OSR^2 increases as more items are included and it is about twice as high as that of the baseline model when all 14 items are included. These results show that respondents at different ages behave differently in solving PSTRE items and that response processes can reveal the differences significantly better than final outcomes. A closer examination of the action sequences shows that younger respondents are more likely to use drag and drop actions to move emails while older respondents tend to move emails by using email menus (left-hand panel of Figure 14). Also, older respondents are less likely to use 'Search' in spreadsheet environments (right-hand panel of Figure 14).

As for gender, the highest prediction accuracy of the baseline models is .55, which is only 0.02 higher than the proportion of female respondents in the test set. The prediction accuracy of the process model is almost always higher than that of the corresponding baseline model and it can be as high as .63. These observations imply that female and male respondents have similar performance in PSTRE items in terms of final outcomes, but there are some differences in their response processes. In our data, male respondents are more likely to use sorting tools in spreadsheet environments, as shown in Table 5. The p -value for the χ^2 test of independence between gender and whether 'Sort' is used is less than 10^{-6} for the three items with spreadsheet environments.

5. Concluding remarks

In this paper we have presented a method to extract latent features from response processes. The key step of the method is to train an action sequence autoencoder for a set of response processes. We showed through a case study of the process data for PSTRE items in PIAAC 2012 that the extracted features improve the prediction of response outcomes, and literacy and numeracy scores.

In Sections 4.4 and 4.5 we show that features extracted from action sequences can be used for predicting variables of interest. One can also build neural networks to predict a response variable directly from an action sequence. Such neural networks are often a combination of an RNN and a feed-forward neural network. If variables are predicted in this way, we may possibly need to fit separate models for each response variable and each

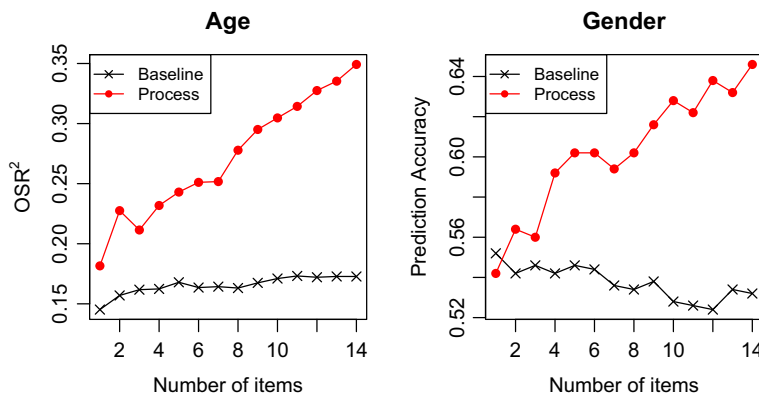


Figure 13. Prediction results for age and gender. [Colour figure can be viewed at wileyonlinelibrary.com]

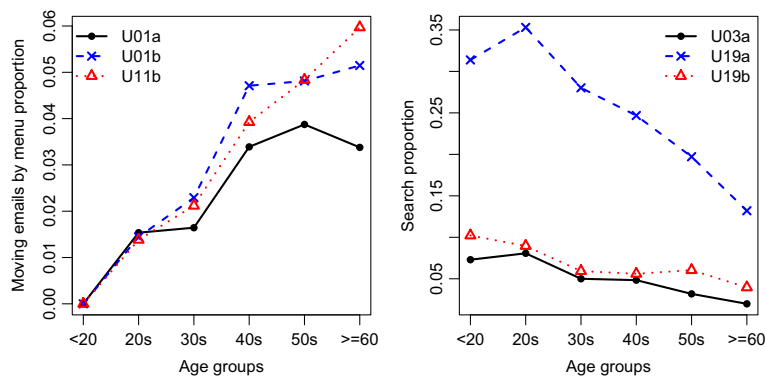


Figure 14. Left: Proportion of respondents moving emails by menu in different age groups. Right: Proportion of respondents using ‘Search’ in different age groups. [Colour figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com)]

Table 5. Contingency tables of gender and whether ‘Sort’ is used in U03a, U19a and U19b

Gender	U03a		U19a		U19b	
	Yes	No	Yes	No	Yes	No
Male	418	1,238	365	1,291	661	995
Female	359	1,630	311	1,678	564	1,425

of the models involves RNNs. Fitting models with RNN components is generally computationally expensive because of its recurrent structure. With the feature extraction method, we only need to fit a single model (the action sequence autoencoder) that involves RNNs, and then fit a (generalized) linear model or a feed-forward neural network for each variable of interest. As the results presented in Appendix B.3 show, the two approaches are often comparable in terms of prediction performance. In fact, the approach without feature extraction could produce much worse results although it seems to be a more efficient approach intuitively since a neural network is trained for each variable. This is because a large amount of noise is contained in the action sequences. Performing dimension reduction prior to supervised learning tasks can improve prediction performance, especially in the presence of a large amount of noise. Our feature extraction is a dimension reduction procedure that filters out the noise in action sequences. Thus less noise is contained in the feature-based (generalized) linear models than in training the neural network with action sequences as input.

We use five-fold cross-validation to select an appropriate number of features to extract in both simulation and real data analysis. As previously mentioned, training RNN-based neural networks is often computationally expensive. Performing m -fold cross-validation requires training m models with different inputs, which could be computationally intensive. As the additional data analysis in Appendix B.1 shows, overselecting the feature dimension is often less harmful in terms of prediction accuracy than underselecting the dimension. There could be significant information loss if too few features are extracted, while if too many features are extracted, appropriate dimension reduction or variable selection methods can be used to avoid incorporating too much noise in the subsequent

analysis. Therefore, if computational resources are limited, we suggest simply choosing a reasonably large K for feature extraction without running m -fold cross-validation. Based on our experience, $K = 100$ is often a good choice for response processes as complex as PIAAC data.

Generally speaking, training neural networks such as the action sequence autoencoder requires a large amount of data. In Appendix B.2 we include additional data analysis results on the effect of sample size on the extracted features. The results suggest that, in our case, at least a few hundred response processes are needed for the extracted features to produce reasonable prediction results. We want to point out that there is no simple answer to the question that how many samples are necessary to train the action sequence autoencoder, as it depends on many factors such as the number of features to be extracted and the complexity of the response processes.

Computer log files of interactive items often include time-stamps of actions. The time elapsed between two consecutive actions may also provide extra information about respondents and can be useful in educational and cognitive assessments. The current action sequence autoencoder does not make use of this information. Further study on incorporating time information in the analysis of process data is a possible future direction.

Acknowledgements

This work is supported by National Science Foundation grants SES-1826540 and IIS-1633360. The authors would like to thank the Educational Testing Service and Qiwei He for providing the data, and Hok Kan Ling for cleaning it.

Conflicts of interest

All authors declare no conflict of interest.

Author contributions

Xueying Tang (Formal analysis; Methodology; Software; Writing – original draft) Zhi Wang (Formal analysis; Investigation; Writing – original draft) Jingchen Liu (Funding acquisition; Methodology; Supervision; Writing – review & editing) Zhiliang Ying (Funding acquisition; Supervision; Writing – review & editing).

Data availability statement

The data that support the findings of this study are available from the Organisation for Economic Co-operation and Development (OECD). Restrictions apply to the availability of these data, which were used under licence for this study. Data are available from <http://piaacgateway.com/datasets/> with the permission of the OECD.

References

- Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3, 1137–1155.

- Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), 157–166. <https://doi.org/10.1109/72.279181>
- Bosch, N., & Paquette, L. (2017). *Unsupervised deep autoencoders for feature extraction with educational data*. In Deep Learning with Educational Data Workshop at the 10th International Conference on Educational Data Mining
- Brockwell, P. J., & Davis, R. A. (2016). *Introduction to time series and forecasting*. Cham, Switzerland: Springer.
- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (pp. 1724–1734). Stroudsburg, PA: Association for Computational Linguistics. <https://doi.org/10.3115/v1/D14-1179>
- Deng, L., Seltzer, M. L., Yu, D., Acero, A., Mohamed, A., & Hinton, G. (2010). Binary coding of speech spectrograms using a deep auto-encoder. In *Interspeech 2010*. Makuhari, Japan: International Speech Communication Association. <https://doi.org/10.1.1.185.1908>
- Ding, M., Yang, K., Yeung, D.-Y., & Pong, T.-C. (2019). Effective feature learning with unsupervised learning for improving the predictive models in massive open online courses. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge* (pp. 135–144). New York, NY: ACM. <https://doi.org/10.1145/3303772.3303795>
- Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, 2121–2159.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge, MA: MIT Press.
- Greiff, S., Niepel, C., Scherer, R., & Martin, R. (2016). Understanding students' performance in a computer-based assessment of complex problem solving: An analysis of behavioral data from computer-generated log files. *Computers in Human Behavior*, 61, 36–46. <https://doi.org/10.1016/j.chb.2016.02.095>
- He, Q., & von Davier, M. (2016). Analyzing process data from problem-solving items with n-grams: Insights from a computer-based large-scale assessment. In Y. Rosen, S. Ferrara & M. Mosharraf (Eds.), *Handbook of research on technology tools for real-world skill development* (pp. 749–776). Hershey, PA: Information Science Reference. <https://doi.org/10.4018/978-1-4666-9441-5.ch029>
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313, 504–507. <https://doi.org/10.1126/science.1127647>
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9, 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.
- Klein Entink, R. H., Fox, J.-P., & van der Linden, W. J. (2009). A multivariate multilevel approach to the modeling of accuracy and speed of test takers. *Psychometrika*, 74(1), 21. <https://doi.org/10.1007/s11336-008-9075-y>
- Kraft, P., Jain, H., & Rush, A. M. (2016). An embedding model for predicting roll-call votes. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 2066–2070). Stroudsburg, PA: Association for Computational Linguistics. <https://doi.org/10.18653/v1/D16-1221>
- Kroehne, U., & Goldhammer, F. (2018). How to conceptualize, represent, and analyze log data from technology-based assessments? A generic framework and an application to questionnaire items. *Behaviormetrika*, 45, 527–563. <https://doi.org/10.1007/s41237-018-0063-y>
- Li, J., Luong, T., & Jurafsky, D. (2015). A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (Vol. 1, pp. 1106–1115). Stroudsburg, PA: Association for Computational Linguistics.

- Lord, F. M. (1980). *Applications of item response theory to practical testing problems*. New York, NY: Routledge.
- Lord, F. M., & Novick, M. R. (1968). *Statistical theories of mental test scores*. Reading, MA: Addison-Wesley.
- Lu, X., Tsao, Y., Matsuda, S., & Hori, C. (2013). Speech enhancement based on deep denoising autoencoder. In *Interspeech* (pp. 436–440). Red Hook, NY: Curran Associates Inc.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* 26 (pp. 3111–3119). Red Hook, NY: Curran Associates Inc.
- OECD. (2017). *The nature of problem solving: Using research to inspire 21st century learning*. Paris, France: OECD Publishing.
- Patterson, J., & Gibson, A. (2017). *Deep learning: A practitioner's approach*. Sebastopol, CA: O'Reilly Media.
- Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., & Sohl-Dickstein, J. (2015). Deep knowledge tracing. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama & R. Garnett (Eds.), *Advances in neural information processing systems* 28 (pp. 505–513). Cambridge, MA: MIT Press.
- Prechelt, L. (2012). Early stopping but when? In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade* (2nd ed., pp. 53–67). Berlin, Germany: Springer. https://doi.org/10.1007/3-540-49430-8_3
- Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3), 400–407. <https://doi.org/10.1214/aoms/1177729586>
- Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society, Series B*, 36(2), 111–133. <https://doi.org/10.1111/j.2517-6161.1974.tb00994.x>
- Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 1096–1103). New York, NY: ACM. <https://doi.org/10.1145/1390156.1390294>
- Wang, L., Sy, A., Liu, L., & Piech, C. (2017). Deep knowledge tracing on programming exercises. In *Proceedings of the fourth (2017) ACM conference on learning @ scale* (pp. 201–204). New York, NY: ACM. <https://doi.org/10.1145/3051457.3053985>
- Wang, S., Zhang, S., Douglas, J., & Culpepper, S. (2018). Using response times to assess learning progress: A joint model for responses and response times. *Measurement: Interdisciplinary Research and Perspectives*, 16(1), 45–58. <https://doi.org/10.1080/15366367.2018.1435105>
- Yousefi-Azar, M., Varadharajan, V., Hamey, L., & Tupakula, U. (2017). Autoencoderbased feature learning for cyber security applications. In *2017 International Joint Conference on Neural Networks* (pp. 3854–3861). Piscataway, NJ: IEEE. <https://doi.org/10.1109/IJCNN.2017.7966342>
- Zeiler, M. D. (2012). *Adadelta: An adaptive learning rate method*, Preprint, arXiv:1212.5701.
- Zhan, P., Jiao, H., & Liao, D. (2018). Cognitive diagnosis modelling incorporating item response times. *British Journal of Mathematical and Statistical Psychology*, 71(2), 262–286. <https://doi.org/10.1111/bmsp.12114>

Received 3 September 2019; revised version received 30 January 2020

Appendix:

A. Structures of the LSTM unit and the GRU

A.1. LSTM unit

In an LSTM unit, the hidden state \mathbf{m}_t consists of two components, \mathbf{c}_t and \mathbf{h}_t , storing long-term and short-term memory, respectively, of the sequence up to the current step. The long-term memory \mathbf{c}_t is often called the cell state of the unit. The LSTM unit computes the hidden states and outputs in time-step t as follows:

$$\begin{aligned} \mathbf{z}_t &= \sigma(\mathbf{q}_1 + \mathbf{W}_1 \mathbf{x}_t + \mathbf{U}_1 \mathbf{h}_{t-1}), \\ \mathbf{r}_t &= \sigma(\mathbf{q}_2 + \mathbf{W}_2 \mathbf{x}_t + \mathbf{U}_2 \mathbf{h}_{t-1}), \\ \tilde{\mathbf{c}}_t &= \tanh(\mathbf{q}_3 + \mathbf{W}_3 \mathbf{x}_t + \mathbf{U}_3 \mathbf{h}_{t-1}), \\ \mathbf{c}_t &= \mathbf{z}_t * \mathbf{c}_{t-1} + \mathbf{r}_t * \tilde{\mathbf{c}}_t, \\ \mathbf{v}_t &= \sigma(\mathbf{q}_4 + \mathbf{W}_4 \mathbf{x}_t + \mathbf{U}_4 \mathbf{h}_{t-1}), \\ \mathbf{h}_t &= \mathbf{v}_t * \tanh(\mathbf{c}_t), \\ \mathbf{y}_t &= \mathbf{h}_t, \end{aligned} \quad (10)$$

where $*$ denotes elementwise multiplication and $\mathbf{q}_i, \mathbf{W}_i, \mathbf{U}_i, i = 1, 2, 3, 4$, are parameters. Both $\sigma(x) = 1/\{1 + \exp(-x)\}$ and $\tanh(x) = \{\exp(x) - \exp(-x)\}/\{\exp(x) + \exp(-x)\}$ are elementwise activation functions. The relationship among the components in equation (11) is illustrated in Figure A1. Receiving the hidden state from the previous step $\mathbf{m}_{t-1} = (\mathbf{c}_{t-1}, \mathbf{h}_{t-1})$ and the current input \mathbf{x}_t , the LSTM unit forgets obsolete information in the previous long-term memory and combines the remainder ($\mathbf{z}_t * \mathbf{c}_{t-1}$) with the useful information ($\mathbf{r}_t * \tilde{\mathbf{c}}_t$) from the new input and previous short-term memory to form the updated long-term memory \mathbf{c}_t . As \mathbf{z}_t controls what should be forgotten in the long-term memory, it is often called the forget gate. The candidate cell state $\tilde{\mathbf{c}}_t$ summarizes the information in the previous short-term memory \mathbf{h}_{t-1} and the new input \mathbf{x}_t . The so-called input gate, \mathbf{r}_t , determines which part of this new information should be added into the long-term memory. Once the long-term memory is updated, the LSTM unit selects the information that is useful in the short term from the updated long-term memory with the help of the output gate \mathbf{v}_t . The selected information then forms the updated short-term memory \mathbf{h}_t and is used as the output \mathbf{y}_t of the unit.

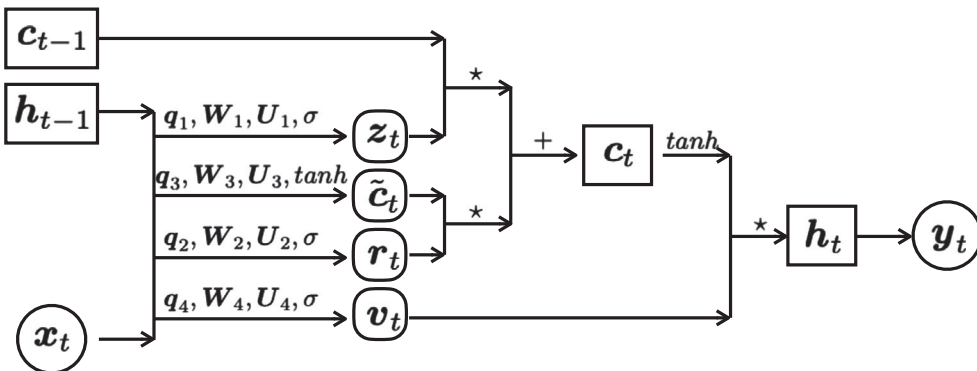


Figure A1. The structure of the LSTM unit.

A.2. GRU

Using the notation in Section 2.2, the GRU computes the hidden states and outputs at time-step t as follows:

$$\begin{aligned} z_t &= \sigma(q_1 + W_1 x_t + U_1 m_{t-1}), \\ r_t &= \sigma(q_2 + W_2 x_t + U_2 m_{t-1}), \\ \tilde{m}_t &= \tanh(q_3 + W_3 x_t + U_3 (r_t * m_{t-1})), \\ m_t &= (1 - z_t) * m_{t-1} + z_t * \tilde{m}_t, \\ y_t &= m_t, \end{aligned} \quad (11)$$

where $*$ denotes elementwise multiplication and $q_i, W_i, U_i, i = 1, 2, 3$, are parameters. Both $\sigma(x) = 1/\{1 + \exp(-x)\}$ and $\tanh(x) = \{\exp(x) - \exp(-x)\}/\{\exp(x) + \exp(-x)\}$ are elementwise activation functions. The structure given in equation (11) is illustrated in Figure A2. The procedure of updating the hidden state in the GRU is simpler than that in the LSTM unit. In the GRU, a candidate hidden state \tilde{m}_t is first computed by combining the information in the input x_t and the previous hidden state m_{t-1} . Notice that not all the information in m_{t-1} is used to construct \tilde{m}_t . Only the relevant information selected by the reset gate r_t is incorporated. The updated hidden state m_t is a convex combination of the previous hidden state m_{t-1} and the candidate hidden state \tilde{m}_t . The weight z_t is called the update gate as it controls how much new information (\tilde{m}_t) should be integrated in m_t . The output of the GRU is the same as the updated hidden state.

B. Additional data analysis results

B.1. Effect of number of extracted features

In this experiment, we use the action sequences of the 3,645 respondents who answered all 14 items in PIAAC 2012 to investigate the effect of the number of extracted features K on the performance of the extracted features. Two aspects of the extracted features are examined in this experiment: derived variable reconstruction accuracy; and score prediction performance.

We consider several choices of K : 10, 20, 40, 80, 100, 150 and 200. For each choice, we extract K features via Procedure 2 for each item. The optimizer, step size, and number of epochs used for training the action sequence autoencoders are the same as those described in Section 2. The 3,645 respondents are randomly split into training (3,145) and test (500) sets for the following analysis.

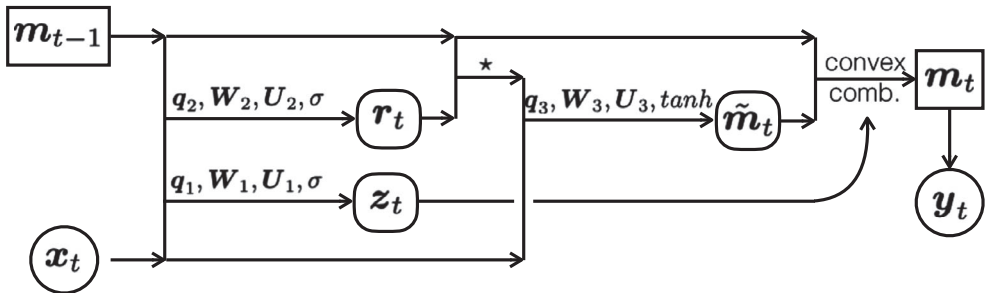


Figure A2. The structure of the GRU.

All the derived variables considered in Section 4.3 is used here to evaluate the reconstruction accuracy. For each choice of K and each derived variable, a logistic regression model based on the features from the corresponding item is fitted on the training set. The prediction accuracy on the test set is recorded as the reconstruction accuracy of the derived variable with K features. We calculate the proportion of the 93 derived variables that can be reconstructed with accuracy above .85, .90 and .95 to examine the overall reconstruction accuracy for each choice of K . As the results presented in the left-hand panel of Figure A3 show, derived variable reconstruction accuracy increases as the number of extracted features increases from 10 to 40. Further increases in K do not lead to significant change.

To study the score prediction performance of the features, we use a linear model with the first 10 principal features and the binary responses from the 14 items as covariates to predict respondents' literacy and numeracy scores for each choice of K . The linear models are fitted on the training set and the OSR^2 (defined in Section 4.4) on the test set is recorded. To reduce the variability of the results, the above analysis is performed ten times. A different training-test split is used each time. The sizes of the sets are kept the same across the ten splits. The average of the ten OSR^2 values for each choice of K is reported in the middle and right-hand panels of Figure A3. Similarly to the reconstruction results, the score prediction improves as K increases from 10 to 40 and then stabilizes. In the two panels, the prediction results based only on the 14 binary responses are plotted as the dashed line. Regardless of the choice of K , the model based on both features and binary responses outperforms that based solely on the binary responses.

B.2. Effect of sample size

In this section we study the effect of sample size on the extracted features from the same two aspects as in Appendix B.1, namely, derived variable reconstruction and score prediction. Our experiment is again performed on the PIAAC 2012 data set containing action sequences of 14 items from 3,645 respondents. We first randomly sampled 500 respondents as the common test set for all the analyses described below. Then training

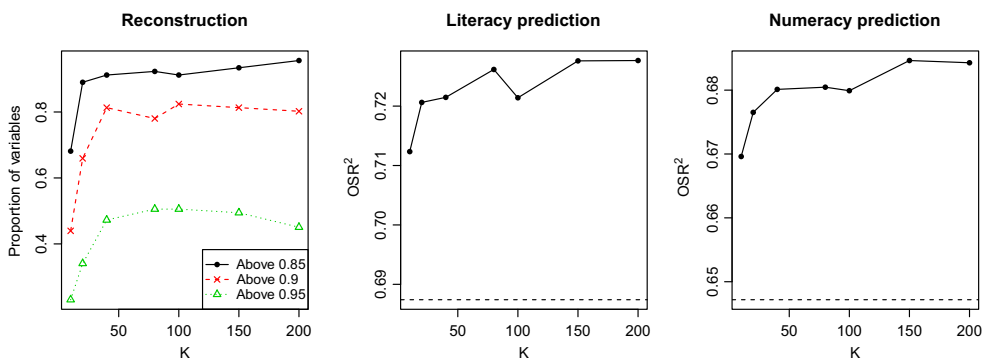


Figure A3. Reconstruction (left) and score prediction (middle and right) performance for various choices of K . The dashed lines in the middle and right-hand panels represent the results from the models based on 14 binary responses. [Colour figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com)]

sets with various sample sizes are created by subsampling the remaining 3,145 respondents with subsampling proportion $p = .1, .2, \dots, .9, 1$. Given a training set, the corresponding action sequences are used to train the action sequence autoencoder with $K = 100$ for each item. Then 100 features are extracted for the action sequences in the training set and those in the test set.

To study the performance of reconstructing derived variables, we fit a logistic regression model on the training set for each derived variable on the 100 extracted features of the corresponding item and record the prediction accuracy on the test set. The proportion of variables that can be predicted with accuracy above .85, .90, .95 are computed to evaluate the overall reconstruction accuracy for each choice of subsampling proportion. The results are presented in the left-hand panel of Figure A4. In general, the derived variables can be reconstructed better when the sample size gets larger. When the subsampling proportion is higher than .5, that is, when more than 1,500 action sequences are used for training the action sequence autoencoder, the proportion of variables with accuracy above .85 and .90 becomes stabilized while the proportion for accuracy above .95 continues increasing.

To study the score prediction performance of the extracted features, we fit a linear model with the extracted features and binary responses from 14 items as covariates to predict respondents' numeracy and literacy score for each choice of subsampling proportion. The models are fitted on the training set and their prediction performance is evaluated on the common test set. The recorded OSR^2 , is summarized in the middle and right-hand panels of Figure A4. Similarly to the reconstruction performance, the score prediction performance improves as the sample size gets larger. The performance becomes stabilized when the subsampling proportion is higher than .4. The dashed lines in the middle and right-hand panels represent the prediction based solely on the 14 binary responses. The linear model that the prediction comes from is fitted on the training set with 3,145 respondents. As the two plots suggest, the feature-based models fitted on a few hundred observations can outperform response-based models fitted on a few thousand observations.

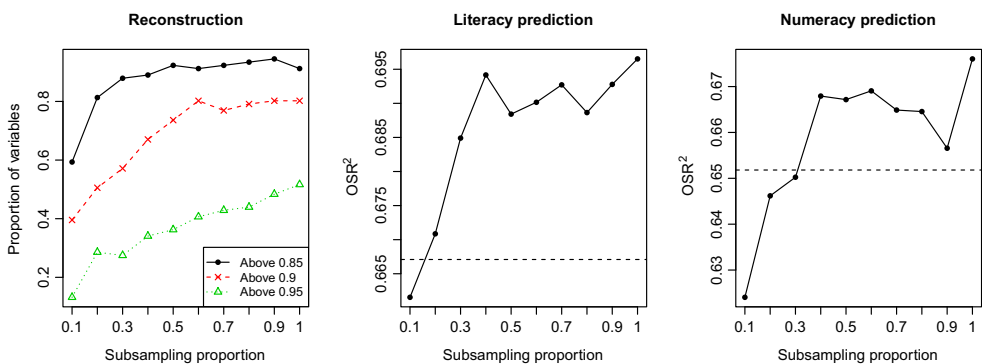


Figure A4. Reconstruction (left) and score prediction (middle and right) performance for various choices of subsampling proportion. The dashed lines in the middle and right panels represent the prediction results from the response-based model fitted on 3,145 observations. [Colour figure can be viewed at wileyonlinelibrary.com]

B.3. Prediction results from one-step approach

In Sections 4.4 and 4.5 the variables of interest are essentially predicted from action sequences via a two-step approach. In the first step, features are extracted from action sequences via fitting an action sequence autoencoder. In the second step, linear models or generalized linear models are used to predict variables of interest from the extracted features. In principle, one can also make predictions directly from action sequences by building a neural network for each variable to be predicted. The neural network takes an action sequence as input and outputs the response variable. In this section we explore the prediction performance of the one-step supervised learning approach and compare it with the results obtained from our two-step approach.

The neural network we use in the one-step approach consists of an embedding layer, an RNN, and an FFN. Its architecture is depicted in Figure A5. The embedding and RNN layers of this neural network resemble the encoder in the action sequence autoencoder. The output vector of the RNN at the last time-step is then fed into the FFN instead of a decoder.

We use both the one-step and two-step approach to predict respondents' gender, age, literacy score and numeracy score from their action sequences in the PIAAC data set. In the two-step approach, 100 features are extracted and used for fitting (generalized) linear models. In the one-step approach, we set $K = 100$ and use no FFN hidden layer to mimic the settings in the two-step approach. The 3645 respondents who responded to all 14 items are randomly split into training (2,645), validation (500), and test sets (500). For each combination of item and response variable, we fit the one-step neural network described above and the generalized linear model in the two-step approach using the training and validation set and evaluate the prediction performance of the two approaches on the test set. The evaluation criteria are the same as described in the main text. The results are presented in Figure A6. For gender, literacy score and numeracy score, the two approaches are comparable in terms of prediction performance. However, for age, the performance of the two-step approach is significantly better than that of the one-step approach.

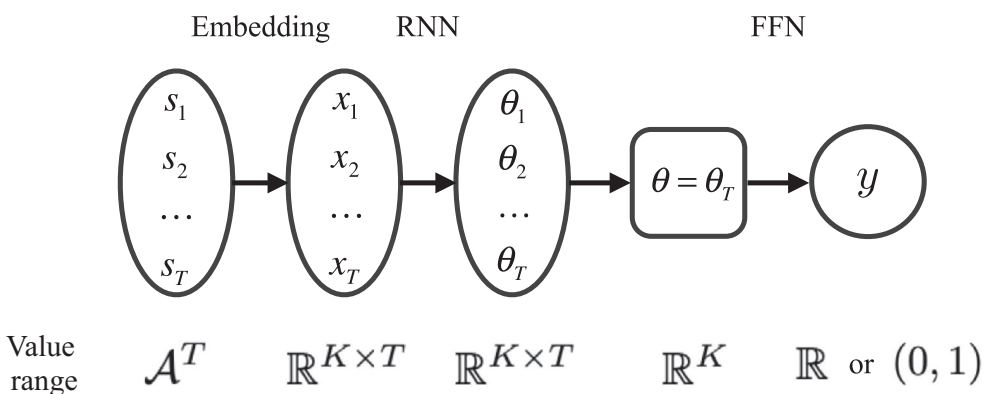


Figure A5. The structure of the neural network used in the one-step prediction approach.

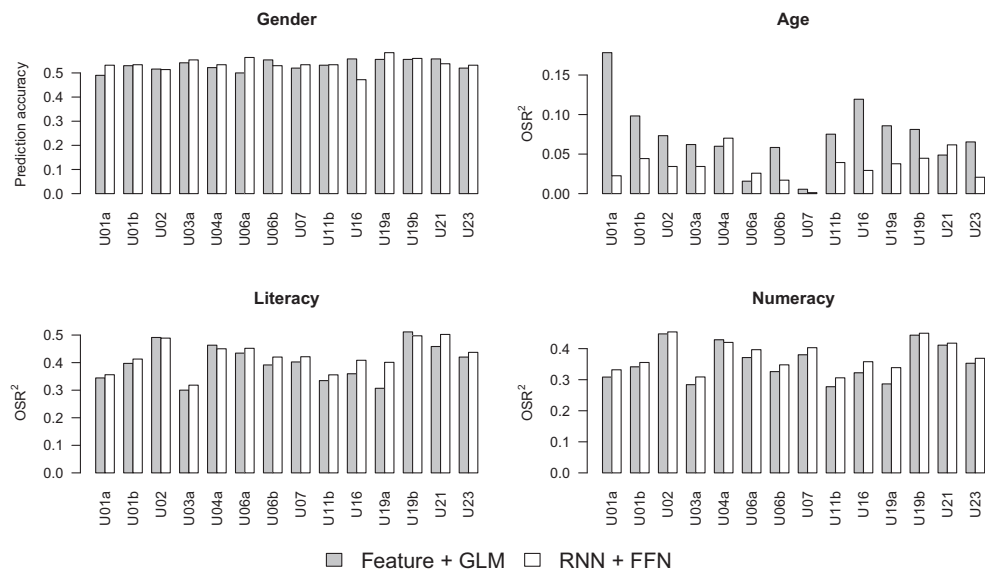


Figure A6. Comparison of the prediction performance of the one-step and two-step approaches.