TOPICS
TOPICS IN COGNITIVE SCIENCE

Topics in Cognitive Science 14 (2022) 873–888
© 2022 Cognitive Science Society LLC.
ISSN: 1756-8765 online
DOI: 10.1111/tops.12616

This article is part of the topic "Best of Papers from the 19th International Conference on Cognitive Modeling," Terrance C. Stewart and Joost de Jong (Topic Editors).

Validating and Refining Cognitive Process Models Using Probabilistic Graphical Models

Laura M. Hiatt,[a] Connor Brooks,[b] J. Gregory Trafton[a]

[a]*Navy Center for Applied Research in Artificial Intelligence, US Naval Research Laboratory*
[b]*Department of Computer Science, University of Colorado Boulder*

Received 1 November 2021; received in revised form 25 April 2022; accepted 3 May 2022

Abstract

We describe a new approach for developing and validating cognitive process models. In our methodology, graphical models (specifically, hidden Markov models) are developed both from human empirical data on a task and synthetic data traces generated by a cognitive process model of human behavior on the task. Differences between the two graphical models can then be used to drive model refinement. We show that iteratively using this methodology can unveil substantive and nuanced imperfections of cognitive process models that can then be addressed to increase their fidelity to empirical data.

Keywords: ACT-R; Cognitive models; Graphical models

1. Introduction

Building cognitive process models of human behavior is a challenging task that has rich rewards. Such models can be used for many purposes, including to better understand where people make errors (Hiatt & Trafton, 2015), to better understand how to effectively teach people new skills (Lee, Anderson, Betts, & Anderson, 2011), and to design effective computer interfaces (Cao, Ho, & He, 2018).

Correspondence should be sent to Laura M. Hiatt, Navy Center for Applied Research in Artificial Intelligence, US Naval Research Laboratory, 4555 Overlook Ave, SW, Washington, DC 20375, USA. E-mail: laura.hiatt@nrl.navy.mil

Because human thought processes can be only observed indirectly, such as by recording external behavior (e.g., reaction times, responses to questions, etc.), each cognitive process model serves as a de facto theory of human behavior on the task being modeled. To validate those theories, the simulated behavior of the models on the task can be compared to the behavior of human participants on the task using measures, such as statistical goodness-of-fit measures (e.g., R^2, RMSE, etc.). A strong fit indicates that the model is a good theoretical candidate for how humans complete the task.

However, there are two related issues with this development methodology that we address in this paper, both of which stem from the fact that a strong or weak statistical fit does not necessarily indicate which parts of the model may fit human behavior better than others. First, a weak statistical fit does not necessarily give any actionable information for which part of the model to modify to improve it. Second, a strong statistical fit for part of the model's task may mask issues with another area.

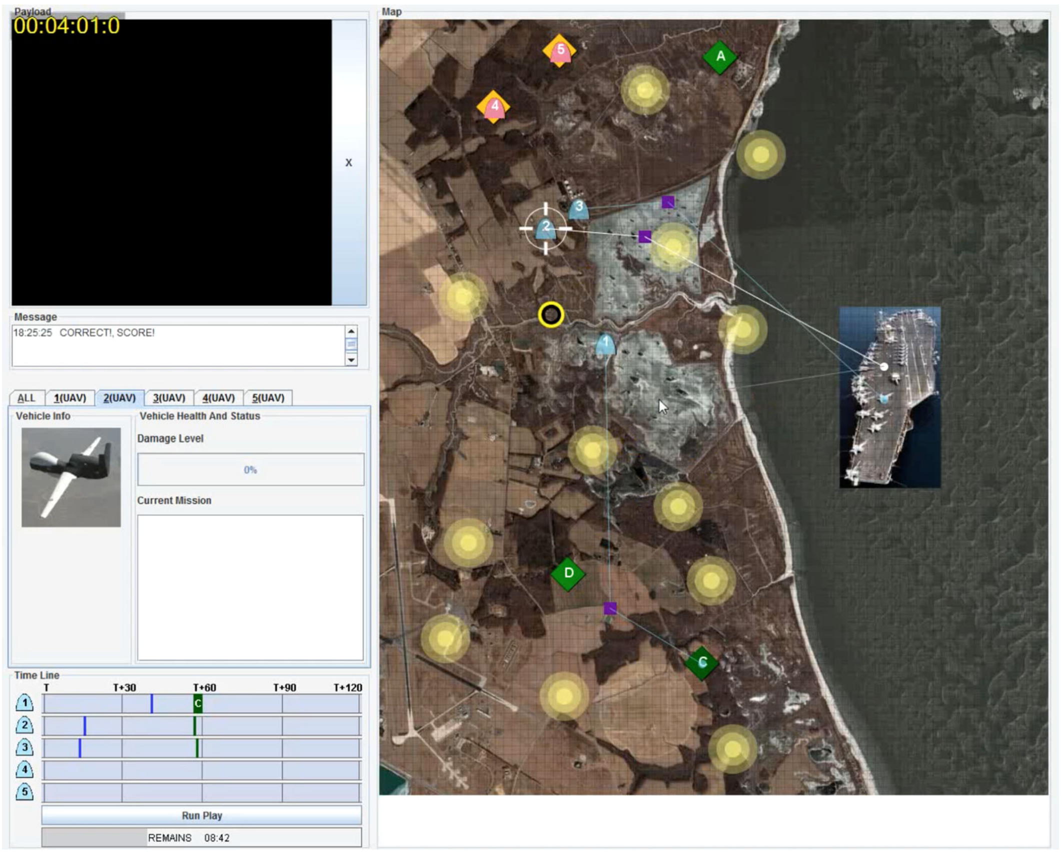
Here, we use hidden Markov models (HMMs), a type of probabilistic graphical model, as a new analysis tool for validating the efficacy of a cognitive model of a human behavioral task. First, we use the cognitive model to generate synthetic data of human behavior on a task. Then, we train two HMMs using two datasets: (1) human empirical behavior on the task; and (2) the generated synthetic data. By comparing these HMMs both qualitatively and quantitatively, we can not only measure how similar the datasets (and thus the underlying behaviors, or models of behavior) are, but also, because HMMs are a form of graphical model, visually see ways in which the datasets differ. We show that using this process can improve cognitive model fidelity by both of these qualitative and quantitative metrics, as well as improve the predictive accuracy of the cognitive model when predicting a human's next action on a task.

In the following sections, we describe: the task; the empirical data that were collected; the graphical model approach; the initial cognitive model of the task; and the revision of the cognitive model. Finally, we discuss the resulting improvement in predictive performance.

2. Task description

In order to study how cognitive models of complex behavior can leverage machine learning as a tool for improvement, we turned to a supervisory control task. Specifically, we considered how people performed while interacting with the research environment for supervisory control of heterogeneous unmanned vehicles (RESCHU) (Boussemart & Cummings, 2008) simulator. RESCHU is an interactive supervisory control task that requires complex decision making, problem solving, and reasoning.

Fig. 1 shows the simulation, which has three panels: a map panel, a status panel, and a payload panel. The map panel (Fig. 1, right) displays unmanned aerial vehicles (UAVs) (blue/red half ovals), targets (orange/green diamonds) toward which UAVs are moving, and hazard areas (yellow circles) which should be avoided by UAVs and can change location over time. The status panel (Fig. 1, bottom left) shows the status of the UAVs and includes information on vehicle damage, time until the vehicle reaches a waypoint or target, and time remaining in the simulation. The payload panel (Fig. 1, top left) is used by the operator to perform a



18:25:25 CORRECT!, SCORE!

Fig. 1. A screenshot of the RESCHU environment simulator used in this experiment.

manual visual acquisition task once the UAV has reached the target. It is not critical to this work so we largely omit its consideration; it is more fully described in Breslow, Gartenberg, McCurry, and Trafton (2014).

The goal for an operator's session in RESCHU was to monitor and guide the five UAVs to reach as many targets as possible, and complete the corresponding payload tasks, while avoiding damaging the UAVs in the hazard areas.

At the start of the simulation, the UAVs were randomly assigned to targets; thus, the UAVs might not be directed toward the optimal target. After a target was reached and the visual acquisition payload task was complete, the target "reset," and the UAV was randomly assigned to a new currently unassigned target, which again might not be optimal.

Because of this suboptimal automation, as well as the changing location of the hazard areas, a critical subtask of RESCHU was changing the target of a UAV. Operators could do this at any time by using the mouse to click on the UAV and then clicking on the UAV's new target. We focus on this subtask here because of its import as well as because, as we will see, even in this "simple" subtask, there is variation in how operators perform it, making it an ideal subtask to test our proposed methodology. The end goal of the modeling effort is to be able to predict an operator's final selected target, so the interface can better assist with the selection process (e.g., preselecting the target, highlighting the target, etc.).

2.1. Empirical data

The empirical data we consider were based on 10 participants using RESCHU. Participants were provided extensive training on the RESCHU system through an online tutorial, in-person instruction, and walk-throughs. Participants also had as much time as they wanted to use the entire system until they were well-versed in the intricacies of RESCHU. Participants were all volunteers, healthy, with age less than 30 years. Details on the methodology of the study are available in Breslow et al. (2014).

After a participant was fully trained on RESCHU, they were seated approximately 66 cm from the computer monitor and were calibrated on an SMI RED eye tracker operating at 250 Hz, which collected eye tracking data once the experiment began. A fixation was defined using the dispersion method based on a minimum of 15 eye samples within 60 ms and within 50 pixels (approximately3° of visual angle) of each other, calculated in Euclidian distance. Fixations on specific objects were automatically identified after all data collection was completed. The simulation also logged all operator *actions*, i.e., mouse clicks, indicating what object was clicked on (i.e., selected) at different times.

All instances of changing a UAV's target were manually extracted from the simulation. A total of 200 sequential process traces, with eye fixations and mouse clicks listed in chronological sequence, were created by these participants. For this subtask, a process trace contained sequences of the set of possible observations of the operator's behavior, *{uav1, uav2, …, haz1, haz2, … tar1, tar2, …, Action-SelectUAV1, Action-SelectUAV2, …, Action-SelectTar1, Action-SelectTar2, …}*. Note that if the observation corresponds to a mouse click, it is designated specifically as an "Action-Select"; otherwise, it is an eye gaze fixation.

To provide preliminary evidence of different strategies people may use on this task, we were able to create a simple coding scheme for the empirical data to categorize the different strategies people used (e.g., a planning strategy needed to fixate on both the UAV and the target before selecting the UAV; an opportunistic strategy picked and selected a UAV before looking for a target). The coding scheme was implemented computationally and run on all the empirical data to separate the data by strategy.

3. Hidden Markov models for comparing datasets

A hidden Markov model (HMM) is a graphical model that stochastically transitions between states using the Markov assumption (i.e., transitions depend on only the current state). The *hidden* term refers to how states are not directly observable; instead, states stochastically emit observations that give clues to what state the model is currently in. Fig. 3 shows the example of HMMs that we discuss later in our analysis.

A typical HMM is formally defined by the tuple $\langle S, Z, A, B, \pi \rangle$:

- S is the set of states.
- Z is the set of observations.
- A is an $|S| \times |S|$ matrix defining transition probabilities between states $a_{i,j} = p(x^t = s_j | x^{t-1} = s_i)$.
- B is an $|S| \times |Z|$ matrix defining observation probabilities of the states $b_{i,k} = p(o^t = z_k | x^t = s_i)$.
- π is a vector with initial state probabilities $\pi_i = p(x^0 = s_i)$.

In this definition, x_t and o_t represent the true state and emitted observation at time t, respectively. Thus, HMMs probabilistically move between states and, when in a state, probabilistically emit observations corresponding to those states. This makes them ideal to use for modeling tasks on human behavior, such as the RESCHU supervisory control task, where people move between hidden states and probabilistically take actions based on the state they are in.

3.1. Learning HMMs

In order to create an HMM that models a dataset of human behavior, existing techniques for learning HMMs can be used. Learning for HMMs can refer to both learning the topology of the HMM (i.e., learning what states connect to others; Singer & Ostendorf, 1996; Siddiqi, Gordon, & Moore, 2007), as well as learning the parameters of it (i.e., learning the values of A and B; Rabiner, 1989). Here, we adapt the basic strategy from Singer and Ostendorf (1996) for learning the HMM topology using repeated "state splitting," followed by standard Baum–Welch parameter learning (Rabiner, 1989). The Baum–Welch algorithm uses successive iterations of forward and backward passes through all available data traces to iterate toward the transition and observation probabilities that maximize the probability of the observed data.

State splitting strategies start with a given HMM of a single state (or a few states), then iteratively split states to expand it until a desired number of states is reached. To find the best topology, at each expansion step, each state is split, the resulting model is trained using standard Baum–Welch learning, and then the split that provides the best increase in likelihood is chosen. Repeating these two steps of state splitting and Baum–Welch parameter learning results in joint learning of both the topology and parameters of the HMM.

The specific training procedure we utilize begins by reading in a set of observation sequences. We prepend each sequence with a *START* observation and append each with an *END* observation to denote the fixed beginning and end of the sequence. Then, an HMM is initialized with a start state that emits only *START*, a middle state that emits all observations other than *START* and *END*, and an end state that emits only *END*. The start state transitions only to the mid state, the middle state transitions to itself and the end state, and the end state terminates the sequence. The middle state's initial observation and transition probabilities are trained using Baum–Welch to be set to the expected observation and transition probabilities from the dataset of observation sequences. Thus, the middle state is initialized to be a single-state HMM topology that maximizes the likelihood of the dataset's observation sequences.

From this point, state-splitting is used to expand the HMM topology, one new state at a time. At every expansion step, a new possible HMM is created and trained for each candidate split, with every state (other than the specialized start and end states) considered a candidate for splitting.

For each new candidate HMM, both of the newly split states are initialized with the same transitions as the original state. The new HMM's parameters are then trained using Baum–Welch learning. After all possible HMMs for the current expansion step have been created and trained, the candidate HMM that maximizes the likelihood of the dataset is selected for use or as the basis of the next expansion step; the rest are discarded. This procedure can end once either a fixed number of states is reached, or some measure of fit (such as the likelihood of a validation set) stops improving.

3.2. HMMs as dataset representations

Learning an HMM directly from a dataset produces a graphical model that represents some underlying structure in the data. In particular, if two HMMs are created from datasets generated by two different sources, the learned HMMs can provide a graphical representation of the differences in the sources underlying the two datasets. This provides a novel way to analyze and validate cognitive process models: comparing HMMs learned from synthetic data from a cognitive model with those learned from human empirical data.

Intuitively, we should be able to learn HMMs that do not depend on the specific label of current action (i.e., looking at *tar1* vs. *tar2*), but instead consider its meaning (i.e., looking at a target of interest vs. looking at a target not of interest). There is an additional consideration, therefore, to using HMMs as an analysis tool for cognitive process models on tasks where a "template"-style process is applied to various versions of a task. As an example, in the UAV rerouting subtask, the same processes are followed no matter which UAV is being rerouted to which target. Ideally, in order to best illustrate this process, the learned HMM would be agnostic to the specific UAV and target in a particular task. But because the specific targets of interest differ across observation sequences, learning a single-standard HMM on a dataset for this subtask would essentially be meaningless for understanding the process generating the data: the HMM will not be able to meaningfully differentiate between targets of and not of interest.

To address this need, we next describe how we collapse observations into composite observation sets in order to train effective HMMs on these types of "template" tasks.

3.3. Composite observations for HMMs

The first step in collapsing observations is to divide all possible raw observations into classes (e.g., group together all hazard observations into a *haz* class, etc.). For each class, we represent it using either one or two composite observations. For classes like *hazard*, all observations can generally be collapsed into one composite observation *haz*. For classes like *target*, however, they cannot: that would meaningfully impact an HMM's ability to understand the operator's observations and actions. Such classes have two composite observations: *tar+*

which indicates the target of interest for a given sequence, and *tar–* which indicates targets not of interest.

It is straightforward to convert raw observation traces into composite observation traces. Each raw observation is replaced with either its single composite symbol, or with the appropriate dual composite (i.e., if the raw observation is *tar2* and *tar2* is the selected target, it is replaced with *tar+*; otherwise, it is replaced with *tar–*). The resulting composite dataset can then be used to train an HMM as described above.[1]

3.4. HMM comparison measures

HMMs can readily be compared to one another to find structural or other similarities or differences in the underlying data used to train them. We consider two ways to compare HMMs: one quantitative and one qualitative. Quantitatively, comparing HMM topology and parameters involves calculating the similarity of the expected outputs produced by two HMMs. A classic approach to do so, described by Juang and Rabiner (1985), is to estimate the Kullback–Leibler divergence between the probability distributions of observation sequences generated by the two HMMs. This estimation is done through a Monte Carlo approach by repeatedly generating observation sequences from one HMM, calculating the probability of each of these sequences being emitted from *both* HMMs, and comparing the two probability values. Although this measure is asymmetric, we transform it to a symmetric measure by calculating it in both directions and averaging them. As the number of observations goes to infinity, the estimate approaches the true Kullback–Leibler divergence of the two HMMs.

Qualitatively, a comparison of HMMs can be done by visually viewing them, and comparing transitions between states as well as observation probabilities. For example, if one HMM always begins by entering a state where it looks at a UAV, while another always begins by entering the state where it looks at a hazard, that can be viewed as a meaningful qualitative difference between the models.

4. Cognitive model development cycle

Our cognitive model development cycle began by one of the authors of this paper handwriting a cognitive model in adaptive character of thought-rational/embodied (ACT-R/E) (Trafton et al., 2013) to capture human performance on the UAV rerouting subtask. The original models are identical to that described in Trafton, Hiatt, Brumback, and McCurry (2020); we describe them generally here, but interested readers can refer to that work for their specifics.

4.1. ACT-R/E

We chose the cognitive architecture ACT-R/E to model human performance on the RESCHU task because we needed a cognitive architecture that captures human behavior at a fine-grained level of analysis (goals, eye fixations, physical actions, etc.), and because it has a long history of providing detailed process descriptions of how people perform tasks (Anderson, 2009). Additionally, it can easily model different strategies for completing such tasks, and can generate large amounts of data representing the underlying model that can be used for comparison purposes (Hiatt, Harrison, & Trafton, 2011). Note, though, that our methodology should work with any cognitive architecture.

ACT-R/E is a hybrid symbolic/subsymbolic production-based system based on ACT-R (Trafton et al., 2013). For the purposes of this manuscript, there are no critical differences between ACT-R and ACT-R/E. An ACT-R/E model is, essentially, a set of if-then rules that make requests and access information in the model's working memory (which is designed to reflect people's working memory). Depending on the current contents of working memory, different if-then rules can fire, producing behavior and potentially changing the contents of working memory to encourage other if-then rules to fire. These sequences of firing if-then rules capture the process of human cognition and, via the contents of working memory, are influenced by what the model sees, knows, and does (just like people are).

More technically, ACT-R/E consists of a number of modules, buffers (that collectively represent working memory), and a central pattern matcher. Modules contain a relatively specific cognitive faculty associated with a specific region of the brain. For each module, there are one or more buffers that communicate directly with that module as an interface to the rest of ACT- R/E. At any point in time, there may be at most one item in any individual buffer; thus, the module's job is to decide what and when to put an object into a buffer. The pattern matcher uses the contents of the buffer to match specific productions (if-then rules). ACT-R/E interfaces with the outside world through the visual module, aural module, motor module, and vocal module. Other current modules include the intentional, imaginal, temporal, and declarative modules. ACT-R/E perceives the physical world by either robotic or virtual sensors (Trafton et al., 2013). ACT-R/E's goals are to maintain cognitive plausibility as much as possible while providing a functional architecture to create models of human-level intelligence.

We briefly discuss the modules that are especially relevant to this project below. Fig. 2 shows a schematic of ACT-R/E, which is discussed more fully in (Trafton et al., 2013).

4.1.1. The intentional module

The intentional module is used to set, change, track, and remove goals. Like people, ACT-R/E does not enforce a strong goal order or goal-stack (Altmann & Trafton, 2002) and enables both top-down and bottom-up goal execution.

4.1.2. The declarative module

The declarative module provides a method to encode, store, and retrieve items from long-term memory into working memory. This helps support, for example, step (4) above of visually acquiring an object, below.

4.1.3. The visual module

The visual module is used to provide a model with information about what can be seen in the current environment. This module is what finds objects in the world that, for example, are needed to perform an action. Critically, numerous empirical studies have shown that there is

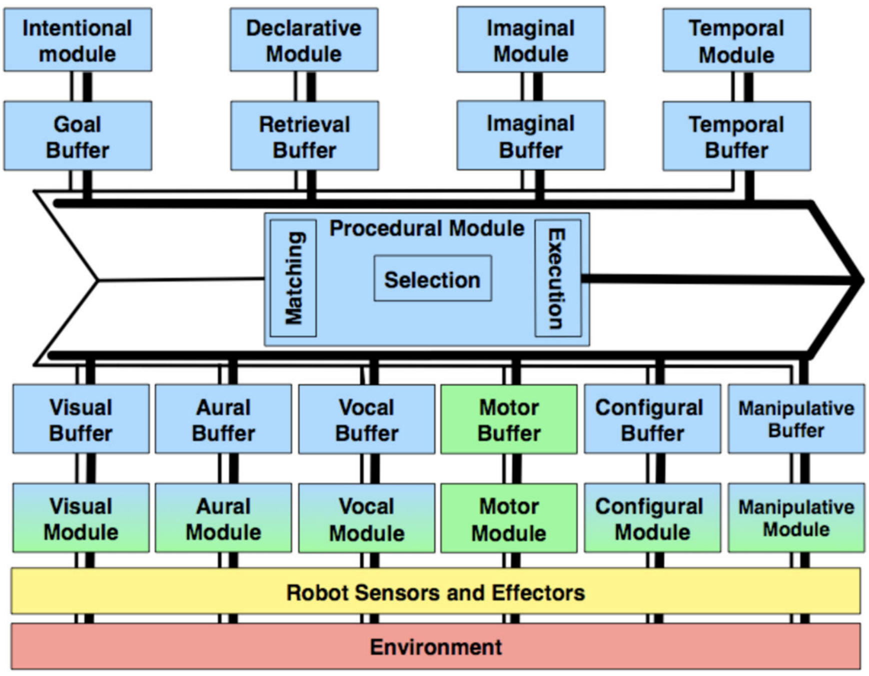


Fig. 2. An architecture diagram of ACT-R/E.

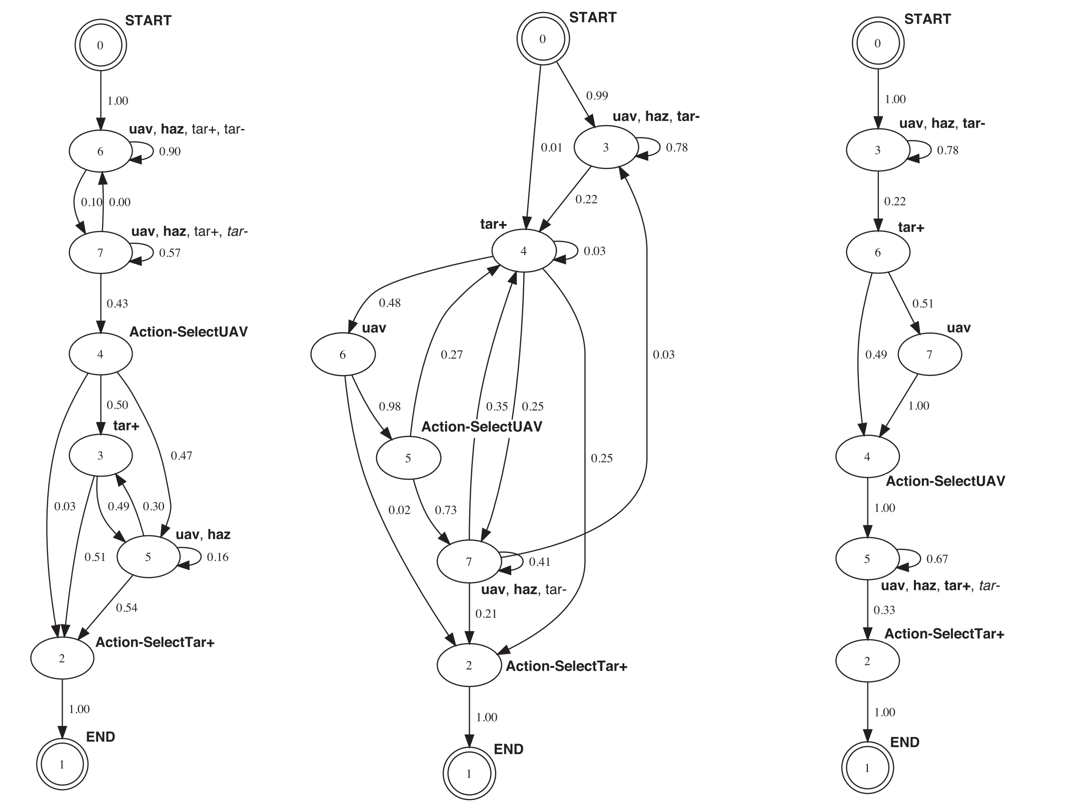
a specific set of perceptual and declarative steps that need to occur in order for a person to recognize, understand, and identify an object in the world in order to take action on it (Byrne & Anderson, 1998):

1. People set a goal to search for specific features (e.g., color or shape).
2. People move their attention to objects that have those features.
3. People make a saccade to that object and identify it visually (e.g., provide an internal label).
4. People make memory retrievals in order to interpret what that object is in the current context.

Additionally, because perceptual attention is rarely on the same thing for an extended period of time, there is a great deal of variability where people "look around" in environments, especially dynamic ones. ACT-R/E is also able to capture this variability in a cognitively plausible way.

4.1.4. The motor module

ACT-R/E's motion is controlled by the motor module. In this work, the motor is used to control physical actions (e.g., mouse movement and clicks). It can be used to provide cognitively plausible mouse movements, including timing and variability.



(a) Planning Empirical (b) Planning Synthetic (Original) (c) Planning Synthetic (Revised)

Fig. 3. HMMs of the planning strategy built from different datasets. The numbers of the states indicate the order in which they were created. The meaning of the states can be derived from the observations they emit, which are shown in the figures. Bolded observations occur with greater than 20% likelihood; italicized observations occur with less than 10% likelihood.

4.1.5. Model trace logging

Given an ACT-R/E model of a task, we can easily and readily execute that model to generate synthetic observational data labeled with observations, actions, and the goals behind those actions. Recall that when a model is executed in the surrounding architecture, behavior is produced by firing if-then rules matching against the fluctuating contents of working memory. At each step of the way, the architecture can log what is occurring. It can log, for example, what the model saw as it executed (such as by looking at the visual module's fixations), and what it did (such as by recording actions taken by the motor module). Each time it logs these events, it can also record the goal driving the events by inspecting the current state of the intentional module. This execution and logging can happen very rapidly, allowing us to quickly generate multiple logged sequential process traces of simulated human performance a model's task. Because the environment is dynamic and changes, and because of the aforementioned variability in perceptual processing, each model run may be slightly different, just as empirical traces of human behavior different from one another.

4.1.6. Model development

Normal model development in ACT-R/E is a cyclic process where a model developer writes a model, has it perform a task several times, and collects summary statistics of that model's performance (e.g., mean response time, task completion, etc.). Based on those summary statistics, the model developer can refine the model to improve the fit of those statistics to statistics of human performance on the task. Importantly, however, such summary statistics do not strongly indicate which aspects of the model should be changed to improve the fit; nor do they provide strong reasoning for why a model's fit is strong or weak to begin with. Our work addresses this gap.

4.2. Original model description

While performing the rerouting subtask, people could use a variety of cognitive strategies; here, we focused on modeling two strategies introduced when discussing the empirical data: a *planning* strategy and an *opportunistic* strategy.

The planning strategy captures the insight that people sometimes plan a few actions ahead, or search for the best action to do, when performing a task. Here, the strategy first searches for a UAV whose target needs to be changed (because it is on a collision course with a threat or is far away from its target, etc.), by using its perception to see the interface, and its memory to interpret what it sees. It then holds the UAV in working memory while searching for a better target (e.g., one that does not intersect a hazard or that is closer). After identifying the UAV and the new target of interest, the model executes the actions to change the UAV's target (i.e., clicking on the UAV, and then clicking on the new target).

The opportunistic strategy occurs when people may not have time, resources, or inclination to plan ahead. The model of this strategy sequences its actions differently. It first searches for a UAV whose target needs to be changed using its perception, and then immediately clicks to select the UAV without a specific target yet in mind. Next, the model searches for a target where the UAV could be sent using its perception. After an appropriate target is found, it clicks on it to change the UAV to go to that target.

Note that while the differences in strategies are subtle, they are different in their actions (i.e., mouse clicks), as well as patterns of eye movements. Also, the planning strategy clearly requires greater utilization of working memory (e.g., needing to hold the UAV in mind while searching for an appropriate target); however, we note that even though the RESCHU task is dynamic, there seems to be enough time to execute both strategies within the constraints of the subtask.

These models were not meant to be comprehensive: they were meant to capture the strategies that operators were most likely to use. Notably, the models did not capture simultaneous strategies, switching strategies, rare strategies, or learning over time; however, as we will see, we can still use them to accurately predict operator performance.

4.3. Analysis setup

There are two steps to setting up the analysis of the cognitive models: acquiring the right data and training the HMMs.

4.3.1. Data

Evaluating the cognitive process models using machine learning techniques requires both empirical and synthetic data stemming from the cognitive models. The empirical data were described when introducing the RESCHU task. For the synthetic data, we used the developed cognitive models to generate observation traces indicating the model's theory of how people perform the task. Critically, these models generate traces of observational data that were identical in form to the traces that were generated from the human participants, including eye fixations and mouse clicks listed in chronological order. All together, both the planning model and the opportunistic model were each run 20,000 times to generate 20,000 individual, distinct traces of synthetic human performance for each strategy.

4.3.2. HMM training

HMMs can then be trained using each data source: empirical planning, empirical opportunistic, synthetic planning, and synthetic opportunistic. Before training, each dataset was converted into its composite dataset as described above. Each HMM was limited to five splits while learning the topography, resulting in a total of eight states per HMM (six "split" states plus the start and end states). For each candidate split, parameters were learned using 500 Baum–Welch iterations. Fig. 3a shows one such trained HMM.

4.4. Analysis

The trained HMMs can next be compared both quantitatively and qualitatively. Quantitatively, the HMM distance measure, described above, calculates how related the different HMMs are. While there is not an exact "ideal" target for these values, lower values indicate that the data generated by the two models overlap more and, as such, lower values are better. These values show that the cognitive model of the planning strategy is closer to that of the empirical data than the model for the opportunistic strategy. These values do not, however, offer any insight into why this is the case.

In contrast, Figs. 3a and b show a qualitative comparison of the empirical planning HMM and the synthetic planning HMM. As they show, there are several qualitative ways in which the HMMs differently characterize the behavior of their respective data. Notably, as we will discuss in the following section, there are several differences in what observations occur directly before and after actions (mouse clicks).

4.5. Revision

The HMM analysis highlighted several differences between the cognitive models and the empirical data. These differences suggested changes in the models that potentially could lead

Table 1
Distances between the empirical-based HMMs and original and revised process model-based HMMs for the planning and opportunistic strategies. Lower scores indicate greater similarity between the HMMs.

	PlanSynOrig	PlanSynRev		OppSynOrig	OppSynRev
PlanEmp	0.209	0.181	OppEmp	0.312	0.181

to not only a greater theoretical understanding of dynamic tasks but also improved fits and better prediction. Two changes were made to both strategy models based on the revealed differences between them and the empirical data.

4.5.1. Looking at the target of interest before selecting it

Before clicking the target of interest, the original cognitive models looked directly at it and then selected it very consistently. However, the empirical data did not show such a strong relationship. One possible explanation for this is that people used a sort of embodied cognition by focusing on a target, moving their mouse to that target, looking around more, and then simply clicking the mouse to finalize the selection. We implemented this aspect in the planning and opportunistic cognitive models by providing a 50% chance that people would use a form of embodied cognition.

4.5.2. Looking at the UAV immediately after selecting it

The empirical data suggested that people frequently (50%) looked at the target of interest immediately after selecting the UAV. In contrast, the original cognitive model used a GOMS-style approach, generally looking at the UAV immediately after selecting it to confirm that it was selected. To address this, instead of increasing the probability that the target was focused on immediately after selection, we instead decreased the probability that the model confirmed the UAV was selected. By verifying that the UAV had been selected less often, we expected that the target would be examined sooner.

Note that for both of these changes, we did not perform any sort of parameter-space search; we simply changed the aforementioned probabilities to 50%. We assume that if we had performed additional parameter-space search, the model fit would be better, perhaps at the cost of over-generalization.

5. Results

We first look at the HMM measures to evaluate whether the model fit to the empirical data has improved after revision. Table 1 shows the quantitative values for original cognitive process models (PlanSynOrig, OppSynOrig) and revised cognitive process models (PlanSynRev, OppSynRev) compared to the empirical data (PlanEmp, OppEmp). As the table shows, after the revision, the quantitative fit did, in fact, improve. The planning strategy showed a moderate improvement, while the opportunistic strategy showed a large improvement. Qualitatively, as Fig. 3 shows, the new planning cognitive model shows a stronger

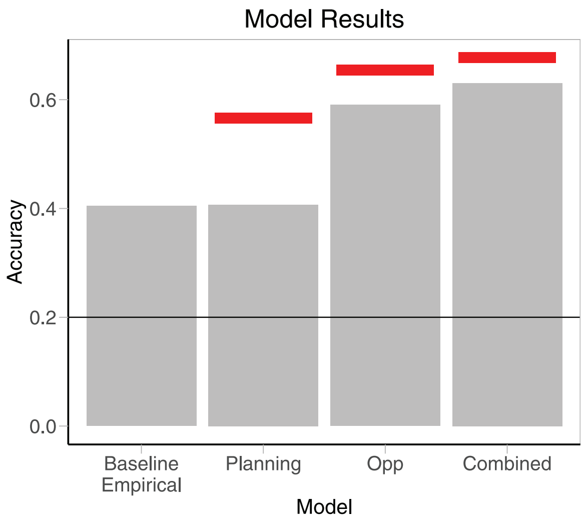


Fig. 4. Original model predictive accuracy (bar) and updated model predictive accuracy (red rectangle). The black horizontal line is chance.

topological and parameter similarity to the empirical HMM; the opportunistic shows a similar improvement.

As an additional measure, we have suggested that an effective way to evaluate a cognitive model is by predicting specific steps during task execution (Breslow et al., 2014; Ratwani & Trafton, 2011; Trafton et al., 2020). In Trafton et al. (2020), we showed that we could predict what target a UAV would be directed to by using synthetic data generated by a cognitive model to train a convolutional neural network (CNN). If the HMM analysis added value by improving the cognitive model, we should see an improvement to the predictive capabilities of the CNN: the cognitive model should capture the patterns in the data better, allowing better prediction of the specific target selection.

Thus, we trained CNNs using synthetic data from the original and revised cognitive models to see whether the revision resulted in better predictive performance. For each CNN, 10-fold cross validation was used to divide the empirical data into training and testing data. All conditions used the same folds for training and testing, and all models were evaluated on the empirical data. In addition to training CNNs for the planning and opportunistic strategies, we also trained a combined strategy CNN that used half of each. As Fig. 4 shows, in all cases, the revision improved the predictive performance, showing that not only did our proposed process improve HMM-based metrics, but it also improved the models' eventual predictive power as well.

6. Discussion

In this paper, we have described a new methodology that cognitive modelers can use to develop, analyze, and verify cognitive process models. In it, we learn HMMs from data from empirical experimentation, as well as from synthetic data generated by candidate cognitive models. By comparing those HMMs both qualitatively and quantitatively, we can see the cognitive models' goodness-of-fit, as well as determine concrete ways in which the cognitive model can be improved. This is different than validating models with statistical goodness-of-fit measures, which do not offer concrete pointers for improving cognitive models. We also show that this process can lead to increased predictive performance by the cognitive model. In future work, we plan to expand this methodology to demonstrate it for models in additional cognitive architectures and of additional tasks.

Interestingly, this methodology revealed an issue with how visual-based actions are typically done in cognitive models; namely, look at, encode, prep to act, and then act upon it. The HMM built for the empirical data suggests a different paradigm for dynamic tasks: look at, encode, prep to act, and then *possibly look elsewhere* before acting. This type of insight is not possible to glean from summary statistics or other typical measures of cognitive process models; a graphical model allowed us to find this approach.

Another key insight is that, perhaps surprisingly, the improvements in HMM similarity and predictive performance did not mirror one another. For the HMM measure, the opportunistic model saw the greater improvement after its revision. Considering predictive performance, the planning model saw the greater improvement. This is because the different representations of the HMM and the CNN lead them to capture different aspects of the data: the HMM focuses on underlying structure; and the CNN focuses on nonlinear patterns. This subtlety only highlights the need for more diverse tools for cognitive model analysis like the one that is proposed here.

Acknowledgments

We thank the Office of Naval Research for funding this work (LH and GT). We also thank Tony Harrison for useful discussions. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the US Navy, Department of Defense, or the US Government.

Note

1 An additional benefit of an HMM that reasons over composite UAVs, hazards, targets, and so on is that it allows for adding or removing additional items (such as more targets) at run time without going through training again.

References

Altmann, E. M., & Trafton, J. G. (2002). An activation-based model of memory for goals. *Cognitive Science*, *26* (1):39–83.

Anderson, J. R. (2009). *How can the human mind occur in the physical universe?* (volume 3). Oxford University Press.

Boussemart, Y., & Cummings, M. (2008). Behavioral recognition and prediction of an operator supervising multiple heterogeneous unmanned vehicles. *Humans Operating Unmanned Systems*.

Breslow, L. A., Gartenberg, D., McCurry, J. M., & Trafton, J. G. (2014). Dynamic operator overload: A model for predicting workload during supervisory control. *IEEE Transactions on Human–Machine Systems*, *44*(1).

Byrne, M. D., & Anderson, J. R. (1998). Perception and action. In J. R. Anderson & C. Lebiere (Eds.), *Atomic components of thought* (pp. 167–200). Mahwah, NJ: Lawrence Erlbaum.

Cao, S., Ho, A., & He, J. (2018). Modeling and predicting mobile phone touchscreen transcription typing using an integrated cognitive architecture. *International Journal of Human–Computer Interaction*, *34*(6), 544–556.

Hiatt, L. M., Harrison, A. M., & Trafton, J. G. (2011). Accommodating human variability in human–robot teams through theory of mind. In *Proceedings of IJCAI*.

Hiatt, L. M., & Trafton, J. G. (2015). An activation-based model of routine sequence errors. In *Proceedings of the International Conference on Cognitive Modeling*.

Juang, B.-H., & Rabiner, L. R. (1985). A probabilistic distance measure for hidden Markov models. *AT&T Technical Journal*, *64*(2), 391–408.

Lee, H. S., Anderson, A., Betts, S., & Anderson, J. R. (2011). When does provision of instruction promote learning? In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Ratwani, R. M., & Trafton, J. G. (2011). A real-time eye tracking system for predicting and preventing postcompletion errors. *Human–Computer Interaction*, *26*(3).

Siddiqi, S. M., Gordon, G. J., & Moore, A. W. (2007). Fast state discovery for HMM model selection and learning. *Artificial Intelligence and Statistics*, 492–499.

Singer, H., & Ostendorf, M. (1996). Maximum likelihood successive state splitting. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE.

Trafton, J. G., Hiatt, L. M., Brumback, B., & McCurry, J. M. (2020). Using cognitive models to train big data models with small data. In *International Conference on Autonomous Agents and Multi-Agent Systems*.

Trafton, J. G., Hiatt, L. M., Harrison, A. M., Tamborello, II, F. P., Khemlani, S. S., & Schultz, A. C. (2013). ACT-R/E: An embodied cognitive architecture for human–robot interaction. *Journal of Human–Robot Interaction*, *2*(1), 30–55.