Detecting stress level through computer usage pattern

name1 EMAIL1
school addy

name1 EMAIL1
school addy

Abstract

Stress is inevitable. Sometimes people fall back to their old habits or perform a series of actions that can be directly linked to their stressful states. Identifying stress is important but the approach scales poorly when invasive sensors have to be used or the method relies on induced stress in a timed lab experiment. Our solution aims to identify stressful states based on continuous logging of keystroke dynamics, mouse patterns, and foreground application usage. We propose a privacy-aware system, E-stress detector, that logs participants' computer activities alongside their stress self report. After extracting relevant features, our top classifiers provide 85 to 95 % accuracy for predicting user future stress level based on historical data.

1. Introduction

As we know, overstress undermines our productivity, and more seriously, it might bring harm to our health, even cause mental diseases. However, sometimes we tend to feel exhausted and stressed, but unaware of what cause us to feel stressed exactly. Therefore, it is crucial to find out the sources of stress, and at some appropriate time, give users some reminders that you are overstressed and you should take a break and relax yourself right away.

When people are under stress, their behaviors tend to change accordingly. In particular, we have observed that peoples computer using patterns, the way they type, click, browse webs change with regard to their stress, or anxiousness level. For example, when people feel stressed, they click the mouses with higher frequency; they tend to have more typos, and thus the frequency of doing error correction raises; the frequency of switching between different web pages also increases; they might check their phones more frequently but unconsciously, etc. We suppose these are good indicators to users stress level, and by analyzing users baseline stress level and detecting abnormal stress level, we will be able to understand whether they are overstressed and give proper reminders.

Since the computing age has encouraged working on personal computers, stress detection correlated to usage of these electronic devices could lead to insightful results. This research aims to discover which computer activities, if any, correlate with individual stress levels. Can ones typing pattern, duration on a browser, rate of mouse clicking or overall computer usage be linked to their stress level? Insightful results could have multiple applications such as personalized health recommendation systems for handling stress, timely interventions for detecting stress at an individuals peak, holistic feedback for doctors during diagnosis, among other health driven and personal purposes.

2. Related Work

Computer activities such as keystroke dynamics and mouse click patterns have been used in detecting user emotional and stress states. Related works can be broadly grouped into three major categories: Affective Computing, Keystroke Dynamics, and Mouse Usage.

2.1. Affective Computing

Epp Clayton et al, arguably did the first major work connecting keystroke dynamics and emotional states. This research measured 15 emotional states of users using periodical self report and keystroke features throughout the day (Epp et al., 2011). Hong proposed StressSense, which can detect human stress by focusing on paralinguistic features instead of actual vocal

Appearing in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012. Copyright 2012 by the author(s)/owner(s).

contents (Lu et al., 2012). StudentLife utilized SurveyMonkey and mobileEMA to measure participants stress as ground truth (Wang et al., 2014). Similarly, we will use the same approach as our ground truth. Instead of 15 emotional states, our research primarily focuses on user's level of stress.

2.2. Keystroke Dynamics

Hernandez, Javier, et al in Under Pressure use a pressure sensitive keyboard and capacitive mouse to discriminate between stressful and relaxed states in a lab setting (Hernandez et al., 2014). To allow our application to scale easily, we do not rely on using custom hardware.

2.3. Mouse Usage

Early work done by Ark Wendy S. et al in The Emotion Mouse relate the mouse touch to the emotion attached to a computer task. GSR and chest sensors were used to collect heart rate, temperature, galvanic skin response (GSR), and somatic movement, which were measured against the six Ekmans facial emotions (Ark et al., 1999). In more recent work, Sun, David et al in MouStress proposed an approach to detect stress from mouse motion, but from a physiological perspective— a model of the arm (Sun et al., 2014). Our approach is less intrusive as we do not propose an external sensor; rather, we focus on non-invasive continuous mouse tracking activites such as click rate and coordinates.

2.4. Computer Applications

Karpathy on his blog shows how he measures computer activities for self tracking. However, this is done with the idea of quantified self but without linking user stress state. Some of the ideas used in our project design were inspired by work on Karpathy's system (Karpathy).

Our work differs from all other works in 3 major ways: a custom hardware does not need to be installed; data collection is not done in a stress-induced lab setting for a short period of time, rather data is continuously logged with the intent of long-term continuity; as far as we know, we are the first to incorporate computer usage application as part of stress sensing.

Unlike previous works, which focus on only keystrokes, only mouse, or both, our work leverages foreground application usage as an extra data source. This is based on the intuition that certain applications might immensely affect user emotional state compared to others. For instance, a non-programmer using Matlab for the first few days might hit the backspace key to delete wrongly written code; click on the menu buttons several times, perhaps in search of a functionality or switch applications multiple times between Matlab software and a browser page showing "how to perform matrix multipication in Matlab".

It is worthwile to note that previous works have logged user computer activities without explicitly discussing how user data is protected; our application takes a different approach by only logging specific keys when users type. Every other key is blocked with the symbol "$$". For instance, alphabetic and numeric keys are logged as "$$" as a user could type highly sensitive information such as passwords or personal messages. In addition, every data logged is not uploaded to a cloud server; the data is locally stored on the user's computer. This way, a user can decide to opt out of the data logging process and delete all data collected.

3. Methodology

The work involves two major components: the data logging process done in-situ – as participants carried out their daily activities, and the data processing required to classify stress states. For the data collection, users' keystrokes, mouse activities, and computer applications usage are timestamped and logged. The data processing involved extracting relevant keystroke features in order to build Machine Learning classifiers.

3.1. Experience Sampling of Computer Activities

Experience Samping Method (ESM) involves continuously logging computer activities while periodically collecting user responses to self-reports of their level of stress (Hektner et al., 2007). Stress states were measured using Photographic Affect Meter (PAM) (Pollak et al., 2011)and three-question likert-scale survey.

Figure 1 shows the PAM survey. During periodic interval—we arbitrarily select 30 minutes interval—a webpage automatically opens up and a user is asked *"How do you feel right now?"* Then the user selects one of 16 grid pictures that reflects their internal emotional state.

Figure 2 shows a self-report questionnaire. After selecting a picture in PAM, a 3-question survey page opens up where the user gets to pick one response out of five categories.

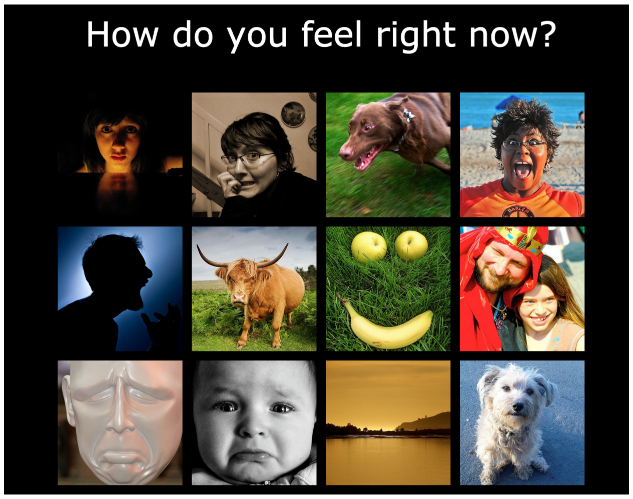
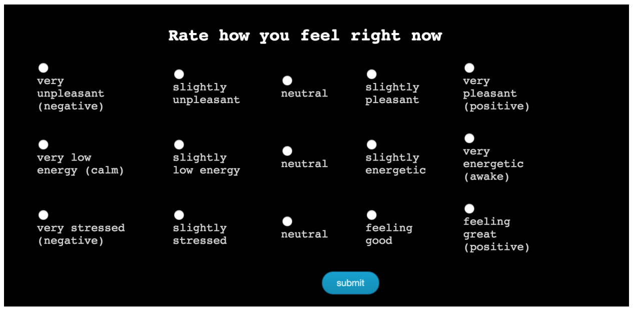


Figure 1. PAM: the user selects one of 16 grid pictures that reflects their internal emotional state.



Rate how you feel right now

● very unpleasant (negative)	● slightly unpleasant	● neutral	● slightly pleasant	● very pleasant (positive)
● very low energy (calm)	● slightly low energy	● neutral	● slightly energetic	● very energetic (awake)
● very stressed (negative)	● slightly stressed	● neutral	● feeling good	● feeling great (positive)

submit

Figure 2. A 3-question survey: the user gets to pick one response out of five categories for each question.

3.2. Data Collection Software

A system was built that logs user keystrokes, mouse activities and applications used throughout the day. While there are a few open sourced tools that could have been adopted, a custom application was built because existing tools suffer from poor documentation and incomprehensible programming approaches.

Upon installation, our software tool contains a keystroke logger, a mouse logger, an application logger and a self-report logger. The keyboard and mouse applications continuously logs keystrokes and mouse activities; the foreground application is logged every 3 seconds; while the stress self-report is logged every 30 minutes. The self-report application starts up a PHP server, opens PAM webpage with the user's default browser. Upon selecting a choice from the 16-grid pictures, the response is saved in a file. Thereafter, the 3-question survey is opened as the next webpage, which upon answering a "thank-you" page loads. After about 5 seconds, the "thank-you" page automatically closes. Self-report application logs user response based on when they responded and not when the page opens. For instance, if the Web page opens at 4pm but the user responds at 7pm, then the response is logged at 7pm.

Every 15 minutes, a background script checks if all the applications are running or if they have been stopped either manually by the user, another running process, or through a power down of the user computer. This ensures that user logging continues non-intrusively as users do not have to manually start the applications if they were shut down. In order to maintain user access control, every activity application has an icon that appears in the dock so that the user can close the application whenever they want to; users can also delete the data directory if they don't want to partake in the experiment. These intervals for each application were arbitrarily chosen.

mouse logging format 2015-05-03 12:18:55.438360 — MouseMoveHooker — NSMouseMoved — y=606.19140625 x=683.0390625 2015-05-03 12:19:51.473070 — MouseButtonHooker — NSRightMouseUp — y=22.234375 x=274.37109375 2015-05-03 12:19:55.261508 — MouseButtonHooker — NSLeftMouseUp — y=272.79296875 x=293.51953125 2015-05-03 12:19:53.588968 — MouseButtonHooker — NSLeftMouseDown — y=81.90234375 x=109.9609375

keyboard logging format

application logging format

pam logging format

ema logging format

Using our custom applications, we will install our background scripts on the participants computers. Ideally, our background programs will record the keys users type, but the corresponding values will be randomized due to privacy sensitivity, and the frequency of mouse clicking as well. Data for Web browsing and application usage will be collected through available python APIs. All the data will be stored as text files awaiting further analysis.

3.3. Feature Extraction

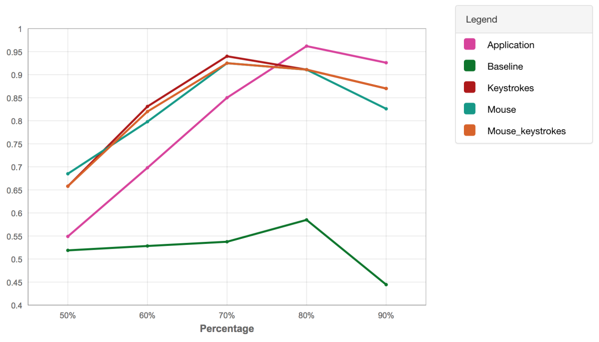
We segmented the raw computer usage activities with sliding window size = 1 minute and the extracted three types of computer usage features, **M**, **K**, and **A** for mouse clicks, keystrokes, and application usage within each sliding window. $\mathbf{M} = [d_m, r_m, t_m]$, where d_m is the total distance of mouse movement, r_m is the mouse click rate, and t_m is the average mouse button pressed duration. $\mathbf{K} = [r_{k_1}, t_{k_1}, r_{k_2}, t_{k_2}, t_{k_i}, ..., r_{k_N}, t_{k_N}]$, i $\in N$, where k_i stands for different keys and N stands for total number of key types we logged, and $N = 9$ in our case, including function keys, letter keys, enter key, tab key, space key, delete key, escape key, arrow key, and all the keys that were pressed (that is, the previous key types were also taken into account), and t_{k_i} is key pressed time for key k_i. $\mathbf{A} = [t_{a_1}, t_{a_2}, t_{a_j}, ..., t_{a_P}, x]$, j $\in P$, where a_i stands for different applications, P is the total number of applications users might open when using computers, t_{a_j} is the time users spent on application a_j, and x is the number of active applications within each time window. To know decide what P is, we first iterated through the users' historical application usage logs.

3.4. Algorithms

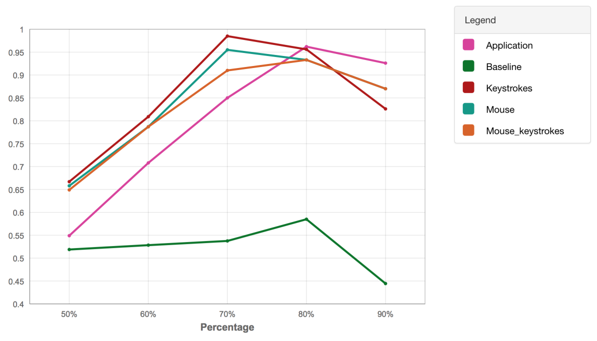
Since the self-report survey popped up every 30 minutes, we only considered raw computer usage features within the 30-minute duration, if the computer was active, before the survey is filled out. That is, assuming the user filled out the survey at timestamp T_end, and the computer got active at timestamp T_c, we only considered raw features with timestamps in $[T_{begin}, T_s)$, where $T_{begin}{=}min(T_{end-1800}, T_c)$. And according to the previous section, we extracted high level features from the raw features among each period and got computer usage samples $S_{T_{begin}}, S_{T_{begin+60}}, ..., S_{T_{end-60}}, S_{T_{end}}$, where each sample S was labeled with the corresponding binary results of users' stress level based on the participants' EMA self-report score. Then we used Decision Tree, Random Forest, support vector machine (SVM), as well as participants' historical data for future stress prediction.

4. Results

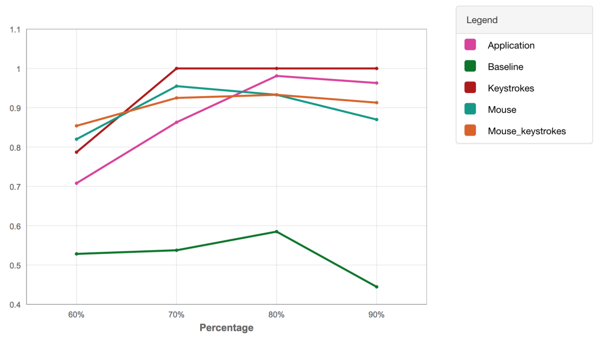
Due to time limit, we could only collect data from one participant for consecutive 3 days after the deployment, and we got 352 samples from the deployment. We tried different ways of splits for training and testing, from 50-50 split to 90-10 split, as well as different combinations of features (M, K, A, $M + K$). But we found that the prediction accuracy was very low using only A, because during each time window, the participant tended to open just a couple of applications, which made the feature vector of A become very sparse and in turn diminished A's ability for prediction. As a consequence, we decided not to consider all the applications users might use as features. Instead, we only took into account the top 5 frequently used



a. Decision Tree



b. Random Forest



c. SVM

Figure 3. The accuracy of stress prediction with different algorithms: (a) Decision Tree, (b) Random Forest, (c) Support Vector Machine (SVM) and with different ways of data splits.

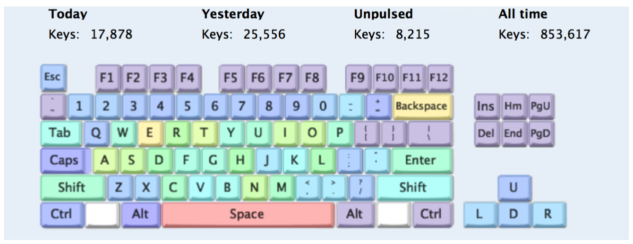


Figure 4. Keyboard Heatmap

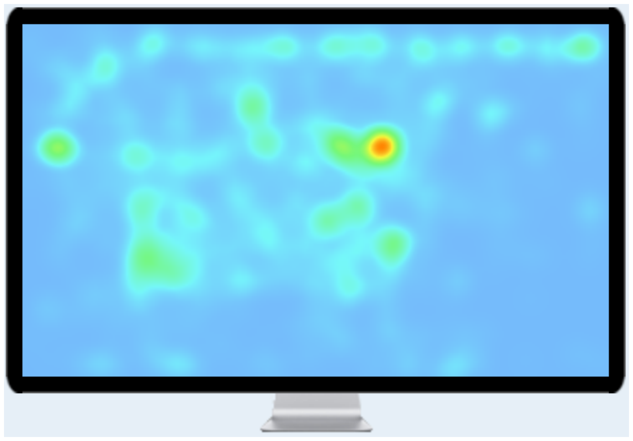


Figure 5. Mouse Keystrokes

applications as features. That is, the application usage features became $A' = [t_{a'_1}, t_{a'_2}, t_{a'_3}, t_{a'_4}, t_{a'_5}, t_{a'_6}, x]$, where $t_{a'_1} \sim t_{a'_5}$ are the amount of time users spend on the top 5 most frequently used applications during each 1-minute window, and $t_{a'_6}$ is the amount of time users spend on the rest of the applications. After the features were adjusted, the accuracy significantly improved. Figure 3 is the accuracy of stress prediction with different algorithms, different combinations of features, as well as different ways of data splits. For the baseline algorithm, it randomly guesses whether the user is stressed or not. The reason why the split for SVM is only down to 60-40 is that the participant's self-report score is quite consistent during the first few days of study, where the participant kept reporting "not-stressed" among the first half of the data we collected. Thus, we would have no training data labeled with "stressed" if we did 50-50 split.

In general, the algorithms achieve the highest accuracy when 70% $\sim$ 80% data is used as training data, and the accuracy goes down if more data is used as training data. We have to collect more data and calculate their p-values in order to know what kind of data splits can really achieve the best accuracy and yet the result it obtains is statistical significance, since it is a must for the system to know to how many historical data it has to acquire in order to predict future stress level correctly for the next whole day.

Initially, we assumed that the more features we used, the better accuracy we would get, but it turned out we got worse accuracy if we built the models using multiple features at the same time, $M + K$ for instance. According to Figure 3, the models trained with $M+K$ performed worse than models trained only with single feature during most of the time. Our speculation is that owning to the limited amount of data, our models tend to over-fit the data if too many features are given. On the other hand, we also noticed that different algorithms perform differently given different types of features. For example, Decision Tree gives the highest prediction accuracy with application usage features, whereas Random Forest somewhat does better with keystrokes features. Therefore, for our next step, we are considering using ensemble methods, combining all these different models together and generating the final prediction with the linear combination of the results given by individual algorithms, where each algorithm is only trained with single type of features with which it is most discriminative.

Modalities		M			K		
Algorithms	DT	RF	SVM	DT	RF	SVM	DT
50%	0.649	0.676	0.631	0.658	0.631	0.658	0.703
60%	0.809	0.787	0.798	0.843	0.809	0.831	0.764
70%	0.821	0.896	0.940	0.955	0.940	0.940	0.970
80%	0.911	0.911	0.956	0.978	0.956	0.911	0.956
90%	0.783	0.870	0.913	0.913	0.826	0.913	0.913

4.1. Limitations

Our dataset is limited. Since recruited participants were PhD students, our prediction model performance cannot be generalized to a broader audience. Since self-report surveys are subject, the definition of stress for different participants will have varying likert scale values.

Survey popping up periodically can become annoying to users thereby making reducing performance over time. Since emotional states are short-lived and a user can have multiple emotions in a short time, the surveys cannot fully capture the end user's entire emotional states but only a fraction of it

5. Future Work

In order to improve multimodal sensing, future work will involve external sensors for measuring HR, BP, EEG. The current application focuses on batch learning of data already amassed so for next iteration will involve online learning especially for making efficient recommendation systems. Participants for this experiment were PhD students recruited from a research lab familiar with some of the measuring tools used; more insightful applications might involve workers in industries especially end users oblivious of the methods of the current research.

do some form of active learning

collect data for a longer period of time — like several months

5.1. Conclusion

Acknowledgments

Special thanks to Prof Daniel Cosley, students of INFO 6010, reviewers of Advanced Machine Learning course (Spring 2015), members of the PAC lab at Cornell University Information Science, and Prof Tanzeem Choudhury, whose guidance was instrumental in this project.

References

Ark, Wendy S, Dryer, D Christopher, and Lu, Davia J. The emotion mouse. In *HCI (1)*, pp. 818–823, 1999.

Epp, Clayton, Lippold, Michael, and Mandryk, Regan L. Identifying emotional states using keystroke dynamics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 715–724. ACM, 2011.

Hektner, Joel M, Schmidt, Jennifer A, and Csikszentmihalyi, Mihaly. *Experience sampling method: Measuring the quality of everyday life*. Sage, 2007.

Hernandez, J, Paredes, P, Roseway, A, and Czerwinski, M. Under pressure: Sensing stress of computing users. *Proc. of Computer and Human Interaction*, 2014.

Karpathy, Andrej. Quantifying productivity.

Lu, Hong, Frauendorfer, Denise, Rabbi, Mashfiqui, Mast, Marianne Schmid, Chittaranjan, Gokul T, Campbell, Andrew T, Gatica-Perez, Daniel, and Choudhury, Tanzeem. Stresssense: Detecting stress in unconstrained acoustic environments using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 351–360. ACM, 2012.

Pollak, John P, Adams, Phil, and Gay, Geri. Pam: a photographic affect meter for frequent, in situ measurement of affect. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 725–734. ACM, 2011.

Sun, David, Paredes, Pablo, and Canny, John. Moustress: detecting stress from mouse motion. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pp. 61–70. ACM, 2014.

Wang, Rui, Chen, Fanglin, Chen, Zhenyu, Li, Tianxing, Harari, Gabriella, Tignor, Stefanie, Zhou, Xia, Ben-Zeev, Dror, and Campbell, Andrew T. Studentlife: assessing mental health, academic performance and behavioral trends of college students using smartphones. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 3–14. ACM, 2014.