Comparing Modeling Techniques in their Ability to Predict Current and Likely Next Tasks for Cognitive Workers

Sylvain Bruni, Lisa Lucia, Patrick Cummings, Benjamin Ford

Aptima, Inc., Woburn, MA 01801, USA
sbruni@aptima.com

Abstract. The pervasiveness of virtual assistants in everyday life is becoming a natural expectation of users of technology at work. Existing commercial offerings are however solely reactive, thus slow, shortsighted, and narrow in their support. This is inadequate for dynamic, time-pressured, uncertain environments like the military or intensive care units. Digital assistants in such critical settings need to be proactive: They must sense the tasks and workflow of the users, understand their goals and intent, and provide support in anticipation of an explicit request. We present here early exploratory research on leveraging machine learning to accurately identify the current task and workflow and predict the likely next ones based on users' interactions with the system. We conceived, trained, and tested six models in their identification and prediction abilities on a mock administrative use case. The best-performing classifiers predicted the majority of tasks correctly, with about 75% accuracy on average, and were able to achieve this accuracy after training on interaction data from three participants. We additionally present detailed performance results for our K-Nearest Neighbor model, to exemplify key gaps and opportunities for continued research.

Keywords: Cognitive work, task prediction, interactions-as-a-sensor, human-computer interactions, digital assistant, decision-support systems, KNN.

1 Motivation and Approach

Beyond the commercial competitive market, the pervasiveness of virtual or digital assistants in everyday life, from home devices (e.g., Google Home) and online search (e.g., Amazon's Alexa), to wearable, mobile instant support (e.g., Apple's Siri) and desktop assistance (e.g., Microsoft's Cortana), is slowly becoming a natural expectation of users of technology at work [1]. However, such commercial offerings are typically reactive: A human prompt, such as a click, a voice command, or a recognizable action, will trigger processing by the digital assistant that will subsequently yield a response. This process intrinsically creates cognitive issues such as *delays* in getting a correct or exploitable answer, the *shortsightedness* of an automated reaction solely based on limited inputs, and the *narrowness* in performance optimization [2]. These shortcomings may be acceptable for routine home or shopping activities but constitute significant

limitations in the adoption of digital assistants in dynamic, time-pressured, uncertain environments (e.g., in the military or in hospitals' intensive care units).

Resolving these gaps through enhanced proactivity of the automated agents that drive the assistant's behavior offers an encouraging research avenue. Conceptually, we define proactivity as the ability to minimize cognitive overhead (e.g., asking for specific support, refining requests, waiting for answers) through the anticipation of the user's needs. Concretely, we propose to enable proactivity through the following trifecta: *sensing* the tasks and workflow of the users (Principle 1); *understanding* their goals and intent (Principle 2); and *providing* support in anticipation of an explicit request (Principle 3). Earlier efforts have demonstrated the potential for this approach.

Research for the Navy in the domain of multi-uninhabited systems supervisory control has explored how to derive intent from observation by an automated agent of a human planner organizing an air tasking order [3,4]. Flight deck decision-supports were investigated for NASA as a means to provided scalable support depending on the context of operations [5,6]. The characterization of context in critical environments has also been an active topic of research in the information management and mission planning domains [7,8], to derive a formalized representation of user- and mission-related parameters that would influence the behavior of assistive technology. The Air Force recently funded research that combines these advances in the domain of distributed mission planning [9]. And the Army is actively supporting such research in the intelligence domain to accelerate and improve the DCGS-A platform [10].

An in-depth understanding of the set of tasks comprising a workflow for a given role is required to sustain Principle 1 described above. We posit that task definitions should be multi-faceted and include such details as (1) the *user activities* that are undertaken to complete each task; (2) *references* associated with each task (e.g., documents, web sites, software applications, deadlines); and (3) *topic(s)* of interest. Essentially, our approach combines these three components as the basis of knowledge for programmed automation to monitor a user's workstation activity and identify their current task with some level of certainty, for the purpose of serving up actionable access to resources (e.g., software or files used in support of the detected task) or critical information (e.g., a news report on a topic of interest). Ultimately, this approach seeks to increase the efficiency of task completion and improve the quality of the task output.

2 Model Generation and Training

2.1 The Administrative Worker Use Case

To begin development of task models, our team conducted a data collection with a convenience sample of 17 volunteer colleagues. Participants in this effort were directed to perform a set of common administrative tasks intended to be simple enough so that any adult with some familiarity with the Windows operating system and internet browsers could complete them in under an hour. Apart from being asked to use Gmail (for email) and Google Calendar (for event scheduling), the instructions of Table 1 were the only directives. These task instructions were displayed in a web page alongside SRI's Task Assistant [11], a checklist interface for participants to mark tasks as "in progress" once

started, and "completed" when finished. This setup permitted the collection of interaction data (i.e., what software, documents, and website they opened, closed, clicked on and so forth), labeled by task. Such labeled data are necessary to subsequently train and test machine learning algorithms that recognize tasks based on interaction activity.

Table 1. Sample tasks and instructions for the administrative worker user case. Additional tasks include "5-Find Restaurant," "7-Lookup Weather Forecast," "8-Create Event Document," "9-Review Event Document," "10-Send Email to Guest," and "13-Email Travel Form."

#	Task	Instructions
1	Check Email	Read unread emails from the researcher (and follow instructions as they are explained in the email)
2	Read Briefing	Read through the PowerPoint briefing on your desktop; you may be asked questions about it
3	Find Date	Find the soonest date that works for a one-hour lunch for both you and Suzy MacFarland within the next month or so (use the Paptima@gmail.com Google Calendar); add an event into your calendar for this
4	Complete Survey	Complete Survey 1 (link found in email from researcher); open link into new tab in Chrome Browser
6	Fill Out Travel Form	Fill out the Travel Reimbursement Form (on the desktop) with expected expenses for travel to this restaurant
11	Add Event To Calendar	Add time in your own calendar to make a reservation at a restaurant the day before the lunch date(s) [i.e., schedule a reminder for yourself to call the restaurant]
12	Review Travel Form	Review the Travel Reimbursement Form and update it (to list the alternative restaurant and date in the notes section), save it

The tasks in this data collection were intended to mimic the complexity of operational military tasks that intelligence analysts or command decision-makers would undertake: As a result of a triggering event, users would search for, gather, and evaluate information from one or many sources, commit to a decision, and reformat those decisive details for dissemination to others. Furthermore, the tasks were designed to yield variety and overlap in the interactions and activities performed by participants and in the application or program they used to complete the task. Our assumption to justify such a structure in the data collection is that task recognition models trained on such data would be robust enough to distinguish tasks that appear similar in how they are done.

2.2 Topic Modeling and Inference

At the core of our approach is a Topic Model that links workflow to interaction activity by infering a *topic* match between both, thus enabling near real-time sensing and predicting of the tasks performed by the user (Principle 1). The inputs to this topic model are (a) a *workflow* that consists of a list of tasks (and, sometimes, their definitions) and (b) *training data* that consist of user-workstation interaction logs labeled by task. Each log line consists of five items: (1) Time: the unix time of the action; (2) Program Type:

the program being used (e.g., text document, spreadsheet, browser); (3) Document Name: the title of the open file or url for browser applications; (4) Action Type: category of action (e.g., key press, mouse click, mouse over); and (5) Action Value: value associated to that action (i.e. what was clicked, typed, or hovered over).

To convert these values to appropriate inputs to a classifier, we convert each input to a vector using the following methods. For the Program Type and Action Type variables we use one-hot encodings of each of the programs seen within the data. For the Document Name and Action Value components we use Latent Dirichlet Allocation (LDA), after some preprocessing, to vectorize each item. That is, we process the data by removing all stop words (e.g., the, and, a, then, as, etc.), then removing all digits and numbers from the set, and finally removing any non-English words unless they occur greater than N=50 times in the dataset (this discards miscellaneous urls or hyperlinks, but keeps commonly used ones that might not be in an English dictionary). After this processing, we generate topics via LDA based on each processed row within the dataset. Finally, the time field is used via an exponential recency-weighted average to provide predictions based on the previous set of actions, as opposed to providing predictions based on one action in isolation.

2.3 Task Assessment and Prediction

The task classifier uses the vector described in 2.2 as an input, concatenating one-hot action and program types with title and value topic distributions. Its output is a probability distribution of task prediction. We built our task prediction to be classifier-agnostic, and therefore tested it with a set of different models. The following were all trained via skLearn [12]. Those models are: K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Extra Trees Classifier (ERT), and Logistic Regression Cross-Validation (LogisticRegressionCV). Since these are all well documented, out-of-the-box models, we omit the details of their implementations from this paper.

3 Evaluation: Comparing Models' Performance at Predicting Current Task

3.1 Method

Data Preprocessing and Grouping. As users interact with the system, interaction data are streamed continuously to the model. Looking at individual interaction data points in isolation misses a richer *interaction history* that provides necessary context for making accurate predictions. As such, we consider all interaction instances within the last 30 seconds (i.e., a window) when predicting the user's current task. For each interaction window, we (1) find k-nearest labeled actions and use each to "vote" for the classification of a new action (via probability distribution; predict_proba from skLearn), and then (2) average all those classifications with an exponential recency-weighted average to generate a more accurate estimate. The result is a probability distribution table of all the tasks, i.e., the model's normalized prediction of what the user is currently working

on. We consider the task with the highest probability to be the user's current task (ties are broken randomly). Finally, although we did an evaluation on models where we pruned their feature space with feature selection and also conducted hyperparameter tuning with stratified k-fold cross-validation, these evaluations did not yield significant improvements and thus they are omitted from this paper.

Baseline Models. In addition to the models discussed in Section 2.3, we evaluated the performance of two baseline prediction models. While these baseline models are not meant to be used in a task prediction system, they serve as an appropriate lower bound for our expectations of performance; if an advanced model cannot significantly outperform a baseline model, the advanced model is not suitable for use. These baseline models are uniform random (i.e., uniformly randomly predict a task) and majority class (i.e., always predict the majority task seen in training data, for the testing data).

Evaluation Process. We employed a standard, cross-validation technique, using the 17 separate data sets from the data collection (one per participant). We trained and tested the model 17 times with 17-fold cross-validations (16 trained, 1 tested, repeated for all test data). The primary metric in this evaluation is accuracy (ratio of correct task predictions to total number of trials); we also present a confusion matrix for one of the top-performing models. Prior to each evaluation we removed consecutive duplicate entries as well as filtered a subset of actions. In particular, we removed all actions that took place in the task checklist tool (since this was purely for task labelling) as well as any desktop level actions occuring between tasks. The evaluation was conducted in Python [13], using NumPy [14], Pandas [15], skLearn [12], and Matplotlib [16] libraries.

3.2 Results

Overall Results. Fig. 1 shows the performance of each model by test participant. The KNN, ERT, and Logistic Regression CV models perform similarly on average, and better overall than the baseline models. Unsurprisingly, the Uniform Random and Majority models perform poorly, regardless of data set (participant). While a uniform random model would have an accuracy of nearly 50% in a binary classification setting, this multi-class classification setting is significantly more difficult. As such, it is harder to achieve a ~75% accuracy than it is in a binary classification problem.

Individual Results for KNN. KNN's average accuracy across all participants was 71.4%. While slightly lower on average than ERT (75.3%) or Logistic Regression (75.7%), we report detailed results for this method because it exemplifies key research questions of interest for our approach. The confusion matrix for KNN is shown in Fig. 2. In such a matrix, rows correspond to the ground-truth task label while columns correspond to predictions by the model. Each cell shows the percentage of predictions that correspond to a ground-truth label aggregated across all data sets. The title indicates the total number of predictions (i.e., trials) and the average accuracy.

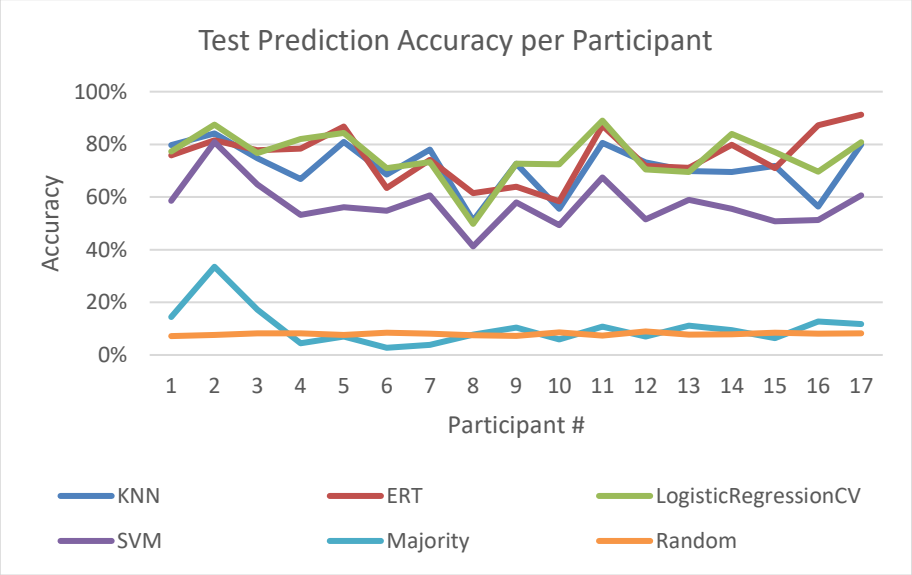

Fig. 1. The accuracy rate (% correct, y-axis) of explored models is plotted against that of baseline defaults (random and majority) per participant (x-axis). These data show that, for the most part, the explored models perform relatively similarly and significantly better than baselines.

While KNN performed very well on some of the tasks (achieving an accuracy of up to 84.8%), other tasks were not classified well. For example, the Review Travel Form task, where KNN achieved an accuracy of 26.2%, was more commonly misclassified as the Fill Out Travel Form task (62.5% of the time). It is interesting to note that these two tasks appear similar to each other, and that the correct task is not usually misclassified as any other task (i.e., misclassifications are concentrated in one other class). While this appears to be the case for other task pairs that seem similar to each other (e.g., Add Event to Calendar, Find Date), Check Email appears to be one task where its misclassifications are distributed across two other tasks: Read Briefing and Complete Survey. Indeed, these three tasks do share important similarities: When users are completing the Read Briefing and Complete Survey tasks, they need to check their email first; the temporal dependencies between tasks, while not explicitly modeled, still appear to have an impact on classfiers' abilities to distinguish between temporally related tasks. Finally, while Review Travel Form is commonly misclassified as Fill out Travel Form (62.5% of the time), the reverse is not true. Fill out Travel Form is only sometimes misclassified as Review Travel Form (15.5%). This asymmetry may suggest that there is an asymmetry in task similarity. Review Travel Form may be more similar to Fill Out Travel Form than Fill out Travel Form is to Review Travel Form.

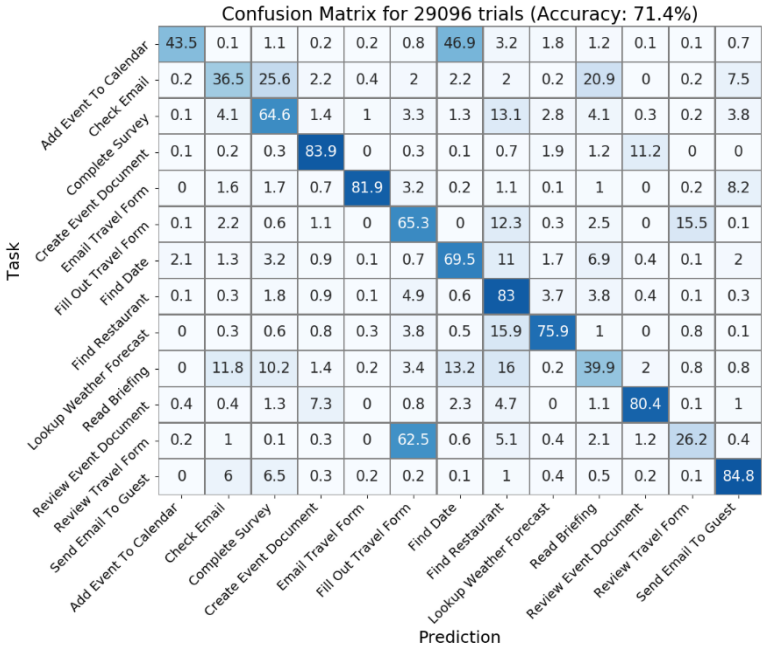

Fig. 2. Confusion matrix for the cross-validation results of the KNN model. The diagonal represents the rate that the correct task was predicted for each task. Other cells show the rate that a different task was predicted for each task. Darker shades of blue indicate higher accuracy rates.

Exploratory Evaluation with KNN. Because some users expressed privacy-related concerns regarding activity logging, we repeated our analyses while purposely omitting some inputs. Results are shown in Fig. 3. Note that we limited this exploratory analysis to KNN. Firstly, disabling keylogging yields a significant drop in average accuracy to 33% (from 71.4%). Secondly, as many military users tend to be restricted to Windows-based operating systems and Microsoft products like Internet Explorer or Edge internet browsers, our models may not be able to leverage browser-related inputs since those have only been implemented as a Chrome Extension. With this limitation, average model accuracy drops to 35%. Finally, we investigated what amount of data was required to produce a suitable, "good enough" model. We ran a series of tests (again using cross-validation in each step) where we trained with more data each time to observe the effects on accuracy. Results are plotted in Fig. 4. Accuracy plateaus near 75% but, somewhat surprisingly, it reaches that accuracy after only three training sessions. This suggests that these task recognition training methods will approach suitable accuracy after learning from three sessions in which individuals complete their entire workflow. Also worth noting is the fact that this quick learning model achieved suitable accuracy with data from three separate individuals performing the same tasks. The benefit to this is that it seems likely that, if individuals used different sets of actions to accomplish the same tasks, the model was still able to capture this sufficiently.

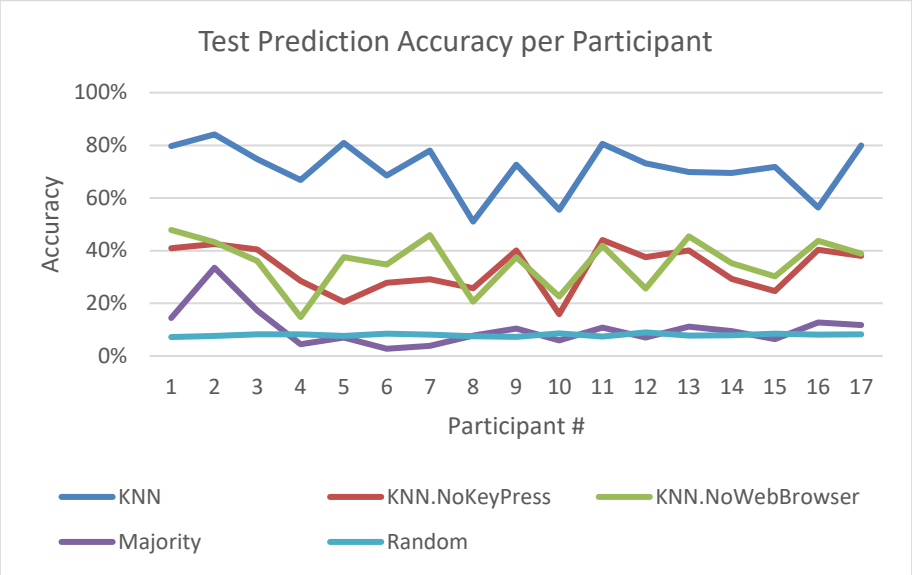

Fig. 3. The accuracy rate (% correct, y-axis) of our full KNN model is plotted against that of KNN without key press logging (KNN.NoKeyPress), KNN without web browser logging (KNN.NoWebBrowser), and baseline defaults (random and majority) per participant (x-axis). These data show that, without exception, having the full monitoring capabilities results in far superior prediction accuracy.

4 Conclusion and Next Steps

With the exception of a few tasks that were hard to classify, the best-performing classifiers predicted the majority of tasks correctly (approximately 75% accuracy on average). Additionally, this performance was achieved after the model was trained on few (three) workflows. These initial exploratory analyses only constitute a preliminary foray into the domain, and we have identified the following next steps.

First, the dataset may benefit from a feature extraction step. Similar to how we use LDA for topic modeling to generate more features, we could also experiment with more temporally-explicit features such as "User's Previous Action" or "Time Spent by User on Previous Action." Such features will introduce dependencies between the samples in the training data, and it remains to be seen whether those will help or hinder the training process. Additional features can help improve the similarity and dissimilarity characterization of the tasks; if the overall dissimilarity of the tasks increases, it may make the classification problem easier and thus improve performance. Another variation could reorganize training such that each "sample" is actually a sequence of N samples. Doing so would change the classification task to "predict the user's task based on this sequence of actions they took." This would add a significant amount of variance into the dataset, which may also help or hinder the model training process. If there is too much variance/noise in the training data, the classifier may overfit to that noise instead of the underlying relationships (i.e., the classic bias-variance trade-off).

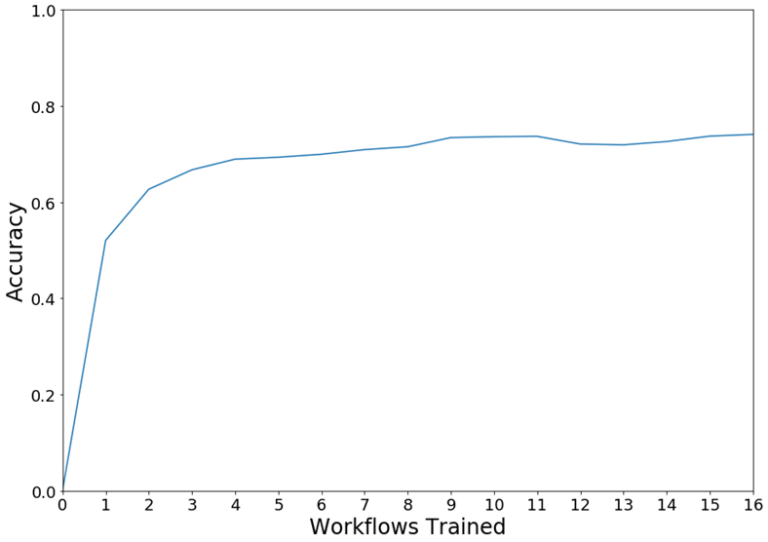

Fig. 4. In a series of cross-validation tests, we demonstrate that accuracy plateaus near 75% after training with workflow data from three participants (i.e., three individual sessions).

Additionally, we must recognize the limitations of this approach. Our data collection was structured in a way to reduce much of the natural variability that occurs when users conduct tasks. This structure resulted in data distributions that were more similar to each other than would be the case in an unstructured or real-world setting. A much larger collection is thus needed so that the data distribution is representative of the underlying ground truth. Such an increase in sample size would also permit testing more data-hungry methodologies, like deep learning (e.g., convolutional neural networks).

Finally, it is critical that interventions by a proactive assistant based on such task identification also account for their likely disruptive effect. We intend to enable cognitive state analysis as a means to tailor the modality, timing, and format of interventions to prevent a worsening of the user's cognitive state. Recent work has shown some promise leveraging interactions in the identification of engagement vs. boredom [17], stress [18,19], and fatigue [20]. Such quantification of a user's cognitive state acts, in turn, as additional data for the assistant to tailor its interventions for maximal effect.

References

1. Hoy, M. B.: Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants. Medical Reference Services Quarterly. 37 (1): 81–88 (2018).
2. Langdorf, J.: Three challenges facing virtual assistants. VentureBeat AI (2016).
3. Bruni, S., Riordan, B., Schurr, N., Cooke, N., Freeman, J., Roberts, J., & Rima, N.: Designing a mixed-initiative decision-support system for multi-UAS mission planning. Proceedings of the 55th Annual Meeting of the Human Factors and Ergonomic Society (HFES), Las Vegas, NV (2011).

4. Riordan, B., Bruni, S., Schurr, N, Freeman, J., Ganberg, G., Cooke, N., & Rima, N.: Inferring user intent with Bayesian inverse planning: Making sense of multi-UAS mission management. Proceedings of the 20th Behavior Representation in Modeling and Simulation Conference (BRIMS), Sundance, UT (2011).
5. Bruni, S., Chang, A., Carlin, A., Swanson, L., & Pratt, S. (2012). Designing an adaptive flight deck display for trajectory-based operations in NextGen. Proceedings of the 4th International Conference on Applied Human Factors and Ergonomics, San Francisco, CA.
6. Bruni, S., Chang, A., Carlin, A., Marc, Y., Swanson, L., Pratt, S., Mizrahi, G. : Patent US9293054B2 Systems and methods to react to environmental input (2013).
7. Bruni, S.: Characterizing Mission and User Context for Proactive Decision Support, at the Approaches to Context-Based Proactive Decision Support symposium. Invited panelist at the annual meeting of the Human Factors and Ergonomics Society, Washington, DC (2016).
8. Brown, J., Bruni, S., Bennett, J., Fournelle, C., Hanna, C., Lucia, L., Ward, D. & Woodward, B. (2016). Characterizing Mission and User Context for Proactive Decision Support. Proceedings of the Human Factors and Ergonomics Society Annual Meeting (HFES), Vol 60, Issue 1, pp. 228–232, Washington, DC (2016).
9. Bruni S., Lucia L. : Patent-Pending USPTO #62/617,028 Human-System Interactions As A Sensor For Performance Optimization (2017).
10. Bruni, S.: Why Automated Agents Should Be More Like Alfred Pennyworth. Proceedings of the 2018 Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), Orlando, FL (2018).
11. Peintner, B., Dinger, J., Rodriguez, A.C., & Myers, K.L.: Task Assistant: Personalized Task Management for Military Environments. IAAI (2009).
12. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-Learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830 (2011).
13. van Rossum, G., et al.: The Python Language Reference, Python Software Foundation. http://docs.python.org/py3k/reference/index.html, last accessed 05/15/2019.
14. van der Walt, S., Chris Colbert, S., Varoquaux, G.: The NumPy Array: A Structure for Efficient Numerical Computation. In: Computing in Science & Engineering, vol. 13, pp. 22–30 (2011).
15. McKinney, W.: Data Structures for Statistical Computing in Python. In: Proceedings of the 9th Python in Science Conference, pp. 51–56 (2010).
16. D. Hunter, J.: Matplotlib: A 2D graphics environment. In: Computing in Science & Engineering, vol. 9, pp. 90–95 (2007).
17. Bixler, R. & D'Mello, S.: Detecting Boredom and Engagement during Writing with Keystroke Analysis, Task Appraisals, and Stable Traits. International Conference on Intelligent User Interfaces (2013).
18. Vizer, L. M.: Different strokes for different folks: individual stress response as manifested in typed text. Computer Human Interaction Conference on Human Factors in Computing Systems (2013).
19. Rodrigues, M., Goncalves, S., Carneiro, D., Novais, P., & Fdez-Riverola, F.: Keystrokes and Clicks: Measuring Stress on E-learning Students. Advances in Intelligent Systems and Computing, 220: 119-126 (2013).
20. Epp, C., Lippold, M., & Mandryk, R. L.: Identifying Emotional States using Keystroke Dynamics. Computer Human Interaction Conference on Human Factors in Computing Systems (2011).