Providing Contextual Assistance in Response to Frustration in Visual Analytics Tasks

Prateek Panwar* Adam Bradley† Christopher Collins‡

University of Ontario Institute of Technology

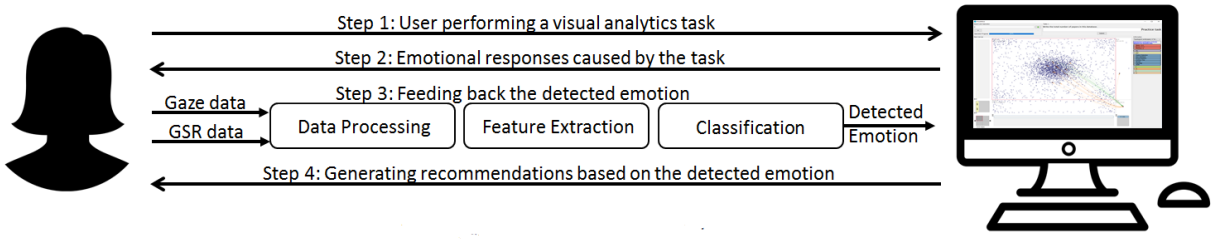

Figure 1: Work flow of the proposed model — appropriate assistance is provided based on detection of frustration state.

Abstract

This paper proposes a method for helping users in visual analytic tasks by using machine learning to detect and respond to frustration and provide appropriate recommendations and guidance. We have collected an emotion dataset from 28 participants carrying out intentionally difficult visualization tasks and used it to build an interactive frustration state detection model which detects frustration using data streaming from a small wrist-worn skin conductance device and eye tracking. We present a work-in-progress design exploration for interventions appropriate to different intensities of frustrations detected by the model. The interaction method and the level of interruption and assistance can be adjusted in response to the intensity and longevity of detected user states.

Index Terms: Human-centered computing—Visualization—Visualization theory, concepts and paradigms; Human-centered computing—Human computer interaction (HCI)—Interaction paradigms

1 Introduction

Information visualization (InfoVis) helps users to understand trends and patterns in big data. There are common analytic workflows which have been shown to be effective in analysis. Novice analysts, or even experts faced with new interfaces or datasets may find analysis challenging, requiring high cognitive attention [10, 22]. Some of the significant challenges are: (1) Dealing with a complex dataset involving multiple variables; (2) Using a new or unfamiliar dataset where the nature of the data is unknown; (3) Using a new tool or merely a new visualization. Any of these cases requires extra effort from the user which results in a high cognitive load. Moreover, working in a real world scenario where things like deadline pressure, work stress, and personal life crises are a normal part of work life and can affect a user's performance [5], it often becomes difficult to stay focused while solving these data analysis tasks.

Furthermore, getting stuck in a difficult visual analytics task for a long period could induce a range of negative emotions such as frustration and anger; which would increase the likelihood of disengagement. For example, if a user is new to a particular visualization tool and encountered problems while doing a task due to the complex user-interface (UI), then the user may experience a range of negative emotions because this complexity is blocking the workflow and distracting his attention. Therefore, after a certain amount of time the user will feel demotivated and ultimately disengage [1].

There are many existing visualization tools available which help in facilitating the data analysis process such as Tableau and Microsoft Power BI, as well as many custom-made solutions for specific scenarios and problems, and solutions reported in the visualization literature. Using these software tools could be frustrating for users, either due to the many interface features or due to the dataset and analysis tasks. We are interested in the potential for support systems which would provide users with meaningful recommendations or guidance to help them to overcome data- and task-related issues and hence, prevent disengagement.

Providing meaningful recommendations in a visual data analytic tools is a challenging task because: (1) Recommendation systems need to know when it is the right time to intervene as continuously providing help may cause distraction; (2) If a user is stuck and feeling frustrated, the system doesn't know the cause of the frustration and how to fix it (the same emotion can occur for different contexts: interface-related, data-related or external factors); (3) System cannot predict the intensity of an emotion and therefore, fail to vary its intrusiveness accurately. Traditional interaction is single-initiative (one-sided). That is, users provide explicit input, and then system responds. This work argues that to build a smart recommendation system, there is a need for leveraging mixed-initiative interaction (bi-directional interaction) approach so that the system could understand implicit cues from the user and react accordingly.

A user's emotional state can be detected by capturing data from physiological sensors (e.g. electroencephalogram (EEG), electrocardiogram (ECG) and, skin conductance) or physical sensors (e.g. facial expression, speech, body posture and gaze tracking) or a combination of both. When selecting between various biometric sensor options, we prioritized those which were less intrusive and less likely to cause discomfort. We decided to use the combination of a galvanic skin response device (GSR), and an eye tracker to measure arousal and valence. We recorded skin conductance from the GSR device and gaze location and pupil diameter from the eye tracker.

It is likely that in analytic tasks the detected features (e.g., gaze scan-paths) will be different than when reading a text, looking at an image, or playing a video game. Visual analytics requires visual

*e-mail: prateek.panwar@uoit.ca
†e-mail: adam.bradley@uoit.ca
‡e-mail: christopher.collins@uoit.ca

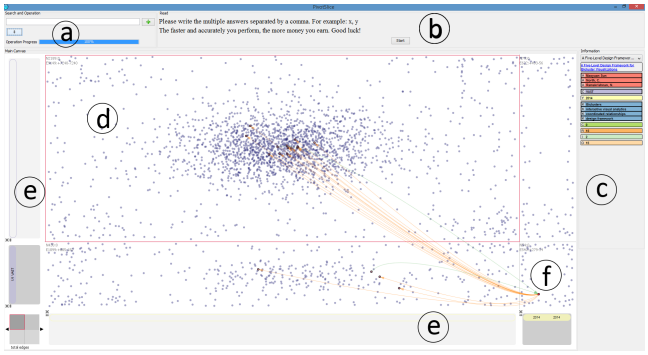

Figure 2: The PivotSlice visualization tool [26]. The interface is divided into six sections: *(a)* search panel, *(b)* task panel, *(c)* information panel, *(d)* unfiltered area, *(e)* filter axes, and *(f)* filtered area.

search and pattern-finding, which is very different from left-to-right reading. Therefore, a specific model for this scenario is required. Using two types of sensors (GSR and eye tracker), we compiled a dataset and model from 28 participants performing visual analytics tasks with a visualization interface called PivotSlice [26] (see Figure 2). The interface visualizes published research documents in a scatter plot-based design and allow users to customize the visualization by adding multiple filters and making queries. For example, the total number of authors who published a journal paper in the years between 2011 and 2014 in the InfoVis conference.

The collected data was later used to train a machine learning classifier. We tested various classifiers on the dataset, with the random forest classifier achieving the highest accuracy of 88% in differentiating between a frustration state and a normal state. The details about the user study design, data processing and the classification model can found in our previous work [13].

For building a smart recommendation system, we followed the key principles of Conati et al. [2] and Olmo et al. [3] in adaptive visualization contexts, and applied them to our model. The three key principles are: when to show help, what type of help should be shown, and how to show the help. Making these steps the foundation of this work, we explored the design space of interventions and recommendations that could potentially aid in the improvement of the users' performance.

2 Related Work

Recommendation systems have been extensively studied and are successfully implemented in many different areas; for example, in games [14], e-commerce [17], and tutoring systems [24]. Usually, these systems construct a user profile for each individual and use implicit or explicit feedback from the users to learn about their preferences.

An article by Isinkaye et al. [9] provides detail about the principles, methods and evaluation techniques for recommendation systems. In this work, the authors talk about how implicit and explicit feedback can be used to learn users' preferences and list the pros and cons of each feedback type. Also, the article explores different recommendation techniques — content-based filtering, collaborative filtering and hybrid filtering. This article provided us with a clear understanding of the types of feedback and techniques that can be used to build a recommendation system. Voigt et al. [21] have discussed some of the challenges of data scale and proposed a context-aware recommendation algorithm which leverages online annotations to provide help. Similarly, Gotz et al. [7] have proposed a system that generates recommendations in visualization, driven by user behaviour (implicit signals) and successfully reduced overall

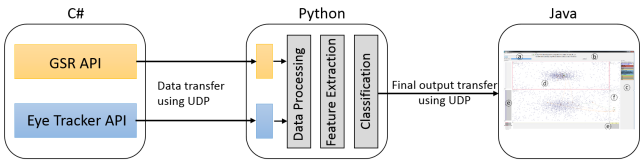

Figure 3: Internal working of the frustration detection model. Different platforms are communicating using UDP socket programming.

task completion times and errors.

Steichen et al. [18, 19] have proposed a method for adapting aspects of visualization with every individual by measuring a user's cognitive abilities and predicting performance from eye gaze data. The research questions this article tried to answer were: (1) To what extent can a user's current task, performance, long-term cognitive abilities, and visualization expertise be inferred from eye gaze data? (2) Which gaze features are the most informative? Moreover, the authors talked about how the visualization interface can change its functions and provide user-specific support. Additionally, they investigated gaze areas of interest for finding which part of the interface should be changed to support the user's cognitive abilities. Finally, the authors found that the linear regression model consistently achieved the highest accuracy but overall accuracy for each was in the range of 55% to 60%. Also, the results indicated that the contribution of features changes with every task goal but the area of interest related gaze features were most crucial.

Lastly, a study by Sun et al. [20] detects mental stress using a combination of ECG, GSR and accelerometers. Here, the authors used arithmetic problems for inducing stress in participants and then recorded the emotional response signals using a combination of biosensors. In their study, the decision tree algorithm achieved the highest accuracy. Lastly, the authors found that the GSR features were independent of tasks.

These papers informed our understanding of biosensors for detecting mental states, and the common experimental designs in this research area. Additionally, there are many research studies that use various combinations of bio-sensors [11, 12, 16, 25] for building a machine learning emotion classifier.

3 Frustration Detection Classifier

Our previous work has a detailed explanation of the feature extraction process and the classification model for detecting frustration state in analytic tasks [13].To briefly summarize, the collected raw data from the user study was processed to remove high frequency noise from the GSR and pupil diameter data using low-pass filters. We then used a standardization technique and re-scale the data between 0 and 1. Next, gaze samples with low confidence scores were removed and replaced using a linear interpolation technique, and fixations (maintaining of the visual gaze (focusing) on a single location) and saccades (gaze movement between two fixations (distance)) were calculated from the gaze location data. We calculated a total of 21 features from the dataset: 8 from the pupil size, 5 from gaze location, and 8 from the GSR data. Since the GSR and pupil size signals are linear, we calculated the features which would change with the deviation in the signal baseline. For example — mean, standard deviation and number of maxima in a processing window (here 10 seconds). From the gaze location data, we calculated features related to the fixation and saccade as any change in the mental state would affect these values. For example, users tend to fix their gaze on a certain point in case of high cognitive load (lesser number of saccade) [6]. The features related to gaze locations were — number of fixations, mean saccade length, mean fixation duration and standard deviation of fixation points from the centroid

Sequence	Weight Distribution
T T T T F F F F	1+2+3+4 = 10
T F T F T F T F	1+1+1+1 = 4
F T T T T T T T	1+2+3+4+5+6+7 = 28

Table 1: Weight distribution of sequences for estimating the intensity of detected frustration.

of all the fixations in the window (indicating the extent of the area of interest). Also, this feature set was found to be most contributing towards achieving high accuracy in the past literature [15, 20, 25]. The finalized feature set was used to test different classifiers, with the random forest classifier achieving the highest accuracy, 88%.

We optimized the model by selecting only those features which were contributing with the most impact towards the accuracy of the model. This step was necessary as the classifier needed to work on a live stream of data coming from the biosensors and needed to be fast. For that, we used a brute force technique and tested the accuracy of the model with all the arrangements and combinations of the features. After a certain number of combinations, the accuracy remained approximately constant. We picked the combination which gave the maximum accuracy for our classifier. Finally, we reduced the total number of features from 21 to 7.

In this work, we used the same model as in our previous work, but instead of reading biosensor data from a saved file, the model now uses User Datagram Protocol (UDP) socket programming to read the data directly from the biosensors (GSR and eye tracker) with low latency. We merged the data processing, feature extraction, and classification steps into a single Python program. We also leveraged UDP socket programming to send the final detected state data to the visualization tool built in Java. Figure 1 gives an overview of the internal working of the classifier. Since the classifier is working on a 10 second window with 60% overlap, it is classifying a frustration state (true or false) every 4 seconds and sending the results to the visualization tool in the form of T for the frustration state, and F for the normal state. We tested our model with different window sizes and overlaps and achieved the highest accuracy with the above mentioned settings. Lastly, the visualization tool is responsible for analyzing the received mental state and generating a recommendation. This implementation architecture allows the frustration detection and classification to be separated from the visualization tool and assistance engine.

For generating the recommendations and testing our model prior to the follow-up user study, we simulated the data received by the sensors with the existing data from 28 participants. To achieve this, the system reads participants' raw data and perform the calculation in precisely the same frequency rate as the biosensors. This simulation approach allowed us to compare the predicted results of the real-time model with the previous model. It also helped us to move forward to the intervention part without conducting another user study.

4 INTERVENTIONS

This section describes the process of generating meaningful recommendations based on the detected frustration state. Here, the visualization interface back-end is receiving outputs using the UDP socket programming and computing "when, what, and how" to intervene.

We have demonstrated some of the ideas for interventions on a specific interface (PivotSlice), but these ideas can be applied to other visualization systems. Moreover, the concepts are explained in detail but only a few were implemented, and none have been evaluated yet.

4.1 Detecting *When* to Help

Every 4 seconds, the visualization system receives the user's current state from the classifier. Saving these outputs in a list and monitoring the sequence reflects the duration of frustration which is correlated with the intensity of frustration. For example, if the system has received constant T's more than 15 times (1 minute), then the frustration intensity is higher compared to receiving less T's in 1 minute.

Since the system is classifying the frustration state every 4 seconds, the next task was to trigger the assistance function only when the system confidence is high. Taking action every 4 seconds would make the system too sensitive, would consume more computational power, and could also be tedious or annoying. Additionally, assigned confidence scores would make the system robust towards wrong classifications. Consider a case where the system detected a frustration state and displays help, based on the detected bio feedback every 4 seconds. This would divert the analyst from his task and induce more frustration which will again be detected by the system. For preventing this loop, there was a need to smooth the transition and trigger an action only when the system is confident. This can be determined by either a rule-based or machine learning technique.

4.1.1 Action Transition Smoothing

Instead of taking action every 4 seconds, we created a moving window of 32 seconds (eight classifications in a window) to smooth the action transition and build-up confidence. The moving window is overlapping 28 seconds, that is, leaving the first sample out. The rationale behind choosing a 32 second window was to provide enough data to take reasonable and meaningful action but also minimizing the risk of mixing two separate emotional phases in a single window. We did not investigate the optimum window size for this case and saved this part for the future work.

The system counts the total number of T's in the 32 second window, and if the total T's are more than or equal to a threshold value of 5 T's (20 out of 32 seconds), an action will be triggered. The threshold is an average value based on patterns observed in study participants' data. Using a higher threshold makes the system less sensitive and avoids taking actions on short-term frustration states which last for less than 20 seconds. This threshold could be tuned in further testing. For example, if the user repeatedly dismisses recommendations, the threshold may go up.

Since frustration does not come back to the neutral state right away, the system should ignore classifications produced right after a recommendation has been displayed. If the total number of T's are five or more in the window, an action will be triggered. After that, the system sleeps for the next 1 minute before monitoring the next window. This period allows users to look at the recommendation, use it, and give time for the impact to affect the emotional signals. This also prevents the system from taking repetitive actions and avoids the loop discussed above.

4.1.2 Intensity of a Frustration State

Another major component for building a smart intervention system is to add the frustration state intensity information into the model. As mentioned above, calculating the intensity of the detected frustration state can be used to infer the necessary intrusion level. Here, a different sequence of T's and F's can tell the intensity or might also differentiate between types of frustration states. For example, following are the three cases where a sequence of T and F are different in a window:

F T T T T T T T
T T T T F F F F
T F T F T F T F

In the first sequence, there is a constant occurrence of T's after the first F. This may be an indication that the user is feeling strong

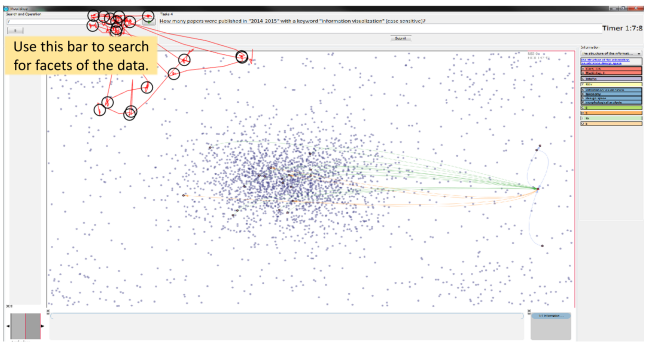

Figure 4: The system is using gaze location information to determine where the user was looking at the time of frustration and providing a mock-up recommendation based on that area. The red lines represent the gaze path, and the black circles are the fixations.

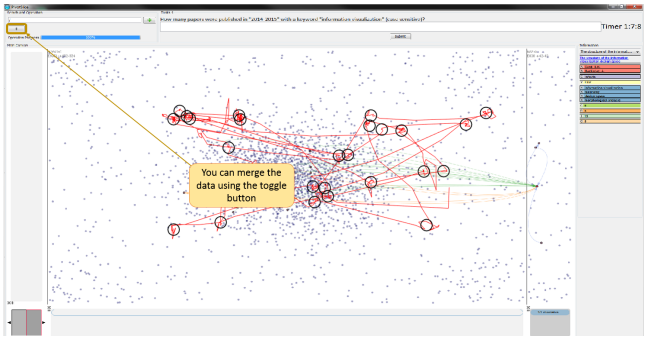

Figure 5: Dataset operation suggestion. Here, the system detected the context as dataset-related and then suggested operations that can be done on the dataset. The red lines are the gaze path and black circles represents fixations.

frustration. Next, the number of T's and F's are the same in the last two sequences, but the ordering is different. The second sequence has four constant T's which means a high-intensity frustration (short term) that faded afterwards and the third sequence means light but consistent frustration. Since these two frustration states are different, the help for these sequences should be different too. Also, the variation in these two sequences might classify the different type of negative emotions and could be explored in the future work.

To differentiate between the sequences, we have created a rule to assign a non-uniform weight to each T and calculate the total weight for each window. The rule is — any T after an F would have a weight of 1 and any immediate T after a T will have a weight one more than the previous T. Table 1 shows some sequences and their weight distributions. By using this formula, we were able to differentiate between various sequences and hence calculate the intensity of frustration for the user. This allows the system to condition the type of help based on the frustration intensity of the user.

In summary, we calculated *when* to intervene by counting the number of T's in a 32 second window. Also, by using a weight distribution formula, we have calculated the intensity of a frustration state.

4.2 Deciding *What* to Recommend

Generating a meaningful recommendation for helping users relies on knowing what is causing them frustration. Knowing the context is important in this situation because a user can exhibit the same state of

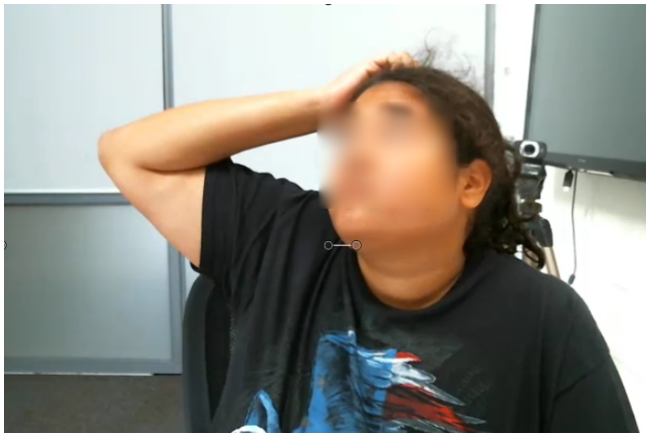

Figure 6: Participant disengaged and looking away from the task screen.

frustration for different reasons. Using the gaze signals, we classified three different contexts which are first identified by the proposed model, then a contextual help generated in response. These contexts are: *the interface, the dataset, or total disengagement*. Knowing the context would help in maximizing the chance of generating a useful recommendation. For example, if a user is having problems with the new visualization interface, then the system should not show any dataset-related help. The help should be related to the interface options or else the generated suggestion won't be able to help the user to overcome the issue causing frustration. For differentiating between each contextual state, the system uses gaze location information and calculates the centroid point of the area of interest using fixations made in a particular window. In the future, we will add more features such as the location of the most extended fixation and the total fixation time per area of the screen for more robust classification of the context.

The assumption, which admittedly requires further verification, which drives our design is that the area of recent focus is likely to be the source of any detected frustration. Observing sequences in the 32 second frustration window helps the system to decide. For example, if the system detects that a user is frustrated and out of eight times (total samples in the 32 second frustration window), five times the centroid was on the interface toolbar, then it implies that the user might be having problems using the interface functions.

Figure 4 demonstrates the system capabilities in finding the possible source of frustration with the help of gaze location. In the figure, the system is detecting the area on the interface when a frustration state was detected and displaying a mock-up recommendation. We are currently investigating the design dimensions of a recommendation, such as where to show the message box, what to display, and what color to use. The three context classes are discussed in detail below.

4.2.1 Interface

An interface-related problem occurs when a user doesn't understand the interface functions and options. This type of problem may be more likely to occur with a new interface or one with many functions on the screen. Also, in the user study we conducted, all the participants were new to the interface, and even though detailed instructions were demonstrated to them in the introductory session, they used the provided interface manual often. Since going through the manual every time is redundant, consumes time, and is a visual search task by itself, the design of recommendations for the interface is thus focused on providing contextual help on the screen to reduce

the need to consult the interface manual (one of the possible solutions is shown in Figure 4). This would prevent the users from looking away from the screen and thus may maintain task focus.

4.2.2 Dataset

Dataset-related problems occur when dealing with a new, unfamiliar or a substantial multi-dimensional dataset. Imagine an experienced analyst who is proficient with a given visualization tool, but is having trouble finding elements of interest in a new dataset. Here, showing interface-related help is not meaningful. Instead, recommendations could include showing different operations that are possible on the dataset. Figure 5 demonstrates a mock example where the system is suggesting that the dataset can be merged to get common information and also highlighting the button on the interface for merging. Other ideas are, showing interesting parts in the dataset [23], suggesting the user change the visualization view, or highlighting hidden and unseen points in the dataset. Moreover, step-by-step guidance is also possible to help the user to carry out complicated operations.

4.2.3 Disengagement

Disengagement is the case when a user is not looking at the screen. In our user study, we observed that when participants felt frustrated (later verified in retrospective think-aloud session) for an extended period, they tended to look away from the screen (Figure 6). Using voice-based dialogues can be helpful in re-orienting the user's attention back to the screen as done by D'Mello et al. [4].

While in our lab-study environment, outside distractions were reduced, but in the real world scenario, there are other factors that could make the user look away. Forcing a user to return attention to the screen could be annoying. There may be other appropriate interventions. For example, after a period of disengagement, a recommendation may replay the past few minutes of analysis, to assist with resumption of the task. There is a need to investigate this case in detail to better understand the source of disengagement. Additional sensors may be required. We did not implement this case as it would require a separate study. Hence, for this work, the system can detect disengagement by analyzing the gaze data (loss of eye tracking data for an extended period) but does not take any action.

In summary, we explored different types of contextual help that can be generated based on user's gaze information.

4.3 Deciding *How* to Recommend

Finally, after computing when to intervene and what to recommend based on different contexts, the next and the final step is to show the user help. Again, it is crucial to display the suggestion in a way that it would not distract the user from the task and would adjust the intrusion level according to the intensity of the detected frustration. Deciding the way of showing a recommendation and considering the intrusion level are the two major components discussed in this section. There is a balance to achieve between overly interruptive and prescriptive help and being too subtle and vague so that the user does not notice. Hernandez-del-Olmo et al. talk about the importance of considering the intrusion level in recommendation systems because a useful help can still be annoying if displayed in an intrusive way [8].

For better understanding of the intrusion level, consider an example using the PivotSlice interface (shown in Figure 2). If a user doesn't know how to apply two or more filters (one of the interface functions) on the interface then the system can help in three possible ways: (1) Giving a hint by displaying a short message over the search panel; (2) Break down the help into steps and guide the user by showing the help step-by-step; (3) Pause the interface and open the instruction manual. All three ways can help the user, but which one is the best is still an open question. One of the possible answers can be to show help (1) when the frustration intensity is low, and the user needs help for the first time. Help (2) is useful when the intensity is medium, and the user didn't understand help (1). Help (3) is the extreme case which may be activated upon detecting that the user is consistently getting confused with the filter functionality and help (1) and help (2) are not enough. The system needs to show different types of help depending on the intensity of frustration, as showing the same suggestion every time won't benefit the user. Since users' frustration intensity is directly affected by the interaction with the recommendation system, the system should learn from the previous interaction and decide the next recommendation based on the change in frustration intensity.

4.3.1 Degree of Intervention and Guidance

Since we have calculated the intensity of the detected frustration state using weights, we are able to calculate how much to intervene. The overall idea is that when frustration is detected, the system will analyze and compute how much to intervene based on the user's actions. For example, displaying a simple and less intrusive help window when the intensity of the frustration is low such as, highlighting options or displaying hints about the dataset in the margins. To adjust the intrusion level, we explained a simple approach — firstly, check if the user has followed the recommendation or not. The system can alter the guidance strategy based on this information. For example, if the suggestion says to use the search bar for creating a filter in the PivotSlice tool, then the backend part of the system can track if the search box was successfully used or not.

If the user has used the suggestion and the window weight (frustration intensity) decreased from the previous window, that means the help worked. If the user has used the suggestion and still the total weight of the window is increasing or constant, then the user needs more guidance, and the intrusion level can be increased. Furthermore, if the system detects over a longer term that a user is frustrated but not using the suggestion, that might be an indication that the user doesn't need that specific help. In this case, the system will display a new help and keep the degree of intervention same.

In conclusion, we designed a rule-based approach that we explored in terms of *how* a recommendation system should balance the intrusion level and what factors a system should consider facilitating a suggestion. This system is driven by data from a machine learning classifier (frustration detection classifier) which uses data from two biosensors. Replacing the recommendation rule set with a machine learning approach may make the decision making more effective, but this would require additional information and a separate model trained for a specific application and task.

5 Exploring Ways to Generate Recommendations

After discussing how to show a recommendation and what are the factors that should be considered before intervening, the next step is to generate a recommendation that would guide the users and helps them to sustain their engagement. For this, we investigated three possible scenarios, and we explain below how a recommendation can be generated based on each scenario.

When the Task is Known: This case can be applied to real-world work-flows where the task remains the same and only the variables change. For example, examining a company's financial progress using the same visualization settings and targets but with a new dataset, or a newly hired data analyst that is trying to understand relationships between different variables using past reports. In other words, when the start and end points are known, and the path between these points is unknown.

When the task is known, displaying task-related help is possible and can be valuable. In this case, the system can direct the user to the right work-flow path and help the user to solve a particular task. Chances of recommendation success are higher as the system knows the form of the final expected result, but the suggestions are not generalizable as the system is designed to solve a particular type

of task and can only generate recommendations which are related to that dataset type and task.

When the Task is Unknown: In the case where the task is unknown, the system also doesn't know how it can be solved or what to expect in the end. Therefore, the system can only show general suggestions. For example, a system could make a dataset suggestion to show outliers or to highlight data similar to the data currently in the view. These data-driven suggestions may be applicable to many tasks. In this case, the recommendations can only help users to understand the interface or dataset options and explore the possibilities, but the support may or may not direct them to the right path for solving the task. Here, the recommendations are more generalizable as they are task independent but are not direct.

When Logs are Available: Here, the system uses data from previous users to decide on a recommendation for the current user in a particular scenario. For example, if most of the previous users felt frustrated at the beginning of the task and took similar approaches to overcome that frustration then the system will analyze this trend and generate a recommendation for the next user based on the most common remedy. This case is task independent and analyzes the trends from the past logs to generate recommendations. We believe that this is the best option because it increases the chances of generating meaningful recommendations.

In conclusion, we discussed and demonstrated some ideas about how the three key steps, "when, what and how", can be answered using frustration as feedback. Also, we explored the intervention design space and went over the possible cases to generate automatic recommendations.

6 Results

In this paper we have argued that recommendation systems need to understand the user's mental state to answer "when, what and how" to intervene. We demonstrated how to classify a user's frustration state, as detected through physiological signals, and how, when, and why to use that information to generate useful recommendations in the user interface.

Moreover, we explored this concept to fill the gaps in the existing recommendation techniques with mixed-initiative interaction, specifically by detecting and analyzing a user's frustration state and the intensity of that feeling. Finally, we successfully built a working recommendation model for a visual analytics tool that uses detected frustration and its context to decide how to provide guidance. This project contributes to the better understanding of mixed-initiative interactions and machine-learning to customize visualization interfaces and opens up ample of future research opportunities.

7 Limitations and Future Work

The project is in its early stages of development and requires implementation and experimental validation of the suggested intervention techniques. The paradigm suggested here could be generalized to additional analytic interfaces and scenarios. Moreover, calculating an optimum frustration window size and classifying different negative emotions based on the weights could help the system to provide more customized and contextual assistance.

References

[1] D. Cernea, A. Ebert, and A. Kerren. A study of emotion-triggered adaptation methods for interactive visualization. In *Proc. UMAP Workshops*, 2013.

[2] C. Conati, E. Hoque, D. Toker, and B. Steichen. When to adapt: Detecting user's confusion during visualization processing. In *Proc. UMAP Workshops*, 2013.

[3] F. H. Del Olmo and E. Gaudioso. Evaluation of recommender systems: A new approach. *Expert Systems with Applications*, 35(3):790–804, 2008.

[4] S. D'Mello, A. Olney, C. Williams, and P. Hays. Gaze tutor: A gaze-reactive intelligent tutoring system. *Int. J. of Human-Computer Studies*, 70(5):377–398, 2012.

[5] J. E. Driskell and E. Salas. *Stress and Human Performance*. Psychology Press, 2013.

[6] L. Fridman, B. Reimer, B. Mehler, and W. T. Freeman. Cognitive load estimation in the wild. In *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, p. 652. ACM, 2018.

[7] D. Gotz and Z. Wen. Behavior-driven visualization recommendation. In *Proc. Int. Conf. on Intelligent User Interfaces*, pp. 315–324. ACM, 2009.

[8] F. Hernandez-del Olmo, E. Gaudioso, and J. G. Boticario. Evaluating the intrusion cost of recommending in recommender systems. In *Proc. Int. Conf. on User Modeling*, pp. 342–346. Springer, 2005.

[9] F. Isinkaye, Y. Folajimi, and B. Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015.

[10] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. In *Information Visualization*, pp. 154–175. Springer, 2008.

[11] H. Kurniawan, A. V. Maslov, and M. Pechenizkiy. Stress detection from speech and galvanic skin response signals. In *Proc. Int. Symp. on Computer-Based Medical Systems*, pp. 209–214. IEEE, 2013.

[12] Y. Liu, O. Sourina, and M. K. Nguyen. Real-time eeg-based human emotion recognition and visualization. In *Proc. Int. Conf. on Cyberworlds*, pp. 262–269. IEEE, 2010.

[13] P. Panwar and C. M. Collins. Detecting negative emotion for mixed initiative visual analytics. In *Extended Abstracts of the SIGCHI Conf. on Human Factors in Computing Systems*, pp. LBW004:1–LBW004:6. ACM, 2018.

[14] N. Peirce, O. Conlan, and V. Wade. Adaptive educational games: Providing non-invasive personalised learning experiences. In *Proc. Int. Conf. on Digital Games and Intelligent Toys Based Education*, pp. 28–35. IEEE, 2008.

[15] M. Rodrigue, J. Son, B. Giesbrecht, M. Turk, and T. Höllerer. Spatio-temporal detection of divided attention in reading applications using eeg and eye tracking. In *Proc. Int. Conf. on Intelligent User Interfaces*, pp. 121–125. ACM, 2015.

[16] N. Sharma and T. Gedeon. Objective measures, sensors and computational techniques for stress recognition and classification: A survey. *Computer Methods and Programs in Biomedicine*, 108(3):1287–1301, 2012.

[17] S. Sivapalan, A. Sadeghian, H. Rahnama, and A. M. Madni. Recommender systems in e-commerce. In *Proc. World Automation Congress*, pp. 179–184. IEEE, 2014.

[18] B. Steichen, C. Conati, and G. Carenini. Inferring visualization task properties, user performance, and user cognitive abilities from eye gaze data. *ACM Trans. on Interactive Intelligent Systems (TiiS)*, 4(2):11, 2014.

[19] B. Steichen, M. M. Wu, D. Toker, C. Conati, and G. Carenini. Te, te, hi, hi: Eye gaze sequence analysis for informing user-adaptive information visualizations. In *Int. Conf. on User Modeling, Adaptation, and Personalization*, pp. 183–194. Springer, 2014.

[20] F.-T. Sun, C. Kuo, H.-T. Cheng, S. Buthpitiya, P. Collins, and M. Griss. Activity-aware mental stress detection using physiological sensors. In *Proc. Int. Conf. on Mobile Computing, Applications, and Services*, pp. 282–301. Springer, 2010.

[21] M. Voigt, S. Pietschmann, L. Grammel, and K. Meißner. Context-aware recommendation of visualization components. In *Proc. Int. Conf. on Information, Process, and Knowledge Management (eKNOW)*, pp. 101–109. Citeseer, 2012.

[22] L. Wang, G. Wang, and C. A. Alexander. Big data and visualization: methods, challenges and technology progress. *Digital Technologies*, 1(1):33–38, 2015.

[23] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Trans. on Visualization and Computer Graphics*, (1):1–1, 2016.

[24] B. P. Woolf. *Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing E-learning*. Morgan Kaufmann, 2010.

[25] J. Zhai, A. B. Barreto, C. Chin, and C. Li. Realization of stress detection using psychophysiological signals for improvement of human-computer interactions. In *SoutheastCon*, pp. 415–420. IEEE, 2005.

[26] J. Zhao, C. Collins, F. Chevalier, and R. Balakrishnan. Interactive exploration of implicit and explicit relations in faceted datasets. *IEEE Trans. on Visualization and Computer Graphics*, 19(12):2080–2089, 2013.