A conceptual framework for an Affective Tutoring System using unobtrusive affect sensing for enhanced tutoring outcomes

Fwa Hua Leong

Thesis submitted for the
degree of Doctor of Philosophy

School of Computing Science
Newcastle University
Newcastle upon Tyne
United Kingdom

November 2018

Acknowledgements

First and foremost, I would like to thank my supervisor Dr Lindsay Marshall for his kind guidance and constant encouragement and my co-supervisor Professor Chris Philips for his valuable inputs at appropriate moments throughout the research.

I would also like to credit my wife for taking care of our family when I was away in UK during the residential period of my study. Without her patience, understanding and support, I would not have made it this far.

Special thanks also to Nanyang Polytechnic for sponsoring my PhD study and to my colleagues who helped to take over my teaching duties when I was away.

Abstract

Affect plays a pivotal role in influencing the student's motivation and learning achievements. The ability of expert human tutors to achieve enhanced learning outcomes is widely attributed to their ability to sense the affect of their tutees and to continually adapt their tutoring strategies in response to the dynamically changing affect throughout the tutoring session. In this thesis, I explore the feasibility of building an Affective Tutoring System (ATS) which senses the student's affect on a moment-to-moment basis with the use of unobtrusive sensors in the context of computer programming tutoring. The novel use of keystrokes and mouse clicks for affect sensing is proposed here as they are ubiquitous and unobtrusive. I first establish the viability of using keystrokes and contextual logs for affect sensing first on a per exercise session level and then on a more granular basis of 30 seconds. Subsequently, I move on to investigate the use of multiple sensing channels e.g. facial, keystrokes, mouse clicks, contextual logs and head postures to enhance the availability and accuracy of sensing. The results indicated that it is viable to use keystrokes for affect sensing. In addition, the combination of multiple sensor modes enhances the accuracy of affect sensing. From the results, the sensor modes that are most significant for affect sensing are the head postures and facial modes. Nevertheless, keystrokes make up for the periods of unavailability of the former. With the affect sensing (both sensing of frustration and disengagement) in place, I moved on to architect and design the ATS and conducted an experimental study and a series of focus group discussions to evaluate the ATS. The results showed that the ATS is rated positively by the participants for usability and acceptance. The ATS is also effective in enhancing the learning of the students.

Content

Chapter 1. Introduction 1

1.1. Problem Statement 2

1.2. Aim and Scope 3

1.3. Significance of Study 5

1.4. Thesis Structure 6

1.5. Publications 8

Chapter 2. Literature Review 9

2.1. Emotions 9

2.1.1. Classification of emotions 10

2.1.2. Role of emotions in learning 13

2.2. Managing Negative Emotions in Learning 16

2.3. Intelligent Tutoring Systems 17

2.4. Affect in Tutoring Systems 18

2.5. Affective Tutoring Systems 19

2.6. Affect Sensing 22

2.6.1. Facial Expressions 22

2.6.2. Voice or Audio 25

2.6.3. Body Postures 26

2.6.4. Head pose 28

2.6.5. Physiology 29

2.6.6. Input devices 31

2.6.7. Multimodal ... 32

2.7. Summary ... 34

Chapter 3. Investigating Viability of Keystrokes and the Granularity for Affect Detection ... 36

3.1. Introduction ... 36

3.2. Setup ... 37

3.3. Annotation of frustration ... 41

3.4. Modeling ... 43

3.4.1. Contextual Features ... 43

3.4.2. Keystroke Features ... 45

3.4.3. Study 1 – Logistic Regression ... 50

3.4.4. Study 2- Bayesian Network ... 51

3.4.5. Study 2 -Naïve Bayes ... 55

3.4.6. Study 2- Additional Feature Processing ... 55

3.4.7. Cross validation ... 59

3.5. Results and Discussion ... 59

3.5.1. Study 1 ... 59

3.5.2. Study 2 ... 66

3.6. Summary ... 69

Chapter 4. Multimodal Techniques for Frustration Detection ... 72

4.1. Introduction ... 72

4.2. Background ... 73

4.2.1. Facial ... 73

4.2.2. Keystrokes and mouse clicks 74

4.2.3. Head pose 74

4.2.4. Multimodal Fusion 75

4.3. Modelling 75

4.3.1. Facial Features 75

4.3.2. Keystrokes, mouse clicks and contextual 77

4.3.3. Head pose 77

4.3.4. Random Forests 80

4.3.5. Decision Fusion and Feature Fusion Techniques 81

4.4. Results and Discussion 84

4.4.1. K-fold cross validation 84

4.4.2. Student-level nested cross validation 89

4.5. Conclusion 93

Chapter 5. Automatic Inference of Engagement 96

5.1. Introduction 96

5.2. Setup 98

5.3. Modeling 98

5.3.1. Head Pose 99

5.3.2. Features 101

5.3.3. Hidden Markov Model (HMM) 101

5.3.4. HMM with continuous observations 104

5.3.5. Unsupervised Learning 106

5.4. Results and Discussion 108

5.5. Conclusion 112

Chapter 6. Architecture of Affective Programming Tutoring System 113

6.1. Introduction 113

6.2. Background 113

6.3. Overall Architectural Design 118

6.3.1. Sensor Subsystem 120

6.3.2. Student Subsystem 120

6.3.3. Tutoring Subsystem 121

6.4. Design of Hint Generation Module 126

6.4.1. Flow of hint generation 127

6.4.2. Program Canonicalization 129

6.4.3. Intention matching 131

6.4.4. Generation of meaningful and explanatory hint messages 133

6.5. Drools Rule 137

6.6. Summary 137

Chapter 7. Evaluation of system 139

7.1. Introduction 139

7.2. Experimental Study 139

7.3. Focus group discussions 140

7.4. Conduct of Evaluations 141

7.4.1. Experimental Study 141

7.4.2. Focus Group Discussions 142

7.5. Results 143

7.5.1. Experimental Study 143

7.5.2. Focus Group Discussions 144

7.6. Discussion of results 147

7.6.1. Experimental Study 147

7.6.2. Focus Group Discussions 148

7.7. Conclusion 151

Chapter 8. Conclusion 152

8.1. Introduction 152

8.2. Summary of work 152

8.3. Findings and contributions 154

8.4. Possible areas for future research 157

Bibliography 159

List of Figures

Figure 2-1: Core Affect Model (Russell, 2003) 12

Figure 2-2 The four quadrant learning spiral model (Kort et al., 2001) 15

Figure 2-3: Examples of some AUs extracted from Cohn and Kanade's database (Kanade et al., 2000) 24

Figure 3-1: Screen shot of the exercise page 40

Figure 3-2: Screen shot of custom annotation program 42

Figure 3-3: Raw keystroke file 46

Figure 3-4: Processed keystroke file 46

Figure 3-5: Bayesian Network Structure For Modeling Frustration 54

Figure 3-6: Overlapping time window slices for annotation of frustration 56

Figure 3-7: Yerkes-Dodson Law 61

Figure 3-8: Notched box plot of number of submissions for compilation 62

Figure 3-9: Notched box plot of number of wait durations from 1 to 2 seconds 63

Figure 3-10: Notched box plot for number of backspace keys 64

Figure 3-11: Notched box plot for logarithm of number of keystrokes 65

Figure 3-12: ROC curves for the various time window sizes 68

Figure 4-1: Yaw, Pitch and Roll angles of head pose 79

Figure 4-2: Decision Template fusion architecure 82

Figure 4-3: ROC curves of the various fusion techniques 86

Figure 4-4: Data pre-processing, features selection and training of features fusion classification models 90

Figure 5-1: Head position and screen setup 100

Figure 5-2: AIC versus the number of states for HMM models from leave-one-out cross-validation 108

Figure 7-1: New design of tutoring system..149

List of Tables

Table 3-1: List of contextual features 44

Table 3-2: Difficulty level of exercise 45

Table 3-3: Details of keystroke files format 47

Table 3-4: List of aggregated keystroke features used in Study 1 48

Table 3-5: List of aggregated keystrokes and mouse click features used in Study 2.. 50

Table 3-6: Mean number of keystrokes by time window width 57

Table 3-7: Performance of contextual only and combined with keystroke models 60

Table 3-8: Performance measures of Naïve Bayes and Bayesian Network models 67

Table 3-9: Performance measures for the various time window sizes 67

Table 4-1: List of Action Units (AUs) and FACS names 77

Table 4-2: Raw and processed head pose files format 78

Table 4-3: List of head pose features 80

Table 4-4: AUC measures of unimodal and multimodal models 85

Table 4-5: AUCs of unimodal models (Random Forests, Logistic Regression and Naive Bayes) 91

Table 4-6: AUCs for feature fusion models across the classifiers 92

Table 4-7: AUCs for 2-channels and 3-channels feature fusion models 92

Table 5-1: Learned Gaussian Mixture Weights 109

Table 5-2: Mean values of features for Gaussian Mixture 1 109

Table 5-3: Mean values of features for Gaussian Mixture 2 109

Table 5-4: Combined means values of features from GM1 and GM2 109

Table 5-5: Probabilities of state transitions 110

Table 5-6: Number of exercises completed and disengagement duration by students 111

Table 6-1: List of Semantic Preserving Variations.. 129

Table 6-2: Levenshtein edit distance matrix of student code and model code 132

Table 6-3: List of top novice programmers’ errors 135

Table 7-1: Descriptive Statistics for determinants of learning achievement......... 143

Chapter 1. Introduction

We live in an age where the global economy is undergoing unprecedented changes. According to the “Future of Jobs” report by World Economic Forum ("The Future of Jobs," 2017), 7.1 million jobs will be displaced with white collar jobs comprising two thirds of it. Technological disruptions such as machine learning, Artificial Intelligence and Internet Of Things (IOT) already have and will continue to disrupt both businesses and jobs. Businesses are already feeling the impact of these technological driven changes with the advent of huge disruptions in their traditional business models. Uber, the world’s largest taxi company owns no taxis and AirBnB, the world’s largest accommodation provider owns no real estate. These new businesses are disrupting the incumbents and bringing about a massive wave of transformation in their respective industries. This transformed business model will in turn necessitate a different set of skills from their employees, shortening the shelf life of one’s knowledge.

The silver lining is that there will be new jobs created which will require a palette of skill sets which does not exist today. It is projected that there will be 2 million new jobs created in the Computer, Mathematical, Engineering and Architecture related fields. As of now, countries such as Spain are experiencing high unemployment and this is attributed to a skills mismatch in the Info-communications and Technology (ICT) sector resulting in structural unemployment. Countries worldwide are realizing that digital skills underpin growth across many sectors in their economy and are incorporating it into both their pre-employment curricular and post-employment training. France has given its elementary school students an option to learn computer programming while Estonia has started to teach their first graders how to code their own computer games. England has updated its national curriculum where primary education’s ICT has changed to computing curriculum and would include programming and internet safety (WHSmith, 2017). Singapore is also planning to introduce software programming in the government-funded schools, advocating that computer programming skills will be the strategic catalyst for their nation’s competitive advantage in the future (techinasia, 2014). Nations globally are and will continue to make huge adjustments to their curricular with the hope of riding on this new tsunami wave of technological driven disruptions.

1.1. Problem Statement

This impetus to introduce computer programming to the masses requires a change in the way we deliver the course. Intelligent Tutoring System (ITSs) which came into existence in the 1970s fulfils the role of providing students with the benefits of one to one tutoring automatically and yet can easily be scaled to large masses of students, resulting in high cost effectiveness. It acts like a personal training assistant that continually assesses one's knowledge through interactions with the system and builds a personalized model of one's acquired knowledge for the provision of tailored instructions or assistance in the form of hints or demonstrations when one seems to require help to move on. However, a common criticism of ITSs is that it underperforms one to one tutoring with a human tutor as it lacks emotional awareness. Human tutors understand the central role of emotion in learning and can adapt their tutoring based on both the cognitive and affective response of their students (Ambady & Rosenthal, 1992).

As a lecturer myself, I have witnessed countless episodes where students give up on an assignment or laboratory exercise due to frustration or boredom. Correspondingly, there are also instances where the students' learning is promoted by their positive emotions. It seems that emotion is the switch that toggles their attention which in turn drives their learning. This critical role that emotion plays in learning is affirmed in a number of studies as well (Kort, Reilly, & Picard, 2001; Pekrun, Goetz, Titz, & Perry, 2002). This affirms the need to augment ITSs with emotion sensing capability to enhance their tutoring effectiveness.

Affective Tutoring Systems (ATSs) are ITSs that adapt their tutoring response based on the sensed emotions of the students, resulting in enhanced tutoring outcomes. The first step to building an ATS is to equip it with the ability to sense the emotions of the tutee. There are various affect sensing techniques and among the many, sentic modulation (Rosalind W

Picard, 1997) is the technique which shows much promise in dynamic affect sensing. Sentic modulation refers to the physical assessment of a person's emotional changes via sensors such as cameras, microphones and wearable devices that register subtle physical modulation produced by emotional states. We humans express our affect in multiple channels e.g. facial expressions, body postures and vocal intonations which thus leads to the belief that the use of multiple modalities for affect detection would more closely emulate human affect expression. However, the majority of the studies in the literature used a single sensor (unimodal) for sensing. This has been attributed to the complexities involved in the use of multiple sensors (multimodal).

The cost, obtrusiveness, availability and ease of use of some of these sensors are also areas of concern. For example, Electroencephalography (EEG) sensors not only require trained expertise to set up but also require electrodes to be placed on a subject's scalp which would be obtrusive to many subjects. In addition, the EEG sensors require constant contact with the scalp, failing which no readings can be obtained from the sensor. These are issues that are too onerous for a teacher to deal with (on top of the time pressures to deliver the curriculum) in a classroom environment, thus constraining its deployment in educational settings.

Lastly, although researchers acknowledged the role that emotions play in learning, ATSs are rarely implemented which may be attributed to the fact that the construction of ATSs requires cross-disciplinary knowledge encompassing education, psychology and computing science.

1.2. Aim and Scope

The aim of this research is to explore the feasibility of building an ATS using unobtrusive sensors for the learning of computing programming. This aim can be broken down into the following sub aims.

1. To develop a method for sensing the emotions of students that are detrimental to learning on a moment-to-moment basis from unobtrusive and cost effective sensors.

   Specifically, I investigate into the viability of keystrokes and contextual logs for affect sensing and the granularity in which affect can be sensed using keystrokes and contextual logs without compromising much on the level of accuracy.
2. To investigate the use of multimodal affect sensing techniques for enhanced affect detection.

   Multimodal affect sensing is hypothesized to be superior to unimodal affect sensing as it is commonly believed that the multiple sensors complements one another. Thus, this leads to my investigation into whether a multimodal array of keystrokes, mouse clicks, facial features, head pose and contextual logs enhances the accuracy of affect sensing as compared to unimodal means.
3. To develop a method for formulating and adapting the tutoring response in accordance with the sensed emotions of the students for optimizing their learning.

   Adapting the tutoring measures in response to the sensed affect of the students enhances engagement, task persistence and sustains their motivation to learn. Thus, it is critical to design a tutoring response module that is both sensitive to the students' affective states and their knowledge gaps.
4. To design an architecture and implement an ATS for the learning of computer programming.

   I propose the architecture for an ATS that though, currently only tutors in computer programming, is designed with the flexibility of extending for use in other tutoring domains.

This research is restricted to only investigating the sensing of frustration and disengagement of the students. The rationale is that negative emotions erodes motivation and draws attention away from the task, resulting in shallower learning. If we can alleviate negative emotions such as frustration or disengagement, students' engagement in learning can be sustained and they can go on to acquire higher level skills latent in their Zone of Proximal Development (Vygotsky, 1978), underscoring the importance of detecting and managing negative emotions in learning.

1.3. Significance of Study

The list of contributions from this research is summarized below.

1. Use of unobtrusive sensors such as keystrokes and mouse clicks for affect sensing.
2. Use of multimodal affect detection (fusion of multiple sensors) which more closely emulates humans’ expression and recognition of emotions.
3. Implementation of an ATS which tutors students in the domain of computer programming.

Sentic modulation is a promising technique in affect sensing. However, the cost, obtrusiveness, availability in typical classroom environment and the ease of setting up the sensors for regular use are factors that will impact the effectiveness and deployment of these sensors in actual academic settings. In this research, I propose the use of keyboard, mouse and web cameras to capture the keystroke and mouse click patterns as well as the facial expressions and head postures of students for sensing of their affect. Keyboards and mouse are standard input devices attached to every computer in a computer lab. Web cameras though not as common, are relatively cheap and can easily be setup for use. Conceivably, these would qualify as economical commodity devices that are unobtrusive and thus apt for capturing the behavioral attributes of users inconspicuously. In addition, keystrokes and mouse clicks have traditionally been researched for the purposes of authentication in the security domain. This underscores the novelty of this research in the use of keystrokes and mouse clicks for affect sensing.

Another area of significance is in multimodal affect detection – the fusing of multiple sensing channels’ outputs. Human emotion is expressed in various channels e.g. facial, vocal and bodily expressions but implementation of a multimodal affect detection system is still rare in occurrence (Jaimes & Sebe, 2007). Thus, I propose a multimodal fusion of keystrokes, mouse clicks, contextual logs, facial and head postures for affect detection in this research, investigating into various multimodal fusion techniques as well as whether multimodal fusion leads to higher detection accuracy, justifying the added complexity.

The last area of significance relates to the implementation of an ATS which tutors students in the domain of computer programming. Despite the recognized benefits of incorporating affect into tutoring systems, ATSs are seldom implemented (Thompson & McGill, 2012), let alone one that tutors in computer programming. For an ATS, the next task after the sensing of the students' affect would be to adapt and tailor the tutoring strategy to the recognized affect to sustain the students' learning. However, this is an area where research is much lacking. In this research, I propose a pedagogical based model of responding to the students expressing negative affect centered on a context sensitive and comprehensive hint system. The effectiveness of the ATS will also be evaluated through both quantitative and qualitative techniques specifically an experimental study and a series of focus group discussions.

1.4. Thesis Structure

This thesis is structured as follows.

Chapter 2: I review the relevant literature on emotional theories, how emotion is measured or categorized and its role in learning. I then move on to a review of Affective Tutoring Systems and the various sensors that were used for affect sensing in literature before discussing the merits and demerits of each which in turn lay the stage for why I select the proposed sensing modes.

Chapter 3: I present the methodology and results for using keystrokes and contextual logs from the tutoring system for the sensing of frustration both on a coarse grained and fine grained basis. Data collection and processing techniques are also detailed here.

Chapter 4: I present the methodology and results for combining multiple sensor outputs (multimodal) in the sensing of frustration. The detailed results of using both unimodal and the various multimodal techniques for affect sensing are also presented and discussed here.

Chapter 5: I present the methodology and results for the modeling of engagement of students through an unsupervised machine learning technique.

Chapter 6: I present the architectural design of the Affective Tutoring System as well as the methodology for the generation of programming hints.

Chapter 7: I present the evaluation of APTS through analysis and discussion of the results of both an experimental study and a series of focus group discussions conducted with the student participants.

Chapter 8: I close the thesis with a conclusion of the work presented in this thesis and suggestions on the opportunities for future research.

1.5. Publications

Below is a list of publications that resulted from this research.

1. "Automatic detection of frustration of novice programmers through contextual and keystroke logs" presented in 10th International Conference on Computer Science and Education (ICCSE 2015) authored by Fwa Hua Leong.
2. "Fine-grained detection of programming students' frustration using keystrokes, mouse clicks and interaction logs" presented in International Conference on Educational Technology Research (ICETR 2016) authored by Fwa Hua Leong.
3. "Modeling engagement of programming students using unsupervised machine learning technique" presented in Computer Science Education: Innovation and Technology (CSEIT 2017) authored by Fwa Hua Leong.
4. "An architectural design and evaluation of an Affective Tutoring System for novice programmers" to be published in International Journal of Educational Technology in Higher Education (provisional acceptance in June 2018) authored by Fwa Hua Leong.

Chapter 2. Literature Review

2.1. Emotions

Emotions permeate most aspects of our life but what exactly are emotions? Lange and James (1922) proposed the James-Lange theory of emotion which defined emotions as "the bodily changes follow directly the PERCEPTION of the exciting fact, and that of our feeling of the same changes as it occurs" (James, 1884). James reasoned that the physiological response occurs first, then the perception of emotion and followed lastly by the reaction. He gave the example of a person seeing a bear would first tremble, becomes afraid and then run. Cannon (1927) and Bard countered that this is counter-intuitive and formulated the Cannon-Bard theory instead. Cannon, on the contrary, argued that firstly physiological changes can possibly occur without experiencing emotions. In some cases, physiological changes may lag the experience of emotions and that the same physiological changes can occur for different emotions. Cannon and Bard theorized that physiological changes occur together with the experience of emotions instead. Scherer (1984) further extended this with the Component Process Model which states that emotion is the result of interrelated and synchronized changes in the state of five components (namely cognitive appraisal, bodily symptoms, action tendencies, expressions and feelings) in response to an external or internal stimulus. Lazarus (1991) proposed that the cognitive appraisal must happen first before the experience of emotion and this is coined as the Lazarus Theory of Emotion. According to this theory, the sequence of events first involves a stimulus, followed by the thought, which then leads to the simultaneous experience of a physiological response and the emotion. From these emotional theories, one can conclude that bodily changes would occur with the experience of emotions and that emotions and cognition are indeed connected.

For many years, the study of emotions has been relegated below that of cognition. The state of neglect on emotions is well summed up by A. Damasio (1999)-

“Throughout most of the twentieth century, emotion was not trusted in the laboratory. Emotion was too subjective, it was said. Emotion was too elusive and vague. Emotion was at the opposite end from reason … and entirely independent from emotion. This was a perverse twist on the Romantic view of humanity. Romantics placed emotion in the body and reason in the brain. Twentieth–century science left out the body, moved emotion back into the brain, but relegated it to the lower neural strata associated with ancestors who no one worshiped. In the end, not only was emotion not rational, even studying it was probably not rational.”

So one may ask, why the need to study emotions then? What if we lose the ability to feel emotions? An unfortunate case study that offers a partial answer to this is that of Elliot (A. R. Damasio, 1994), a man who has undergone immense personality change after an operation to remove a brain tumor on the surface of his frontal lobes in his thirties. His intelligence, kinesthetic and verbal ability were not affected by the surgery but he lost the ability to feel emotion and that flawed his ability to make decision. It eventually led to his bankruptcies and two divorces. Thus, as illustrated in Elliot’s case, emotion deficiency will impair our decision making ability and in turn leading to catastrophic consequences in our daily life.

Other than our decision-making ability, our emotions have also been found to influence our other cognitive abilities. For example, some researchers have found that anxiety, a negative emotion, correlates negatively to working memory capacity (Ashcraft & Kirk, 2001; Beilock & Carr, 2005). Positive emotions have also been linked to a broadening of the scope of thinking leading to more creative problem solving (Isen, Daubman, & Nowicki, 1987).

2.1.1. Classification of emotions

The classification of emotions can be summarized into 3 main approaches – Discrete, Dimensional and Appraisal.

- Discrete

This approach states that all humans have an innate set of basic emotions that are cross-culturally recognizable. These basic emotions are termed 'discrete' because they can be discriminated by one's facial expression and other physiological changes. One of the most influencing studies conducted on the basic emotions that are universally recognizable is that conducted by Paul Ekman et al. (1987). He concluded that there is a set of six basic emotions that are distinguishable by facial expressions cross-culturally. These six basic emotions are happiness, sadness, surprise, fear, disgust and anger. Happiness for instance, can be symbolized by the raising of the mouth corners and tightening of the eyelids. Although most researchers agree on this set of 6 universally recognized set of basic emotion, some argued that not all emotions are representable within the six basic ones. Some of these other emotions include interest, boredom, shame, pride, disgust, and contempt (Rosalind Wright Picard, 1995; Scherer & Ellgring, 2007).

- Dimensional

In the dimensional approach, emotions are categorized into two or three dimensions. Most dimensional models use the dimensions of valence and arousal. Negative emotions such as sadness and anger would have a negative valence while positive emotions such as happiness would have a positive valence. Arousal refers to the intensity of the emotion. For example, anger would rate higher on the arousal scale than boredom. Russell and Pratt (1980) proposed the Circumplex Model of Emotion which suggests that emotions are distributed in a two-dimensional circular space with the dimensions arousal and valence on the vertical axis and the horizontal axis respectively. He further developed this into the Core Affect model which is illustrated in Figure 2-1. The Core Affect model is a 2 dimensional scale with Pleasure-Displeasure on the horizontal axis and Activation-Deactivation on the vertical axis. The horizontal axis starts in a continuum from the positive emotions e.g. happy on the right to the negative emotions e.g. sad on the left while the vertical axis ranges from the least at the bottom to the most aroused on top.

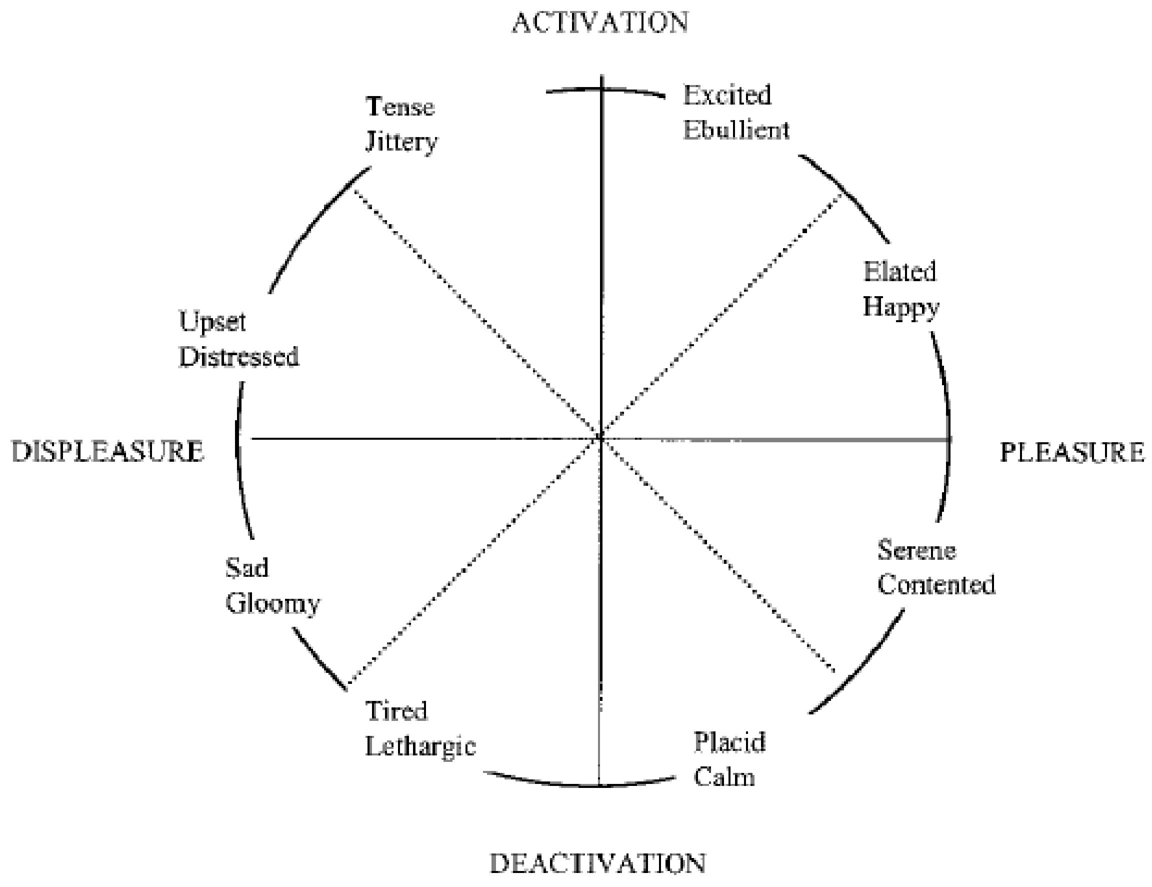


Figure 2-1: Core Affect Model (Russell, 2003)

As compared to the discrete approach, the mapping of emotions into two or more dimensional scales results in a higher spectrum of emotions that can be represented. However, some researchers have questioned that arousal and valence are not fully independent. There is also a loss of information as emotions such as fear and anger cannot be distinguished on a dimensional scale (Zeng, Pantic, Roisman, & Huang, 2007b). The reason being that both fear and anger rate high in arousal and negative in valence. Lastly, complex and subtle states such as pride or shame cannot be represented easily on a dimensional scale (Barthet, Fazekas, & Sandler, 2012).

- Appraisal

The appraisal theory of emotions states that emotions are extracted from our evaluations or appraisals of events. It is our appraisal of the events that determines our emotional response and that also accounts for the varied responses amongst different individuals to the same event. Lazarus (1991) further broke down the appraisal process into two – primary appraisal and secondary appraisal.

In primary appraisal, individuals assessed the occurrence of a situation with respect to their goals in terms of relevance, congruence and content. Relevance deals with the whether anything is at stake, congruence is on whether it is beneficial or harmful and content deals with the kind of goal at stake. In secondary appraisal, individuals evaluate the resources they have at hand to handle the situation, the accountability and the options for coping. Accountability refers to the direction of blame or credit. One can either blame or credit oneself or others for the occurrence of a situation. Lastly, coping potential depends on the amount of influence one has over the environment to cope with the situation.

One benefit of appraisal theory is that it allows us to represent complex emotions such as guilt, pride or shame which cannot be represented on a dimensional model. However, appraisal theory has its share of criticism as well. The key assumption of appraisal theory - appraisal precedes emotion may not be always true. It is probable that sometimes, appraisal is a consequence of an emotional reaction rather than a cause (Zajonc, 1980). A stimulus would, according to Zajonc, first be processed in terms of the affective properties, after which the cognitive process comes into play. In his experiment, Öhman and Soares (1994) demonstrated that processes below the level of conscious awareness can produce emotional reactions as well. In their experiment, snakes or spiders phobic subject experienced fear even before they were able to identify the snakes or spiders stimulus. This adds further support to the argument that appraisal may not precede emotion.

2.1.2. Role of emotions in learning

Pekrun et al. (2002) opined that academic emotions (which are linked to academic learning, classroom instruction and achievements) with the exception of test anxiety are largely neglected by educational psychology research. They embarked on a series of qualitative and quantitative studies to uncover the academic emotions experienced by students and the effect of these emotions on the students' learning and achievements. Their findings revealed that students experience a wide repertoire of emotions in academic settings and positive emotions e.g. hope, pride and relief occur as frequently as negative emotions such as anxiety and boredom. The positive emotions e.g. enjoyment of learning affect achievement positively by strengthening motivation and enhancing flexible learning whereas the negative emotions e.g. anxiety erodes motivation and draws attention away from the task, resulting in shallow learning. The effects that positive emotions have on learning were also corroborated by other researchers (Greene & Noice, 1988; Isen, 2000).

Bloom (1984) highlighted the 2 sigma problem which states that students who are tutored on a one to one basis achieve learning gains of 2 sigma over and above those students who are group tutored. He attributed this difference to the process of constant corrective feedback and high engagement of the tutee which is maintained through close monitoring and constant encouragement by the tutor. Thus, to achieve enhanced learning outcome, it is crucial that the active engagement and motivation of the student be sustained through this continuous affective feedback loop between the tutor and the student.

Kort et al. (2001) postulated that learning occurs in the presence of affective states in his proposed four quadrants learning spiral model in which emotions change as a learner moves through the different quadrants. The four quadrant learning spiral model with the vertical axis denoting learning and the horizontal axis denoting affect is illustrated below in Figure 2-2

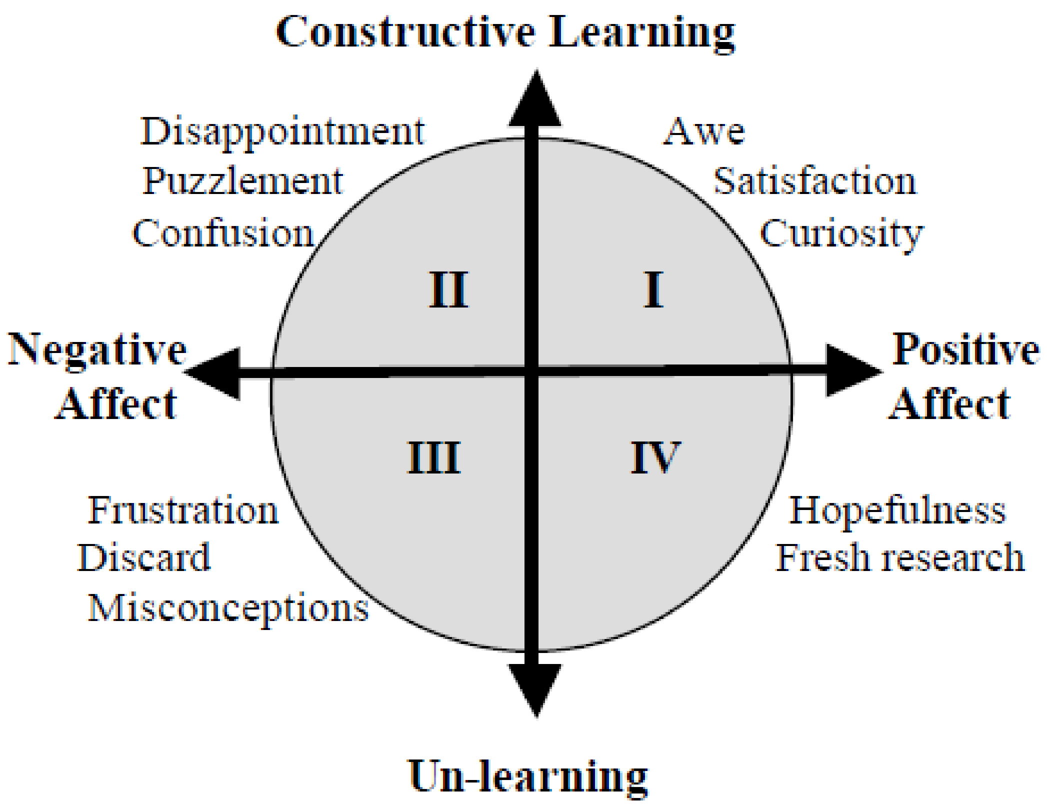


Figure 2-2 The four quadrant learning spiral model (Kort et al., 2001)

In this model, Kort hypothesized that students would typically start in quadrant I where they are experiencing positive affect and constructing knowledge as exemplified by one who is curious and eager to learn new concepts. Once discrepancies start to set in between the students' knowledge schema and the new information, they move to quadrant II where they experience confusion. When students start to discard the misconceptions, they move to quadrant III. Quadrant III is characterized by unlearning and negative affect such as frustration. After the misconceptions are discarded, students move into quadrant IV, marked by unlearning and positive affect. While in this quadrant, students would still be unsure of how to move forward but they would have developed new insights and this would propel them back into quadrant I, concluding one round through the learning spiral. As students move up the spiral in cycles, they become more competent and acquire more knowledge.

2.2. Managing Negative Emotions in Learning

Meyer and Land (2003) proposed that in every subject matter, there is a subset of core concepts which they termed threshold concepts. The mastery of threshold concepts is pertinent in order for one to be regarded as being proficient in the relevant discipline or community of practice. Threshold concepts encompass the 5 characteristics of transformative, irreversible, integrative, bounded and troublesome. They are transformative because it requires a conceptual shift that could change the way one looks at the subject; irreversible because once mastered, they are integrated into your schema and will not be easily forgotten. Integrative is defined as being able to make connections between the different concepts which may not be obvious to one before and bounded because they are marked with a boundary of conceptual knowledge. Lastly, they are troublesome as they are usually difficult concepts to be comprehended and could be counter-intuitive at times.

Meyer and Land also proposed liminality as a space, marked by confusion and frustration where students would struggle to reconcile their old and new knowledge. Upon transcending this state, students would have grasped the threshold concept but some may be stuck in this state for a long time and for others, it may be for eternity. It is plausible that if we can manage the emotions of students when they are in liminality, possibly with the provision of a supportive environment (Land, Cousin, Meyer, & Davies, 2006), we would be able to help our students negotiate the barriers of liminality. Although researchers have yet to agree on the threshold concepts in Computer Science, there is evidence presented for the presence of liminality in Computer Science learning as well (Eckerdal et al., 2007).

In a separate study by Efklides (2006) on metacognition and affect, she elaborated on student perceived task difficulty as a factor influencing metacognition in learning. She reasoned that if a very difficult task is assigned to students and they perceive it as beyond their capability, they would develop negative emotions such as frustration and give up on the task. However, if teachers are able to scaffold the students' learning or employ the use of alternative pedagogical intervention techniques e.g. provision of hints to alleviate the negative emotions, they would be able to sustain the engagement of the students.

This is akin to the concept of Zone of Proximal Development (ZPD) propounded in Vygotsky's Social Development Theory as well. Vygotsky (1978) sees the ZPD as the area of knowledge where learners would not have independently acquired on their own effort alone. This area of knowledge would have been deemed to be beyond their capability if learners are left independently on their own. The learners would likely develop negative emotions if they are left alone without help and over a prolonged period, the learners would likely give up learning totally. However, with guidance and encouragement from More Knowledgeable Others such as peers or teachers, they would instead be able to transcend their perceived capability to acquire the knowledge and even go on further to apply it on other similar contexts.

Although the above studies focus on different aspects of learning, they support the notion that if we can alleviate negative emotions such as frustration or disengagement, students' engagement in learning can be sustained and they can go on to acquire higher level skills latent in their ZPD, underscoring the importance of detecting and managing negative emotions in learning.

2.3. Intelligent Tutoring Systems

The term Intelligent Tutoring Systems (ITSs) came into existence during the early 1970s. ITSs are built with the objective of providing learners with the benefit of one to one tutoring automatically and cost effectively. It acts like a personal training assistant that continually assesses one's knowledge through interactions with the system and builds a personalized model of one's acquired knowledge for the provision of tailored instructions or assistance in the form of hints or demonstrations when one seems to require help to move on.

Some studies have in fact shown that ITSs outperform traditional classroom instruction in some domains (John R Anderson, Corbett, Koedinger, & Pelletier, 1995). However, for most domains, ITSs still underperform one-to-one tutoring. Some have argued that the reason for that is ITSs are impoverished in emotional awareness and empathy. Human tutors understand the central role of emotion in learning and can adapt their tutoring based on both the cognitive and affective response of their students. Ambady and Rosenthal (1992) demonstrated that with as little as six seconds of a teacher's first interactions with a student, participants could predict that teacher's effectiveness and student's end of term grades based on the teacher's exhibited use of affect[1]. This has in turn spawned off many studies that investigate into augmenting ITS with both emotional awareness and the ability to express them with the hope of enhancing their tutoring effectiveness (Conati & Maclaren, 2009; A. C. Graesser, Chipman, Haynes, & Olney, 2005; Kapoor, Burleson, & Picard, 2007; Mota & Picard, 2003; Rodrigo & Baker, 2009).

2.4. Affect in Tutoring Systems

The term "Affective Computing" was introduced by Rosalind W Picard (1997). As defined by Picard, it refers to "computing that relates to, arises from, or deliberately influences emotions". She further divides affective computing into four main areas.

- computer that recognize emotions
- computer that express emotions
- computers that have emotions
- computers that have emotional intelligence

The first step to building an affective tutoring system is to recognize the emotions that students are experiencing. Without first knowing the students' emotions, there is no way to adapt and respond to it. A number of researches have been done to acquire the affective states of students. Surveys and questionnaires offer one of the easiest and most direct methods. One study by De Vicente and Pain (2002) used questionnaires to predict students' motivational state. However, questionnaires are critiqued by many to be static and thus, inadequate for use in modeling the dynamic and fast changing nature of emotions.

Sentic modulation which refers to the physical assessment of a person's emotional changes (via sensors such as cameras, microphones, wearable devices that register subtle

[1] The term emotion and affect are used interchangeably

physical modulation produced by emotional states) offers much promise in dynamic affect sensing (Rosalind W Picard, 1997). Most researchers tapped on either a single sensor or a combination of sensors for affect inference. The majority of studies have relied on the use of a single sensor (unimodal), owing to the complexities involved in affect sensing which multiplies for multimodal affect sensing. However, multimodal affect sensing studies are emerging with advancement in some of the related technologies. The literature on unimodal and multimodal affect sensing will be elaborated in depth in the subsequent sections.

The expression of emotions by the computer will not be discussed in this thesis as the focus is on affect sensing and the associated pedagogical response.

2.5. Affective Tutoring Systems

Affective Tutoring Systems (ATSs) are tutoring systems that adapt to the emotions of students to bring about enhanced learning outcomes analogous to the way human tutors do. Some of the prominent Affective Tutoring Systems in the literature will be discussed in the paragraphs below.

AutoTutor is a fully automated computer tutor that helps students learn Newtonian physics and computer literacy topics. It presents problems or questions to the students and engages them in a dialogue to collaboratively build towards a solution (D’Mello et al., 2008). D’Mello *et al.* aim to incorporate learners’ affect into existing AutoTutor’s pedagogical strategies (e.g. by regulating their negative emotions) and their research efforts culminated in the development of Affective AutoTutor. Affective AutoTutor is one of the few ATSs that detects and responds to students’ affective states (S. K. D'Mello & Graesser, 2010). Affective AutoTutor uses facial cues, body postures and conversational dialogue feature to infer the affect of students. The tutoring actions are then derived from a set of production rules that dynamically assessed the cognitive and affective states of students to address negative emotions such as frustration and boredom.

Easy with Eve is an ATS that is built by Massey University, New Zealand (S. Alexander, Sarrafzadeh, & Hill, 2006). It infers the emotions of learners mainly through facial expressions analysis that was developed in-house. In addition, a case based reasoning program has been written to output a weighted set of tutoring actions and facial expressions based on a given sequence of interactions. The recommended facial expression would be expressed through Eve – an animated intelligent agent embodied within the system.

Empathic Companion is an embodied agent type system that detects and responds to the user's affective state (Prendinger, Dohi, Wang, Mayer, & Ishizuka, 2004). It was developed in the context of a web-based job interview scenario with the objective of regulating the user's negative emotions when faced with difficult job interview questions. It employs the use of a decision network which is used to fuse the inputs from a Galvanic Skin Response (GSR) and an Electromyography (EMG) sensor and translates the sensors' signals into both the emotions and the agent decisions. The results indicated that Empathic Companion does reduce the frustration of its users.

MetaTutor is an adaptive ITS that is designed to encourage students to employ meta-cognitive self-regulated learning (SRL) in the tutoring context of human circulatory system (Azevedo, Johnson, Chauncey, & Burkett, 2010). MetaTutor's focus is on SRL skills that are fostered by managing learning through monitoring and strategy use. This is achieved in MetaTutor through the use of pedagogical agents which, for example responds with an evaluation of the student's current level of understanding upon request. Jaques, Conati, Harley, and Azevedo (2014) initiated a study with the objective of augmenting MetaTutor with affect sensing capability. In the study, she derived fixation time and duration related features from only eye gaze data to predict the occurrence of boredom and curiosity. The emotion labels are obtained through Emotion Values self-reports collected from the participants every 14 minutes during an hour of learning session with the participants. The results demonstrated an accuracy of 69% for boredom and 73% for curiosity with a 14 minute window width.

Crystal Island is a 3-dimensional narrative-centered learning environment for the tutoring of eighth-grade microbiology (Rowe et al., 2009). Sabourin, Rowe, Mott, and Lester (2013) fed the survey scores and in-game progress of students within Crystal Island into a Dynamic Bayesian Model (DBN) for the identification of students' off task behavior. The students' actions within the environment were logged. The logs were then analyzed to extract the behaviors of students that diverge from the learning task. These serve as the off-task labels for the classification task. The results showed that the DBN was able to capture the emotion label and valence with an accuracy of 34.7% and 80.6% respectively. The authors acknowledged that the accuracy for the emotion label was too low and only the predicted valence (with an accuracy of 80.6%) can be used in a tutorial setting. The objective of this study is to identify whether students use off-task behaviors to regulate their emotions e.g. to alleviate frustration.

Genetics with Jean is an ATS that teaches the subject of genetics (Thompson & McGill, 2017). It infers the learners' affective states through skin conductance and heart rate sensors. The outputs of these physiological sensors are translated into the arousal and valence dimensional values which are then used to guide the actions of a pedagogical agent which emulates a human tutor. Three distinct behaviors are scripted into the pedagogical agent to provide affective support to the learners – support affect (to alleviate negative emotions), motivate (to provide encouragement to the learner) and offer revision (to redirect the learner to relevant revision notes if performance is poor). An evaluation of the tutoring system concluded that students who received affective support from the system have higher levels of perceived learning but not higher levels of perceived enjoyment.

Affective Tutoring System for Built Environment Management (ATEN) is an ATS that tutors in the subject area of built environment (Kaklauskas et al., 2015). It senses the heart rate, systolic and diastolic blood pressures, skin humidity, perspiration, temperature and conductance (through biometric sensors) and acquires textual (module keywords) data to infer the learning style, stress and interest level of the students in learning which are then used to recommend personalized learning materials to the students. Other than a case study to demonstrate the capability of the system, no evaluation of the effectiveness of the tutoring system was done.

As stated in the previous section, researchers acknowledge the role that affect plays in learning. However, despite the recognized benefits of incorporating affect into tutoring systems, Affective Tutoring Systems are rarely implemented (Thompson & McGill, 2012). A reason may be that ATSs require cross-disciplinary knowledge encompassing education, psychology and computing science. The next step after the inference of learner’s affect would be how to adapt the tutoring strategy in response to the recognized affect to maximize the learning. However, this is a rare occurrence as evidenced by the fact that only some of the ATSs mentioned above were further developed with the tutoring responses to adapt to the inferred affect of the students. In addition, the current tutoring responses of the above mentioned ATSs are varied and ad-hoc, making a comparison among them difficult. More research will thus be required in this area.

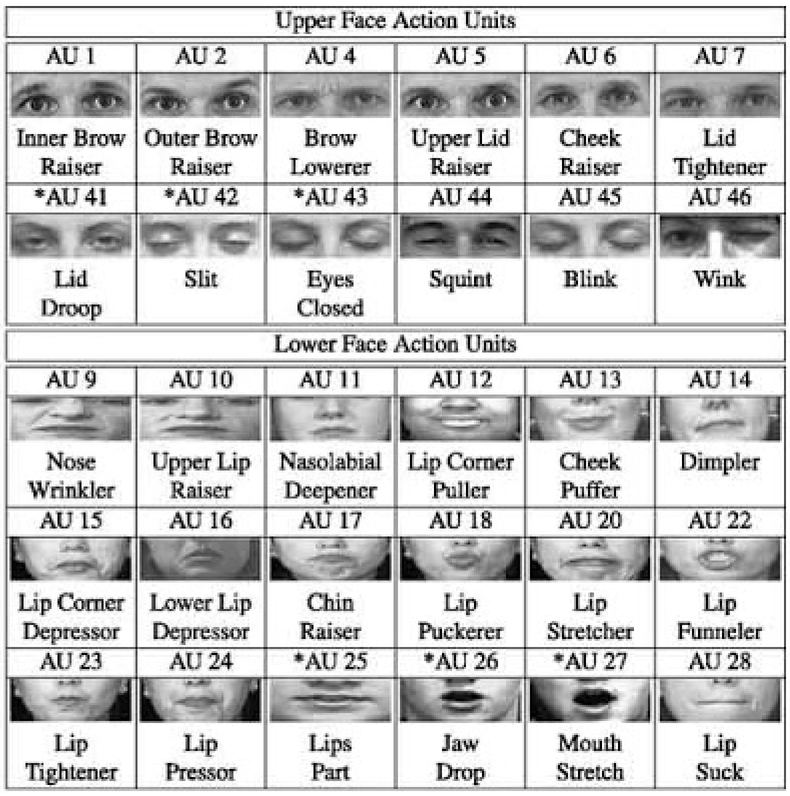
2.6. Affect Sensing

Emotions can be manifested in a wide range of affective channels ranging from facial expressions (Kapoor et al., 2007; Kapoor & Picard, 2005; Teeters, El Kaliouby, & Picard, 2006; Yeasin, Bullot, & Sharma, 2006), body postures (Coulson, 2004; Mota & Picard, 2003), voice (audio) (Litman & Forbes-Riley, 2004; Schuller, Villar, Rigoll, & Lang, 2005), physiological (AlZoubi, Calvo, & Stevens, 2009), text (Shaikh, Prendinger, & Ishizuka, 2008) and input devices (Bixler & D'Mello, 2013) e.t.c. Researchers have tapped on these various modalities or a combination of these modalities for affect sensing. These different modalities would be elaborated in the sections below.

2.6.1. Facial Expressions

One of the most common and studied channel used for the inference of affect is that of facial expressions. To date, the facial channel is still one of the most reliable channels and one that offers a high level of accuracy in basic emotions detection. The common view in facial affect detection is that there is a distinctive facial expression associated with each basic emotion (Paul Ekman, 1984, 1992; Paul Ekman & Friesen, 2003). This distinctive expression is triggered for a short duration when the emotion is experienced and the decoding of this prototypical expression will help us to differentiate between the emotions.

The Facial Action Coding System (FACS) developed by Ekman and Friesen measures facial activity using facial actions as a technique to identify the emotion. The set of atomic facial actions are called Action Units (AUs) and the combination of AUs can be used to identify both basic (namely anger, disgust, fear, joy, sadness and surprise) as well as complex psychological states such as depression and pain. With the use of FACS, almost all anatomically possible facial displays can be manually coded by decomposing it into AUs. Some of the example AUs extracted from Cohn and Kanade database (Kanade, Cohn, & Tian, 2000) is shown in Figure 2-3.



Upper Face Action Units					
AU 1	AU 2	AU 4	AU 5	AU 6	AU 7
Inner Brow Raiser	Outer Brow Raiser	Brow Lowerer	Upper Lid Raiser	Cheek Raiser	Lid Tightener
*AU 41	*AU 42	*AU 43	AU 44	AU 45	AU 46
Lid Droop	Slit	Eyes Closed	Squint	Blink	Wink
Lower Face Action Units					
AU 9	AU 10	AU 11	AU 12	AU 13	AU 14
Nose Wrinkler	Upper Lip Raiser	Nasolabial Deepener	Lip Corner Puller	Cheek Puffer	Dimpler
AU 15	AU 16	AU 17	AU 18	AU 20	AU 22
Lip Corner Depressor	Lower Lip Depressor	Chin Raiser	Lip Puckerer	Lip Stretcher	Lip Funneler
AU 23	AU 24	*AU 25	*AU 26	*AU 27	AU 28
Lip Tightener	Lip Pressor	Lips Part	Jaw Drop	Mouth Stretch	Lip Suck

Figure 2-3: Examples of some AUs extracted from Cohn and Kanade's database (Kanade et al., 2000)

A comprehensive survey of vision-based affect detection has been conducted by Zeng et al. (2007b). They concluded firstly that most of the works still focus on the six basic emotions (due to their universal properties and the availability of the training and testing materials) with some works investigating mental states such as agreeing, concentrated, interested, confused and frustrated (Kapoor et al., 2007; Kapoor & Picard, 2005; Teeters et al., 2006; Yeasin et al., 2006).

One particular study by McDaniel et al. (2007) investigated facial features that accompanied deep learning in conceptual materials within an ITS - AutoTutor. Their results indicated that the set of affective states comprising of confusion, delight, and frustration with the exception of boredom can be discriminated from neutral by the system. However, investigating further into the system's ability to detect the individual affective state, only delight and confusion can be detected. Boredom was barely detectable and frustration detection was equivalent to random guessing. In perspective, Kapoor et al. (2007) used a combination of four sensors in his study for the detection of frustration. This illustrates the difficulty of detecting complex cognitive states such as frustration from facial features alone.

In the review by Zeng et al. (2007b), almost half of the studies relied on the use of acted or posed facial expressions. Posed facial expressions can be recognized automatically with relatively high accuracy but spontaneous expression recognition in less constrained setting remains challenging as the facial muscle actions are less exaggerated and individuals' facial expressions variances are larger. A less constrained setting would also mean that lighting and occlusion issues are more probable. However, with the growing recognition of the implications on detection accuracy when the trained systems are deployed in real-world environment, researchers are embarking on studies focusing on the detection of spontaneous facial expression instead e.g. (M. S. Bartlett et al., 2005, 2006a; M. S. Bartlett et al., 2006b). Sebe et al. (2007) created the first authentic facial expression database consisting of natural facial expressions linked to the emotional states experienced by their test subjects. Their test subjects were shown movie trailers of different genres and the true emotional states are obtained through interviews. Their results showed that all the basic emotions can be recognized with a high accuracy with surprise having the lowest accuracy of 81%

Overall, there has been significant progress in this area with some built systems offering real-time and robust detection e.g. (M. S. Bartlett et al., 2006a). However, researchers acknowledge that facial techniques still suffer from lighting and occlusion issues which will require either further technological development or the fusion of other modalities and contextual information (Calvo & D'Mello, 2010). It also seems that the use of facial expressions alone would not suffice for the detection of complex emotional states such as engagement and frustration.

2.6.2. Voice or Audio

Speech transmits affective information through the linguistic message (what is said) and the implicit paralinguistic features (how it is said) of the expression. The most popular paralinguistic features used in most studies are the prosodic features (e.g. pitch-related features, energy-related features and speech rate) and spectral features (e.g. Mel-frequency cepstral coefficients). Similar to facial expressions, most of the voice affect detection studies were conducted for detecting the basic emotions and were trained and tested by getting actors to speak from prescribed scripts with certain emotions e.g. (Austermann, Esau, Kleinjohann, & Kleinjohann, 2005; Kwon, Chan, Hao, & Lee, 2003).

The majority of the studies uses prosodic features e.g. (Ang, Dhillon, Krupski, Shriberg, & Stolcke, 2002; Austermann et al., 2005) but some researchers realized that the analysis of acoustic or prosodic features will not suffice for detecting the subtle changes in vocal affect. Arising from this, there were some studies that investigated into the combination of acoustic features and linguistic information to enhance vocal affect recognition performance. These include two studies which used spoken words and acoustic features (Litman & Forbes-Riley, 2004; Schuller et al., 2005) and another study by Schuller, Lang, and Rigoll (2002) which used both prosodic features and semantic features such as verbosity level, intention rate and repetition rate.

Overall, the interpretation of para-linguistic features has not yet been fully understood and researchers have not provided an optimal set of voice features that can discriminate emotions reliably (Juslin & Scherer, 2005).

2.6.3. Body Postures

In his book, Darwin (1969) wrote "Whenever the same movements of the features or body express the same emotions in several distinct races of man, we may infer with much probability, that such expressions are true ones, that is, are innate or instinctive", implying a connection between our mental states or emotions and our body movements. However, body postures have been overlooked as a serious contender when it comes to state of the art affect detection systems as compared to facial expressions.

Human bodies are large objects with multiple degrees of freedom, providing them with the capability of assuming a myriad of unique configurations (Bernstein, 1967). These static positions can be concurrently combined and temporally aligned with a multitude of movements, making posture a potentially ideal affective communicative channel (Coulson, 2004; Montepare, Koff, Zaitchik, & Albert, 1999). Furthermore, body postures can provide information that sometimes cannot be transmitted by facial expressions and speech. For example, the emotional type of a person can be distinguished from a long distance through body postures, whereas detection from facial expressions in a similar context would be difficult (Walk and Walters, 1988).

The most significant benefit to posture-based affect detection is that gross body motions are usually unconscious, unintentional, and therefore not susceptible to social editing as compared to facial expressions and speech intonation. Paul Ekman and Friesen (1969), in their studies of deception, showed that it is much more difficult to disguise deceit through body postures as compared to facial expressions.

Coulson (2004) conducted a study where he generated different postural static pictures of a mannequin viewed from three different angles using a computer animation software package. He then required his participants to map each set of static postural pictures into a discrete emotion out of the six basic emotions in order to investigate attribution of emotions to the different body postures. His results showed that some of the basic emotions (anger and sadness) with the exception of disgust were attributed to the same emotion by more than 90% of the samples. The results obtained are comparable to those of static facial expressions.

Mota and Picard (2003) presented a system that extract, process and model sequences of natural occurring postures for the purpose of interpreting affective states that occur during natural learning situations. Using two matrices of pressure sensors made by Tekscan, they captured and recognized both static posture positions and the sequences of postural behaviors of the participants. With the recognized postures sequences, they then associate the postures to the interest level of the participants using the Hidden Markov Model. Their results showed that the system was able to attain an accuracy of 82% in the recognition of interest levels of their participants.

S. D'Mello and Graesser (2009) extended Mota and Picard’s work by developing an intelligent tutoring system to detect cognitional and emotional states from the students’ gross body movements. From the measurement of the body pressure of the learner on the seat and back of the chair, they extracted two sets of features obtained from the pressure maps. The first set is the average pressure applied with the magnitude and direction of transforms in the pressure during the occurrence of affection. The second set examined the spatial and temporal properties of naturally occurring pockets of pressure. Machine learning techniques were used to differentiate boredom, confusion, delight, flow, and frustration from neutral with an average accuracy above 70%.

In summary, body posture based affect detection is still a growing and relatively under explored research area. The benefits of body posture based affect detection are that it is non-intrusive and is less prone to social masking. Social masking refers to the idiosyncrasies in expression of emotions due to familial, personal or culturally learned displayed rules.

Although the need for expensive equipment is a limiting factor in its development, the advances in gestural technologies may mitigate that.

2.6.4. Head pose

Some works were found in the literature utilizing head pose for detection of fatigue and attention level. A study by Vural et al. (2008) used machine learning algorithms to process a combination of head pose and thirty facial action units for the detection of driver's fatigue conditions. The system was able to predict sleep and crash episodes with a close to perfect accuracy of 98%. In another study detecting driving drowsiness, Ji, Lan, and Looney (2006) used Dynamic Bayesian Network, a probabilistic machine learning technique and features such as eye movements, eyelid movements, head movements, facial expressions and contextual information to present a theoretical framework for the driving drowsiness detection system.

The study by Asteriadis, Tzouveli, Karpouzis, and Kollias (2009) uses head, eye and hand movements processed with a neuro-fuzzy network to gauge children's level of interest and attention in the context of reading an electronic document. The results indicated that the proposed system detects attention level effectively.

A recent study by Adams, Mahmoud, Baltrusaitis, and Robinson (2015) shows that head motions and facial expressions convey emotional information that are complementary. The study uses both facial and head pose videos (facial videos with masking of the facial features). The emotion labels were obtained through crowd sourcing from the web. The results showed that some videos were recognizable by the face channel and not the head channel and vice versa. In addition, emotions such as bored, happy, sneaky and surprised can be detected from the face channel alone whereas interested and frustrated can be detected from the head channel alone. This indicates that the head channel carries emotional information that is not redundant to the face channel and some complex affective states such as interest and frustration may be linked better to head poses.

Similar to body posture, head pose is less prone to social masking. The head pose channel also carries emotional information that is complementary to the face channel. An added advantage is that it can be extracted from relatively cheap and easily available equipment e.g. a web camera.

2.6.5. Physiology

Psychophysiology is the branch of physiology that is concerned with the relationship between mental (psyche) and physical (physiological) processes; it is the scientific study of the interaction between mind and body. The field of psychophysiology draws upon the work of physicians, psychologists, biochemists, neurologists, engineers, and other scientists.

The field of psychophysiology has strong influence for affective computing research. It is concerned with the study of both emotions and mental states such as attention, perception memory, deliberation and problem solving. Some of the measures used to monitor physiological signals include Electromyogram (EMG) for the measurement of muscle activity, Electrodermal Activity (EDA) for the measurement of electrical conductivity as a function of sweat glands at the surface of the skin, Electrocardiogram (ECG) for the measurement of heart activity and Electroencephalography (EEG) for the measurement of brain activity.

There are a number of studies that make use of physiological measures for affect detection within the domain of Human Computer Interface. One of these is the study by Nasoz, Alvarez, Lisetti, and Finkelstein (2004) that uses non-intrusive physiological signals consisting of galvanic skin response, skin temperature and heart rate. The emotions were elicited through the viewing of movie clips and the results showed the highest accuracy of 92% for the affective state of sadness. Villon and Lisetti (2006) detailed a system that uses skin conductivity and heart rate for mapping between psychological response and the affective space but no results were presented.

Rosalind W. Picard, Vyzas, and Healey (2001) detailed a study that automatically recognizes eight user-defined affective states (neutral, anger, hate, grief, platonic love, romantic love, joy, and reverence) from a set of sensed physiological signals. Five physiological signals have been recorded: electromyogram from jaw (coding the muscular tension of the jaw), blood volume pressure (BVP), skin conductivity, respiration, and heart rate calculated from the BVP. One highlight of this study is that the data for this study was collected from a single subject over long time period of several weeks instead of multiple subjects over a shorter time period. The results showed an accuracy of 81%, almost six times better than chance.

Kim and André (2008) conducted a study that uses music to induce emotions in their 3 subjects. Physiological signals were acquired from their subjects from four biosensors: electromyogram, skin conductivity, electrocardiogram and respiration. The acquired signals were preprocessed using segmentation (into 160 seconds window), de-noising and filtering techniques before being classified using the Emotion-Specific Dichotomous Classification technique. They were able to achieve correct classification rates of 94%, 94% and 98% respectively for the 3 subjects.

Due to the lack of a neural model of emotions, few studies have employed the use of EEG. In the study conducted by AlZoubi et al. (2009), a wireless sensor headset was used to acquire real-time EEG signals with self-elicited emotions from three different participants. The accuracy varies considerably between subjects with poor recognition accuracy for one of the subject. The best accuracy achieved was 66% for another subject. Nevertheless, this study shows that affect recognition from EEG signals may be possible as indicated by the better than baseline results.

Cost, time resolution, obtrusiveness and the complexity of setting up the experimental protocols are practical issues that hinder the development of applications utilizing physiological techniques. To illustrate, for some of the studies discussed above, the subjects have to attach gelled electrodes and sensing bands on parts of their bodies and future users of systems built on these physiological sensors may find their use intrusive. However, with standardization in the stimulus for identification of physiological patterns, the development of wearables and a physiological emotional model, there should be more advancement within this area in the foreseeable future.

2.6.6. Input devices

Traditionally, keystrokes and mouse clicks analysis are used in the domain of computer security for the verification of user identity. They are in fact well researched into as a form of biometrics for user authentication and identification (P. Dowland, Furnell, & Papadaki, 2002; Joyce & Gupta, 1990). However, its potential use in the affect detection arena remains relatively under explored and only a few studies are found in the literature for this purpose.

Epp, Lippold, and Mandryk (2011) conducted a study to investigate the efficacy of keystroke dynamics for determining emotional state. Keystrokes were gathered as the users perform their daily tasks on their computer over an average period of four weeks with random prompts to remind the users to input their emotional states. Only users who completed more than one questionnaire per day for at least half the study duration were included in the subsequent analysis (which led to half of the population being eliminated). Using a decision tree classifier and the extracted latency and duration features, they achieved accuracies ranging from 77% to 88% for six emotional states – confidence, hesitance, nervousness, relaxation, sadness and tiredness.

In the study by Vizer, Zhou, and Sears (2009), timing, keystrokes and linguistic patterns from free texts were used to detect the presence of cognitive or physical stress. The cognitive stress and physical stress conditions were induced by requiring subjects to perform mental calculations and cardiovascular exercises respectively. A classification rate of 75% was obtained for cognitive stress and 62.5% for physical stress. The results of this study provided support for the use of keystroke and linguistic features in affective computing applications.

Zimmermann, Guttormsen, Danuser, and Gomez (2003) designed an experiment that uses film clips to induce mood states while requiring ninety-six subjects to shop on an e-commerce site for office supplies. The 5 different mood states are PVHA, PVLA, NVHA, NVLA, nVnA (P – positive, N – negative, H – high, L – low, n-neutral, V – valence, A – arousal). During the experiment, keyboard and mouse actions were logged and physiological parameters such as respiration, skin conductance level and corrugator level were also measured. No results were published along with their study.

Keystrokes and mouse clicks can be collected from subjects in a non-obtrusive manner. In addition, the sensors – keyboards and mouse are cheap and easily available in every academic setup. Although keystrokes and mouse clicks have traditionally been used for user identification purposes in the computer security domain, some initial studies have been conducted on the viability of their use in affect sensing. The results from these studies are promising but these are initial stage studies which uses stimuli to induce the emotions of the subjects. Further work will be required to implement them in a naturalistic context.

2.6.7. Multimodal

Human affective expression consists of a complex activation and interaction of involuntary (e.g. physiological), semi-voluntary (e.g. facial expressions) and voluntary (e.g. key presses) signals (Paul Ekman, 1992; Rosenberg & Ekman, 1994). Although, most of the past studies on affect detection have focused on a unimodal approach – use of a single modality (e.g. facial features), it is commonly recognized that a multimodal approach - the use of multiple modalities for affect detection would more closely emulate human affect expression.

Scherer and Ellgring (2007) conducted a study to investigate the possibilities of using a combination of facial, vocal and body movements to discriminate among fourteen emotions. They recruited professional actors to recite sentences that portray the required emotions. The results showed that the multimodal approach attained a much higher recognition rate of 79% than the unimodal approach. The high recognition rate however, dropped to 49.1% when performing cross-validation. After performing a step-wise discriminant analysis, the top 10 features were entered into the model again and a cross-validation accuracy of 50.4% was obtained, slightly above baseline.

Many researchers recognize that the reliance on posed or acted emotions for affect detection will result in a deterioration in accuracy when the system is implemented in a real-life, natural environment. Thus, Kapoor and Picard (2005) investigated the use of facial, postural and information on the learner’s activity (contextual information) to discriminate whether a learner is interested or disinterested in the context of the solving of an educational puzzle. The significance of this study is that it does not require the use of stimulus or posed emotions and instead the emotion labels are provided by the teacher observers. Processing the features obtained with a mix of Gaussian Process model, Kapoor and Picard attained an accuracy of 86%.

Kapoor et al. (2007) extended his study (in 2005) by employing the use of a skin conductivity sensor and a pressure-sensitive mouse to predict children’s frustration while interacting with an avatar in the playing of the Tower of Hanoi game. The children self-report their frustration using a button on the interface and a prediction accuracy of 79% was reported. Fidgets, head velocity and ratio of postures were found to be the three most discriminative features. No single channel classification accuracies were reported, thus making it difficult to make a comparison on the delta that multimodal approach offers over a unimodal approach.

Arroyo et al. (2009) used four sensors – web camera (facial expression), conductance bracelet, pressure mouse, posture analysis seat to augment contextual information logs in an intelligent tutoring system. A set of four emotions namely confidence, frustration, excitement and interest were inferred from the physiological logs extracted from the sensors and compared to the self-reported emotions of the students. They reported the variances for the various combinations of sensors for each of the four emotions. The results showed that the combination of facial features and contextual information offered the best correlation with confidence, excitement and interest. Variances of 51.8%, 68.9% and 29.2% were obtained for confidence, excitement and interest respectively. Compared to a unimodal approach (using only contextual information), the multimodal approach offers significant enhancement in accuracy.

More recently, S. K. D'Mello and Graesser (2010) developed a multimodal system combining conversational cues, gross body language, and facial features. The results indicated that the face was the most diagnostic channel for spontaneous affect judgements (i.e. at any time in the tutorial session) while conversational cues were superior for fixed affect judgements (i.e. every 20 seconds in the session). It was also found that the accuracy of the multimodal model was higher than that of the unimodal model for fixed judgements but not for spontaneous judgements.

A general observation that we can conclude in the above studies is that multimodal model likely offers higher discrimination ability over a unimodal model even in a natural environment (without the use of posed emotion or stimulus). It also seems that complex emotions such as frustration can be recognized albeit at a lower accuracy using a multimodal approach. However, the accuracy of multimodal affect sensing varies widely across the discussed studies. In addition, the possibility of redundant information residing within the channels cannot be eliminated. More research is required to establish which sensors when combined together, offers complementary information and which are the ones that result in redundant information. The techniques for resolving this conflict also needs to be explored.

2.7. Summary

This chapter introduces the psychological theories that underpin the importance of emotions and the techniques of classifying emotions. The merits and demerits of discrete, dimensional and appraisal classification were also discussed. We then looked into the role of affect in ITSs and reviewed some of the ATSs within the literature. Next, we listed the various channels of sensing affect, discussed the related literature that use the individual sensor modes for affect sensing and analyzed the merits and demerits of each of the sensor modes. Lastly, we discussed the related studies on multimodal affect sensing – the combination of the various sensing modes which is hypothesized to offer a higher accuracy over unimodal techniques and identified some of the research gaps in this area.

Chapter 3. Investigating Viability of Keystrokes and the Granularity for Affect Detection

3.1. Introduction

As stated in the literature review chapter, keystrokes and mouse clicks have traditionally been used in the computer security domain for user identification and authentication purposes. Khan, Hierons, and Brinkman (2007) proposed and elaborated on the protocols of his study to investigate whether programmers' keystroke and mouse patterns can be used to predict their emotions but there was no subsequent paper published on the empirical results of the experiment. Zimmermann et al. (2003) also proposed a study that induces the emotions of participants using movie clips and then collecting the keystrokes and mouse logs while the participants are completing online shopping tasks. However, similar to Khan et al.'s study, there was no empirical results published subsequent to his study.

Expert human tutors can achieve learning gains of 2 sigma as they are adept at recognizing the affective state of the student and then dynamically adapting their tutoring responses to sustain the student's learning (Bloom, 1984). Thus, to enhance the learning of students, it is vital to respond to the affect of students in a sufficiently timely fashion. Early detection of frustration of students would afford the tutoring systems sufficient time to enact corrective scaffolding actions. It would be futile to intervene if the student has passed the point of no return and has already given up working on the problem totally. On the other hand, it would be impossible to deduce whether a student is frustrated with just a few seconds of captured keystrokes and mouse clicks data. An appropriate level of granularity for affect recognition would have to be established for effective tutoring intervention.

In this chapter, I first establish the viability of using a combination of keystrokes and contextual logs for the detection of frustration in the context of a tutoring system for Java programming. Next, I address the question on whether the addition of keystrokes enhances the accuracy of frustration detection as compared to the model using only contextual logs.

The detection of frustration in a tutoring exercise session will have to be sufficiently granular for it to be used effectively in tutorial intervention. However, increasing the level of granularity would be at the expense of the detection accuracy and a balance will have to be struck between accuracy and granularity. Having established the viability of using keystrokes and contextual logs for affect detection, I investigate into techniques of increasing the granularity of affect detection. This leads to the third research question, investigating into the level of granularity that can be achieved for affect detection using keystrokes, mouse clicks and contextual logs and the level of accuracy that can be achieved at the various granularity levels.

3.2. Setup

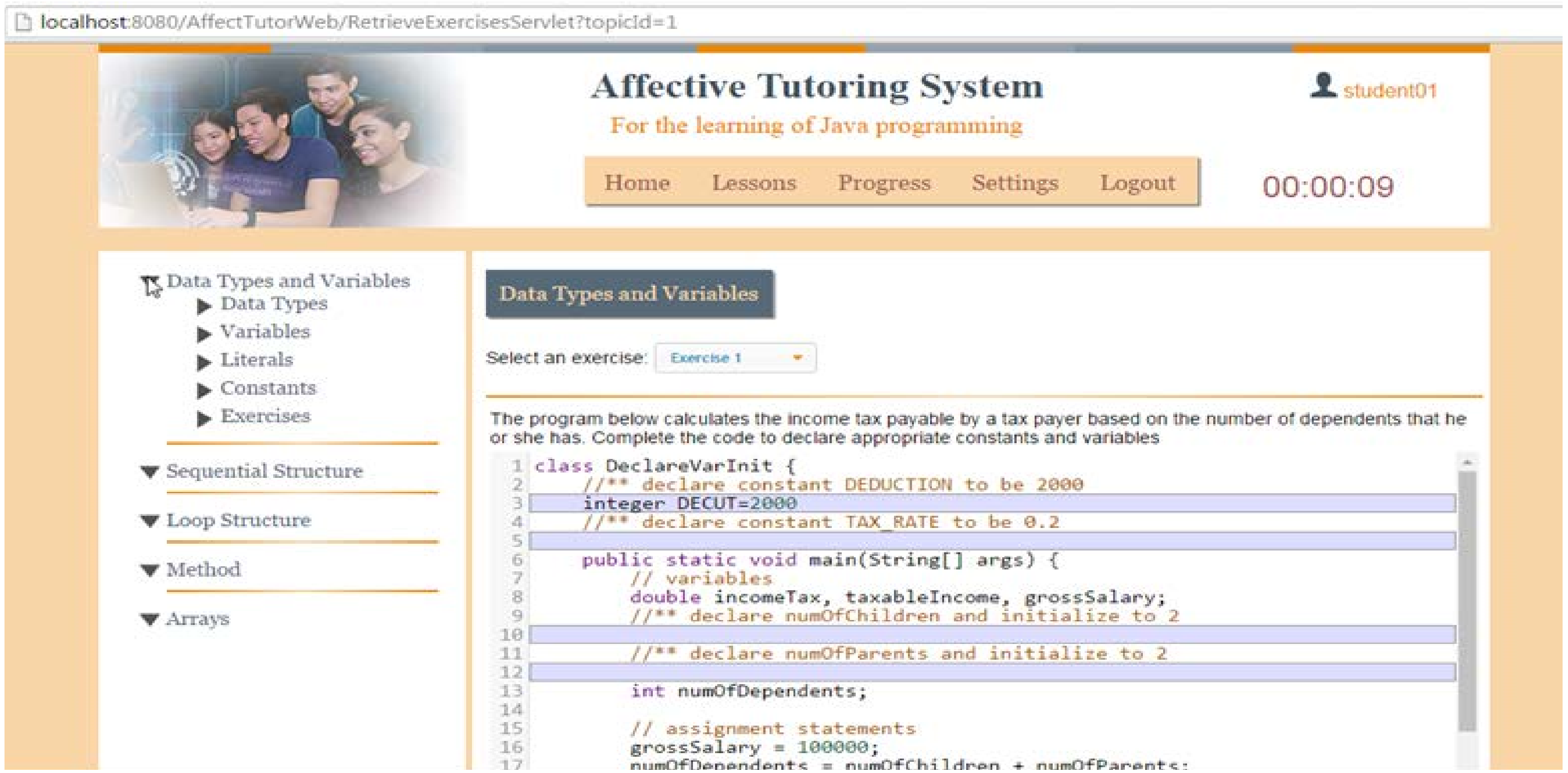
This work is compiled from data gathered in the study conducted in Nanyang Polytechnic, Singapore in the academic year 2014 and 2015. The data for the 14 students who were recruited in academic year 2014 were used in the first study for establishing the viability of keystrokes while the data for 24 students (10 more recruited in academic year 2015) were used in the second study for establishing an adequate affect granularity. The students were 18 years of age on average and were undertaking an Information Technology diploma in Nanyang Polytechnic. They had undertaken one semester of Java programming a semester ago. The studies were conducted in computer labs where the students were enrolled to work on programming exercises in a Java programming tutoring software. Before the start of the session, students were requested to fill in a form granting consent to participate in the study. The students were also briefed on the objectives of the study, what their roles were and what were they required to do. They were also guided on the various functions within the tutoring system. For each session, 5 students were working on the exercises due to constraint on the equipment (only 5 web cameras were available).

The web camera model is Logitech Pro C920 Full HD Webcam which can record at a resolution of 1920 x 1080 pixels. This web camera model was selected as it was the cheapest yet reliable web camera that can record at a full 1920 x 1080 pixels and was compatible with the software that were used in this experiment. For each terminal, the attached keyboard was a standard 104 keys US QWERTY keyboard and the attached mouse was a standard Logitech 2 button wheel mouse. There were 5 rows of tables in the computer lab used for this trial with 2 computer terminals set up for each row of table. The rows of tables were well separated such that each individual web camera which is mounted on top of the monitors is well positioned to capture the entire face of the student sitting directly in front of it. This is done to minimize the undesirable situation of recording multiple student faces in a single web camera. The lab is also a restricted entry room where only authorized students are allowed entry into it, thus minimizing interference of the trial by other non-participants. A video recording software – Camtasia Studio was used to record both the face and screen videos of students simultaneously.

The tutoring software had a total of 6 topics and a set of 2 programming exercises per topic, giving a total of 12 exercises to be completed. The topic contents and exercises were created by a group of lecturers in Nanyang Polytechnic School of Information Technology who had taught the subject for the past 3 semesters. The contents and exercises were stored in a MySQL database on the server. The topics covered the basics of Java programming and included the use of variables, loops, conditional statements and arrays. Conceptual instructional pages were also provided per topic and students could choose to read the conceptual instructional pages first, possibly to refresh their memory on the various programming syntaxes before attempting the exercises.

Each exercise was preloaded with a set of Java codes with missing lines in between (denoted with a comment stating that code is to be inserted) and students would have to fill in the missing lines of code to complete the exercise. Within each exercise page, students can click on a “Submit code” button to submit the code for compilation. They can then check the output window for the compilation output or errors. The compilation output will be verified against the required output when the student clicks on the “Check answer” button and will be marked as completed if the correct output was obtained. A screen shot of the exercise page is shown in Figure 3-1 below.

To achieve the compilation of the students' codes in situ, we will need to either re-create our own Java compiler or interface to a third party service that will perform the compilation of the codes and returns the results of the compilation. At the point when this tutoring software was created, there were no other alternatives but IDEOne Sphere Engine (*Sphere engine*, 2015) which could satisfy our requirement. The way it works is that when students clicked on "Submit code" button, the student's code will be submitted to IDEOne Sphere Engine through a web service call. The IDEOne Sphere Engine server will compile the code before returning either the compilation code for successful compilation or an error message for failed compilation.



The program below calculates the income tax payable by a tax payer based on the number of dependents that he or she has. Complete the code to declare appropriate constants and variables

```
class DeclareVarInit {
    //** declare constant DEDUCTION to be 2000
    integer DECUT=2000
    //** declare constant TAX_RATE to be 0.2

    public static void main(String[] args) {
        // variables
        double incomeTax, taxableIncome, grossSalary;
        //** declare numOfChildren and initialize to 2

        //** declare numOfParents and initialize to 2

        int numOfDependents;

        // assignment statements
        grossSalary = 100000;
        numOfDependents = numOfChildren + numOfParents;
```

Figure 3-1: Screen shot of the exercise page

The students were free to choose the sequence of working on the exercises and topics and they were also allowed to switch from one exercise to the next even if they had not completed the current exercise. The student's actions, keystrokes and the respective time stamps in which each action or keystrokes occurred within the tutoring software were logged in the database. Some examples of the actions that were logged include the start and end time of each exercise, the time stamp of each "Submit code" action and the time when the exercise was completed. Copying and pasting of code from other sources was a common behavior among novice programmers and that was captured in the keystroke logs as well. The list of features would be described in the sections below.

3.3. Annotation of frustration

As stated in the setup, both the students' screens and their facial expressions were recorded using web cameras attached on top of their monitors and each session lasted for an average of 60.5 minutes. The recordings were used for a retrospective annotation of frustration by 2 lecturer observers with an average teaching experience of five years. The 2 lecturer observers are colleagues who are teaching in the School of Information Technology in Nanyang Polytechnic (not involved in the research) and have received good teaching reviews from students. This retrospective observation technique was employed by a number of prior studies (Cetintas, Si, Xin, & Hord, 2010; A. Graesser et al., 2006; Mcquiggan, Lee, & Lester, 2007). Both the facial and screen videos were captured and used for the annotation of frustration as it is difficult to ascertain whether the student is frustrated from facial expressions alone. The addition of screen video enhances the annotation accuracy by providing an additional source of information into the cause of the student's frustration. For example, if the observer observed signs of frustration from the facial video but was not so sure, the observer could then confirm that the student was indeed frustrated (from the screen video) if for example, he or she observed that the student had been trying to rectify a compilation error for quite some time without success. A custom program was developed in Visual Studio C# for the observers to annotate points in the video at which frustration of the students were observed. A screen shot of the custom annotation program is shown is Figure 3-

2. The observer will have to first select the video file to be played and the program will automatically populate the start time and student identifier from the file name of the video file. At any point during the replay when frustration was observed, the observer can just click on submit button in the Affect panel. The details of the annotation (student identifier, date and time when frustration was observed) will be recorded in the MySQL database. After the first annotation session by each of the observer individually, the two observers had another joint session to reconcile the annotations that differ.

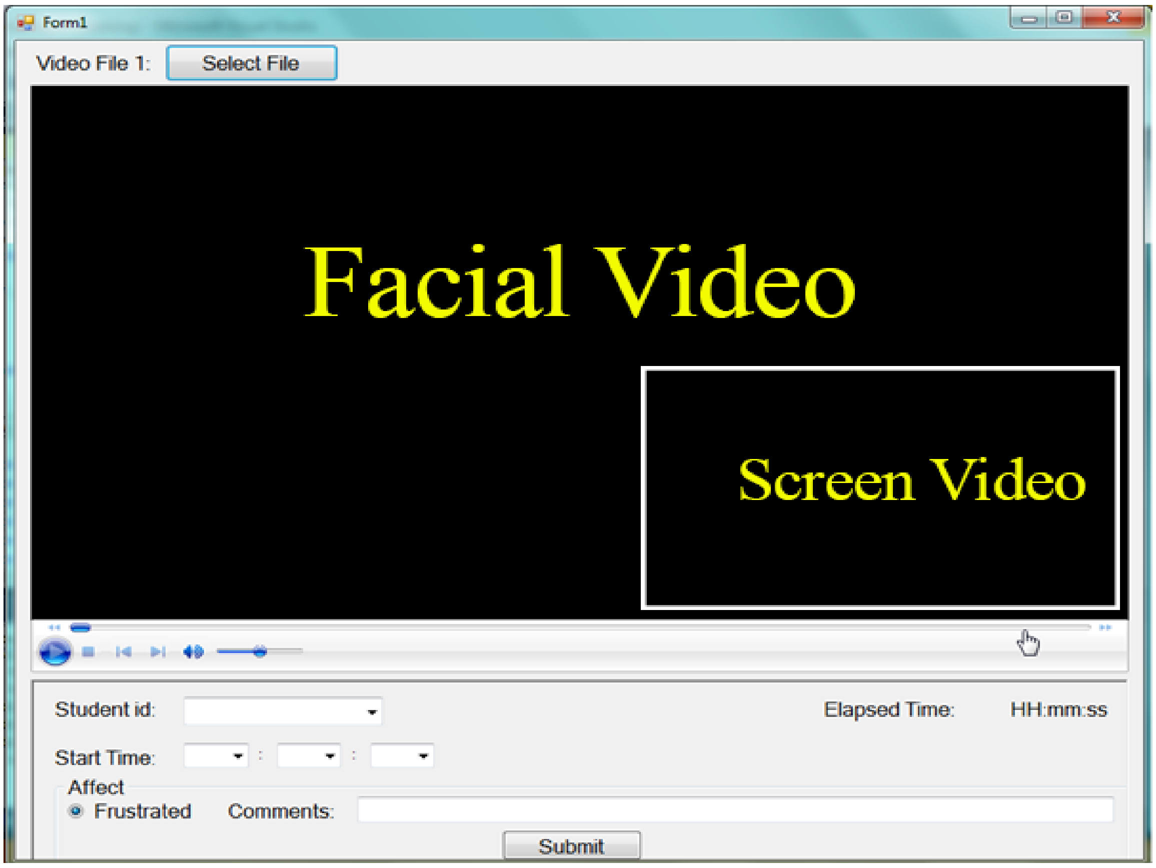


Figure 3-2: Screen shot of custom annotation program

Some examples of the frustration behaviors noted in the session included use of expletives, long sighing, excessive gesturing and roughly ruffling through hair while visibly distressed. The observers noted that these behaviors are usually accompanied with the encountering of compilation errors or being stuck in a particular point in the exercise for a period of time (which can be observed from the students' screen videos). The observations were recorded with a time stamp and this time stamp was used for synchronizing with the captured contextual and keystroke logs.

In the first study, the granularity of the detection of frustration is on a per exercise level. Each student works on an average of about 4 exercises in each session thus giving a total of 56 (N = 14 students x 4 exercises) records. A student working on a particular exercise would be annotated as frustrated when at least 1 instance of frustration was noted while working on the exercise. In the second study, various fine granularity levels within the range of 30 seconds to 180 seconds were investigated.

3.4. Modeling

In this section, I first detail the derivation of the features that are used in the modeling of frustration followed by the machine learning techniques that are used in both of the studies. In study 1, 2 models – one that uses only contextual features and another which uses both contextual and keystrokes features are compared to establish the viability of keystrokes for frustration detection and whether the addition of keystrokes enhances the accuracy of detection. In study 2, the machine learning models of Bayesian Networks and Naïve Bayes (designated as the baseline model) are used for investigating whether finer granularity of detection can be achieved at an appropriate accuracy level. The models built using Bayesian Networks are then compared against the baseline model using Naïve Bayes. The additional feature processing performed in Study 2 is also detailed.

3.4.1. Contextual Features

The list of contextual features includes the completion status of the exercise, the number of submissions for compilation of the exercise, the number of switches between the exercise and other exercises, the difficulty level of the exercise and the effort as shown in Table 3-1 below. The completion status is a categorical variable that denotes whether the exercise attempt was completed. A completed exercise attempt is one that produces the expected program output. The number of submissions for compilation is a discrete variable that denotes the number of times the student has submitted the code for compilation for the designated exercise. During the course of the study, I observed that students clicked on the submit for compilation button several times within a span of two seconds, thinking that more clicks would help to speed up the compilation time. Thus, to prevent duplicate counting, all submissions for compilation logs (by the same student for the same exercise) time-stamped within duration of two seconds are only counted as one submission for compilation. Although multiple clicking on the compilation button within a short span may be an expression of frustration, clarification with the students after the trial indicated that rather than being frustrated, they thought that more clicks will speed up the compilation.

Contextual Features	**Description**
complete_status	The completion status of the exercise which denotes whether the exercise was completed or not.
no_compilations	The number of times the student submit the code for compilation.
no_switches	The number of times the student switches between various exercises.
diff_level	The difficulty level of the exercise the student is working on.

Table 3-1: List of contextual features

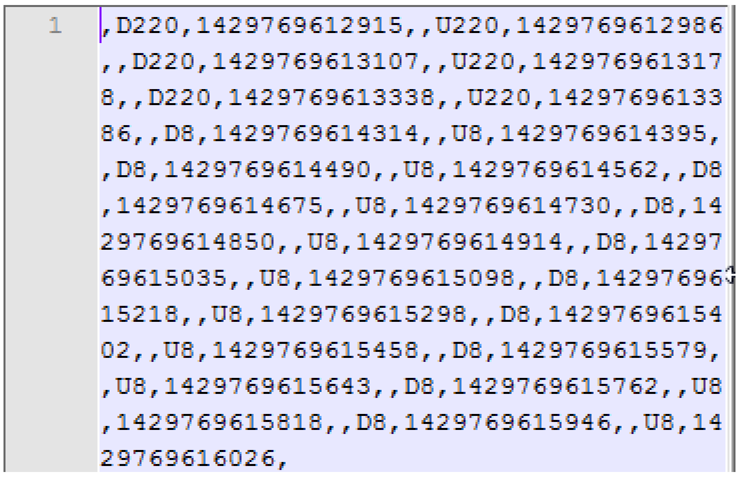
As lecturers, we usually observe during face to face tutoring sessions that students who did not manage to complete the exercises will be experience more incidences of frustration. These are usually the students who have difficulty with solving the exercise and would most probably give up working on the exercise (e.g. by switching to other exercises with the hope that the next one will be easier) after some time. Thus, completion status of the exercise is included in the list of features to detect frustration of students. During the sessions, students were allowed to switch between exercises even if they had not completed the exercise that they were then working on. The number of switches is a discrete variable that tracks the number of switches to other exercises from the designated exercise. For difficulty level, the mean of the time spent on each exercise by the students is computed and the difficulty level is derived from the number of standard deviations from the mean exercise time as shown in Table 3-2. The effort is an interaction term between the difficulty level and the completion status. The intuition is that a student would perceive that more effort is required to complete a difficult exercise than one that is easy. In addition, comparing two students working on an exercise of the same difficulty level, the effort of the student who completed the exercise in a longer time would be higher than that of the student who completed the exercise in a shorter time.

Number of standard deviations from mean exercise time	Difficulty level
Less than 0	1
Between 0 to 1	2
More than 1	3

Table 3-2: Difficulty level of exercise

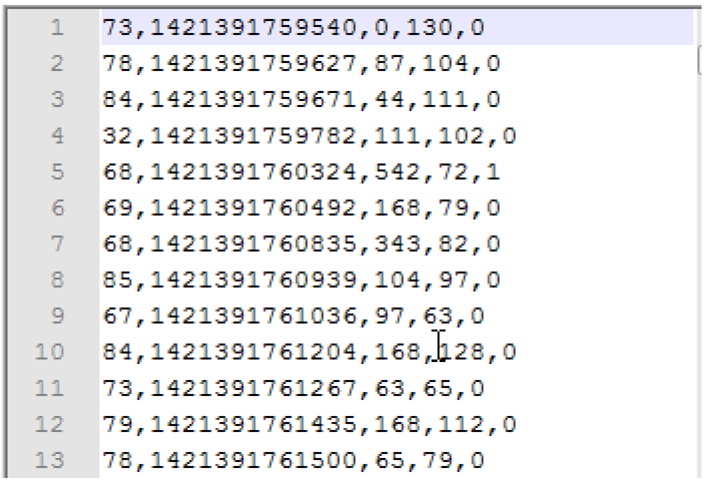
3.4.2. Keystroke Features

Keystrokes are captured only when the students are working on the exercises. A JavaScript function running on the client end captures and sends the raw key information to the server. At the server end, a program (servlet) processes the raw keystroke information to derive details such as the keystroke, time stamp denoting the time of depression of the key up to the milliseconds, latency (duration for the time when the key is depressed till the time the key is released) of the key and whether modifier keys such as shift or control keys were depressed. This processed information is then written to a log file on the server. A screen shot of the raw keystroke file and processed keystroke file is shown in Figure 3-3 and Figure 3-4 respectively. The format for the raw keystroke information and the processed keystroke log file on the server is shown in Table 3-3.



```
,D220,1429769612915,,U220,1429769612986
,,D220,1429769613107,,U220,142976961317
8,,D220,1429769613338,,U220,14297696133
86,,D8,1429769614314,,U8,1429769614395,
,D8,1429769614490,,U8,1429769614562,,D8
,1429769614675,,U8,1429769614730,,D8,14
29769614850,,U8,1429769614914,,D8,14297
69615035,,U8,1429769615098,,D8,14297696
15218,,U8,1429769615298,,D8,14297696154
02,,U8,1429769615458,,D8,1429769615579,
,U8,1429769615643,,D8,1429769615762,,U8
,1429769615818,,D8,1429769615946,,U8,14
29769616026,
```

Figure 3-3: Raw keystroke file



```
73,1421391759540,0,130,0
78,1421391759627,87,104,0
84,1421391759671,44,111,0
32,1421391759782,111,102,0
68,1421391760324,542,72,1
69,1421391760492,168,79,0
68,1421391760835,343,82,0
85,1421391760939,104,97,0
67,1421391761036,97,63,0
84,1421391761204,168,128,0
73,1421391761267,63,65,0
79,1421391761435,168,112,0
78,1421391761500,65,79,0
```

Figure 3-4: Processed keystroke file

File type	**File format**	**Description**
Raw key stroke file	Up/down, key code, time stamp	Up denotes that the key is released. Down denotes that the key is depressed.
Processed key stroke file on server	key code, time stamp, flight time, latency, shift/control key depressed	key code is the ASCII code of the key flight time is the duration of time when a key is depressed till the next key is depressed

	latency is the duration of time when a key is depressed till the time the same key is released.

Table 3-3: Details of keystroke files format

The duration or flight time of the key is the duration from the time a key is depressed to the time when the next key is depressed. The keystroke log files on the server for all the students are consolidated and processed using a batch program to calculate the mean, median and frequency of the different groups of keys (e.g. alphanumeric keys, navigation keys, backspace keys e.t.c.). The aggregated keystroke features that were derived and used subsequently in the modeling for study 1 and 2 are detailed in Table 3-4 and Table 3-5 respectively. The duration between keys refers to the flight time (the duration from the time one key is depressed to the time the same key is released) and the wait duration refers to the duration in which no key was depressed. The keystroke duration and latency features are keystroke dynamic features that were used in previous studies in the domain of user identification and authentication (Bergadano, Gunetti, & Picardi, 2003; M. Brown & Rogers, 1993; P. S. Dowland & Furnell, 2004) and thus included in this research. In addition, the backspace, delete and number of instances of copy of paste are deliberately extracted as features in this research as deletion of codes and copy and pasting behaviors are common programmer’s actions which may precede or provide clues to the incidence of frustration.

For study 2, mouse click features were included as well. Mouse clicks were captured only when the students were working on the exercises. A JavaScript function running on the client end captures and sends the raw mouse clicks information to the server. At the server end, a program (servlet) processes the raw mouse clicks information which consists of the coordinates of the mouse click, characteristics of the mouse click (single click or double clicks) and time stamp denoting the time of depression of the mouse click. This processed information is then written to a log file on the server. The mouse clicks files for students on the server are then consolidated and processed using a batch program to derive the total number of clicks, number of clicks less than 2 seconds and the number of double clicks.

Aggregated keystroke features	**Description**
num_keys	Total number of keys recorded
mean_dur_incl_wait	Mean duration between keys
mean_dur_excl_wait	Mean duration between keys (excluding duration longer than 1 second)
median_dur_incl_wait	Median duration between keys
median_dur_excl_wait	Median duration between keys (excluding duration longer than 1 second)
mean_alpha_key_dur	Mean duration between alphabetic keys
median_alpha_key_dur	Median duration between alphabetic keys
mean_wait_interval	Mean wait duration (duration longer than 1 second)
mean_pause_freq	Number of key pairs with duration longer than 1 second
backspace_freq	Number of backspace keys
delete_freq	Number of delete keys
arrow_freq	Number of arrow keys (up, down, left and right)
copynpaste_freq	Number of copy and paste instances
shift_freq	Number of shift modifier keys
num_wait1to2	Number of wait duration from 1 to 2 seconds
num_wait2to5	Number of wait duration from 2 to 5 seconds
num_wait5to10	Number of wait duration from 5 to 10 seconds
num_waitgt10	Number of wait duration greater than 10 seconds

Table 3-4: List of aggregated keystroke features used in Study 1

Feature Name	**Description**
numKeystrokes	Number of keystrokes
medianLatencyWithWait	Median latency including wait

maxLatencyWithWait	Maximum latency including wait
stdLatencyWithWait	Standard deviation of latency including wait
medianLatencyWithoutWait	Median latency excluding wait
maxLatencyWithoutWait	Maximum latency excluding wait
stdLatencyWithoutWait	Standard deviation of latency excluding wait
medianWaitInterval	Median wait interval
maxWaitInterval	Maximum wait interval
stdWaitInterval	Standard deviation of wait interval
meanPauseRate	Mean number of pauses
backspaceFreq	Number of backspace key
medianBackspaceLatency	Median latency of backspace key
maxBackspaceLatency	Maximum latency of backspace key
stdBackspaceLatency	Standard deviation of backspace key
deleteFreq	Number of delete key
arrowFreq	Number of navigation key
copyPasteFreq	Copy and paste frequency (Ctrl-C and Ctrl-V)
shiftFreq	Number of shift key
medianNonNavKeyLatency	Median number of non-navigation key
maxNonNavKeyLatency	Maximum number of non-navigation key
stdNonNavKeyLatency	Standard deviation of number of non-navigation key
wait1to2	Number of wait between 1 to 2 seconds
wait2to5	Number of wait between 2 to 5 seconds
wait5to10	Number of wait between 5 to 10 seconds
waitgt10	Number of wait greater than 10 seconds

medianFlightTime	Median flight time
maxFlightTime	Maximum flight time
stdFlightTime	Standard deviation of flight time
numClicks	Number of mouse clicks
numClicksLT2Sec	Number of mouse clicks less than 2 seconds
numDblClicks	Number of double mouse clicks

Table 3-5: List of aggregated keystrokes and mouse click features used in Study 2

3.4.3. Study 1 – Logistic Regression

In study 1, logistic regression is used to model frustration (on a per exercise level). It is used here as it is generally well-suited for describing and testing hypotheses about relationships between a categorical outcome variable and one or more categorical or continuous predictor variables. It is also a technique that has seen frequent use in both the social sciences and educational research domains (Peng, Lee, & Ingersoll, 2002). The outcome variable in this study - existence of frustration is binary.

Over-fitting, which refers to the problem of the model having an excellent fit to the training data but yet not fitting well to unseen data in the future is an issue that can occur in situations where data is sparse. Regularization is a technique to control the over-fitting problem by penalizing large values of the predictor coefficients (Hastie, Tibshirani, & Friedman, 2009). Due to the limited number of data instances (N=56) and to prevent over-fitting, I employ the technique of lasso regularization in this study.

The logistic regression is expressed as **Equation 1** below.

$$P(Y_i = 1 / X_i) = f(X'_i \beta) = \frac{\exp(X'_i \beta)}{1 + \exp(X'_i \beta)}$$ **Equation 1**

Logistic regression is usually fitted by maximum likelihood and the log likelihood function is given in Equation 2. To solve for β, we will take its derivative and set it to zero for a solution without regularization.

$$l(\beta|D) = -\sum_{i=1}^{m}\{y_i \log[f(X_i'\beta)] + (1-y_i)\log[1-f(X_i'\beta)]\} \qquad \textbf{Equation 2}$$

where $y_i$is the response at observation i, X_i is the data vector and β is the coefficient matrix.

With lasso regularization, we will add in a regularization term to the log-likelihood function in Equation 2, obtaining β by using the minimization function below.

$$\beta = \arg\min\ \{\ l(\beta|D) + \lambda \sum_{j=1}^{n} |\ \beta_j|\ \} \qquad \textbf{Equation 3}$$

where λ is the non-negative regularization parameter.

The minimization function in Equation 3 can be solved by a non-linear programming method. To prevent over-fitting and ensure the generalizability of the model, the data set is divided into a training set (80% of the data set) and a test set (20% of the data set). A 2-fold cross validation technique is then applied for the selection of the regularization parameter λ on the training set. In this study, 2 models – one that uses only contextual features (with a feature dimension size of 5) and another which uses both contextual and keystrokes features (with a feature dimension size of 23) are compared.

3.4.4. Study 2- Bayesian Network

In study 2, the Bayesian Network (BN) is used to model frustration for investigation of the various granularity levels. Study 1 investigates the frustration detection on a per exercise level whereas in study 2, I investigate into increasing the granularity of detection through additional processing techniques detailed in 3.4.6. A Bayesian Network (BN) is a directed acyclic graph in which nodes represent domain variables and arcs represent conditional dependencies. It enables an effective representation and computation of the joint probability distribution over a set of random variables (Pearl, 1988). A property of BN – each variable is independent of its non-descendants given its parents, is often used to reduce the number of parameters to characterize the joint probability distribution of the variables. This reduction provides for an efficient computation of the posterior probabilities given the evidence. The formula for a BN consisting of n nodes with random variables $(\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_\mathrm{n})$ is given in Equation 4.

$$p(x_1, x_2, \dots, x_n) = \prod_{i=1}^{n} p(x_i | x_{P_i})$$

Equation 4

where $\mathrm{p}(\mathrm{x}_\mathrm{i}|\mathrm{x}_{\mathrm{P}_\mathrm{i}})$ is the local probability distribution associated with node i and P_i is the set of indices labeling the parents of node i (Jordan, 1999).

BN is chosen here as it offers several advantages over other data analysis techniques such as regression (Heckerman, 2008). Firstly, it can handle incomplete data sets. For a regression problem where two of the explanatory variables are strongly anti-correlated, the regression model will produce an inaccurate prediction when one of the inputs is not observed. This is not a problem for BN as it can model the correlation between the 2 explanatory variables. For our problem where one or more sensing mode may not be available in a time window, this is advantageous. For example, keystrokes and mouse clicks may not be available for every time window. Secondly, BN allows us to discover causal relationships. This would allow us for example to know whether frustration causes the number of compilations to increase. This is possible even when no experiment is available about the effects of the exposure. Thirdly, BN allows for the incorporation of domain knowledge and data into the model. The prior or domain knowledge can be valuable in situations where data is scare and BN allows for the combination of prior knowledge and data through Bayesian statistical techniques. In our model, we use an initial domain model to seed the learning of a final BN structure. Lastly, BN offers an efficient approach for preventing overfitting of the data.

Thus for the above reasons, BN is used to model the occurrence of frustration in this study. The Bayes Net Toolbox, an open-source Matlab package by Kelvin Murphy (Murphy, 2001) was used to develop and test the model in Matlab. The structure of the Bayesian network for our study is learned using the Hill Climbing algorithm passing in an initial domain expert structure in the Bayes Net Toolbox. The learned BN structure for the modelling of frustration is shown in Figure 3-5.

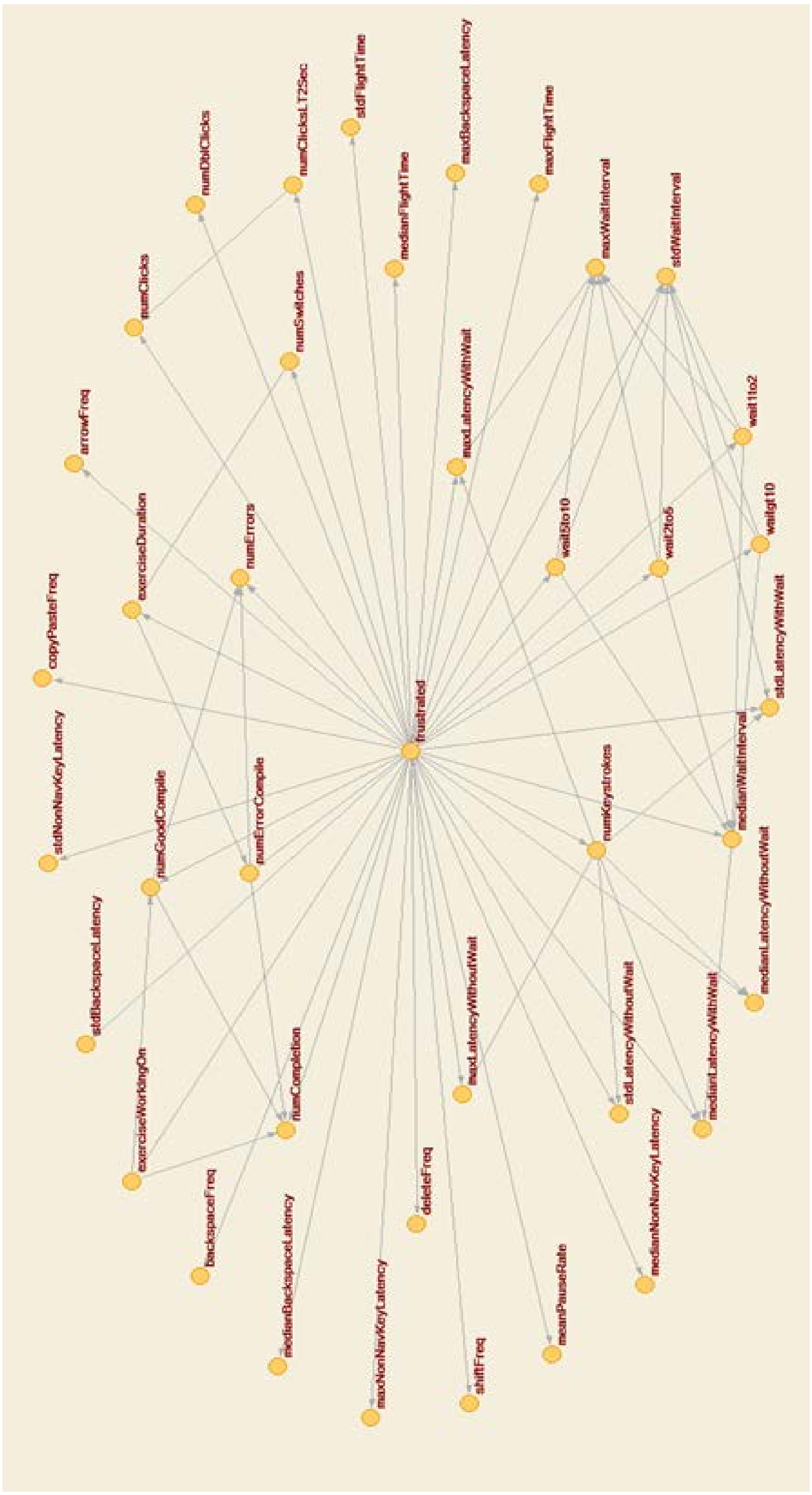


Figure 3-5: Bayesian Network Structure For Modeling Frustration

3.4.5. Study 2 -Naïve Bayes

In study 2, Naïve Bayes is used as the baseline model for comparison against the Bayesian Network model. Naïve Bayes is a classifier that is very popular and frequently applied in document classification (Nigam, McCallum, Thrun, & Mitchell, 2000; Sahami, Dumais, Heckerman, & Horvitz, 1998). It is based on Bayes Rule and a conditional independence assumption which states that each feature X_i is conditionally independently of all the other features given the class variable C. Bayes theorem states that

$$P(C = c_k|\boldsymbol{X} = \boldsymbol{x}) = \frac{P(\boldsymbol{X} = \boldsymbol{x}|C = c_k)\, P(C = c_k)}{P(\boldsymbol{X} = \boldsymbol{x})}$$ **Equation 5**

With the conditional independence assumption,

$$P(\boldsymbol{X} = \boldsymbol{x}\,|C = c_k) = \prod_i P(X_i = x_i|C = c_k)$$ **Equation 6**

For discrete valued inputs, Naïve Bayes can be solved using Maximum Likelihood Estimation. Despite its simplicity and ease of implementation, Naïve Bayes classification works surprisingly well. For these reasons, Naïve Bayes is used as a baseline model for comparison against Bayesian Network model in study 2.

3.4.6. Study 2- Additional Feature Processing

This section details the additional features processing performed in study 2 – synchronization of features with frustration annotation using sliding windows, baselining of keystroke rates and discretization of features. With the additional processing, the number of data instances is 4353 with a feature dimension size of 28.

The features extracted from the students' interaction, keystrokes and mouse logs are aggregated into sliding window sizes of 30 seconds, 60 seconds, 90 seconds, 120 seconds, 150 seconds and 180 seconds with an overlap of two-thirds of the window size. This is done with a custom developed Matlab script. If the time at which frustration is observed falls in the overlap area of 2 consecutive window slices, both window slices would be annotated as slices in which the student experiences frustration. Alternatively, if the time at which frustration is observed falls outside the overlap area, only the time window slice in which it occurs in will be annotated as the slice in which the student experienced frustration. This is illustrated in Figure 3-6 below.

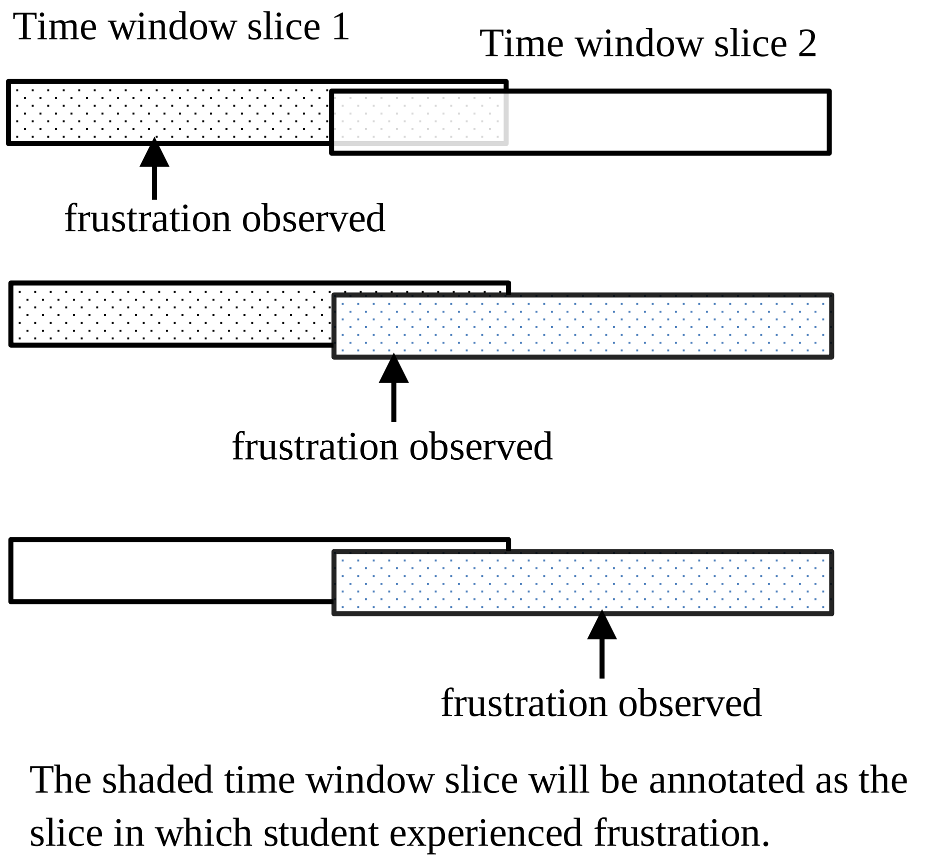

Time window slice 1
Time window slice 2
frustration observed
frustration observed
frustration observed
The shaded time window slice will be annotated as the slice in which student experienced frustration.

Figure 3-6: Overlapping time window slices for annotation of frustration

Establishing an optimal time window of affect detection is critical as it informs on the affective state of students in a timely manner. A larger time window width would mean that students' frustration was left unattended for a long period of time and that increases the risk of the students giving up totally on the learning task. On the other hand, a shorter time window width would result in lower accuracy and performance as lesser data is accumulated within the shorter time window for analysis. Thus, there is a need to evaluate the optimal time window size for the model which balances timely detection of students' affective state with the accuracy of detection.

In cases where keys are sparse for a period, a larger time window width would ensure that more keys are accumulated in a time window which would in turn lead to a more accurate detection of students' affective state. The mean number of keys for each time window slice is shown in Table 3-6. The disadvantage of a larger time window width is that the model will need a longer period of time to establish that the student is frustrated and if this time window is too long, the student may have already given up.

Time Window (in seconds)	Mean number of keystrokes
30	20.74
60	30.87
90	40.79
120	50.88
150	58.16
180	66.84

Table 3-6: Mean number of keystrokes by time window width

An observation noted during the trials is that when students are frustrated, the frustration may be manifested in their actions leading up to the observation and that it also persists for a period of time after the observation. To cater for this, the time window slices are overlapped such that if frustration is observed in the overlapped period, both the keystrokes in the time window slice leading up to the observation and the keystrokes in the time window slice after the observation will be included in the evaluation of the affective state (frustrated versus non-frustrated).

Innately, different students may type at different speeds even when they are in a neutral affective state. In order to mitigate the effect of the different typing speeds of students, we will have to equalize the keystroke latencies across students. To achieve this, the keystroke latencies for individual students are divided by their individual baseline latency. I have devised the following algorithm to calculate the baseline latency for each student.

1. Eliminate all keystroke data with duration greater than or equals to 1 second.
2. Eliminate all navigation keys, backspace and delete keys.
3. For each window interval of 10 keys,
    a. Calculate the mean of the keys duration and store in array.
    b. Move window by 2 keys.
4. Baseline latency = median of the lowest 30 mean key durations calculated in 3(a).

All instances with no keystrokes recorded were eliminated from the data set. By eliminating the instances with no keystrokes, an enhanced classification performance is achieved as compared to the use of the entire data set. To illustrate, for the time window width of 120 seconds, the AUC figure obtained with the exclusion of instances with no keystrokes is 0.81 as compared to 0.54 when all data instances are included. As such, the model can only detect incidences of frustration at an acceptable level of accuracy when there are keystroke activities within the designated time window. To detect incidences of frustration during the period when the students are not typing, it may be necessary to complement the detection with other sensing modes e.g. facial expressions.

The extracted features were discretized using an unsupervised discretization technique – equal frequency binning. Discretization is the conversion of continuous random variables into discrete nominal variables and is a pre-processing step that is commonly utilized in modeling BNs when the continuous random variables do not fit into a Gaussian distribution (John & Langley, 1995; Kotsiantis & Kanellopoulos, 2006). In our study, the features do not fit into a Gaussian distribution and thus, the need for discretization. The supervised discretization technique – Class Attribute Interdependence Maximization (Kurgan & Cios, 2004) was applied but the results were less satisfactory as compared to that obtained using the Equal Frequency Binning technique. The Equal Frequency Binning discretization technique which divides the data into m groups such that each group has the same number of values was thus chosen as the discretization technique. The value m=5 was used in this study.

3.4.7. Cross validation

In building a machine learning model, it is important that the trained model can generalize to new data sets. Overfitting of the model occurs when the trained model does not fit an independent sample of data (taken from the same population as the training data) as well as it fits the training data or data that it has been trained on (Marsland, 2015). To reduce overfitting, cross validation is frequently used. Cross validation works by holding a set of data known as the test set to validate the model on. In our studies, we used k-fold cross validation to test the generalizability of our models. In k-fold cross validation, the original data set is randomly partitioned into k equal-sized subsamples. Out of the k subsamples, 1 subsample is used as the validation data for testing the model while the rest are used for training the model. This is then repeated for all the different subsamples. The model that produces the lowest validation error is tested and used.

3.5. Results and Discussion

3.5.1. Study 1

For study 1, the results for the two models- one that uses only contextual features and another that uses both the contextual and keystroke features are presented in Table 3-7. As stated in the introduction, I would like to address firstly whether frustration can be detected from these features and secondly, whether the addition of keystroke features improve the accuracy of the detection.

Performance Measures	**Model**		
	Contextual only	*Contextual and keystroke*	*% difference (compared to contextual only)*
Accuracy	0.583	0.667	14.40%

Precision	0.571	0.625	9.50%
Recall	0.667	0.833	24.90%
F1 measure	0.615	0.714	16.10%
AUC	0.514	0.722	40.50%

Table 3-7: Performance of contextual only and combined with keystroke models

Using lasso regularization, for the contextual features only model, two features which have non-zero coefficients remain – number of submissions for compilation and effort. For the contextual and keystroke features model, the six features which have non-zero coefficients are number of submissions for compilation, effort, number of wait durations from 1 to 2 seconds, number of backspace keys and total number of keys. This corroborates with the research of Vizer et al. (2009) who in his study, states that the rate of mistakes (which equates to the use of backspace key to correct the mistake) and the pause length decreases when subjects of his experiment were required to perform mental calculations and numbers recall for the induction of cognitive stress. He concludes from the stress response model of Yerkes and Dodson (1908) that the amount of cognitive stress induced was just sufficient to enhance their performance. For my study where subjects being novice programmers are required to solve and troubleshoot programming exercises, the stress and frustration would be considerably higher than numerical recall or mental calculations, putting subjects into the right half of the Yerkes and Dodson curve shown in Figure 3-7 and thus resulting in lowered performances.

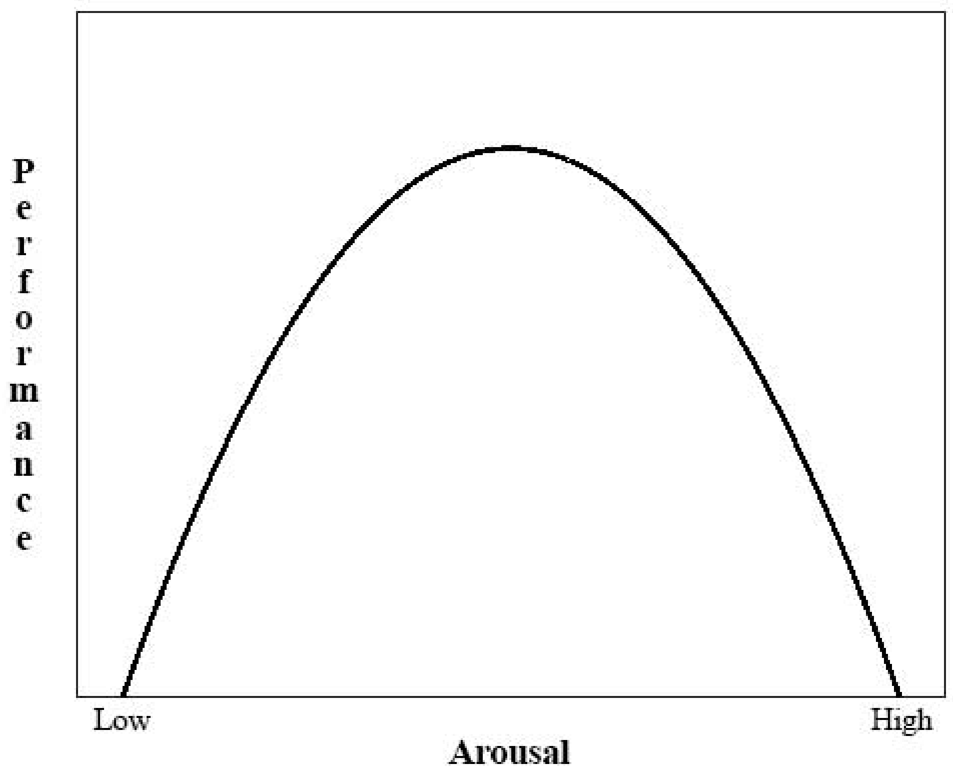


Figure 3-7: Yerkes-Dodson Law

To further confirm the results of the lasso regularized logistic regression, the graphical statistical technique of notched box plot is utilized. Notched box plot is an informal graphical statistical technique developed by Chambers (1983) for a test of significance of the difference in median between 2 distributions. If the notches of the 2 boxplots do not overlap, then we can reject the null hypothesis that the 2 medians are the same; in other words, the difference between the 2 medians is statistically significant at approximately 95% confidence. The formula for the calculation of the notch is given in Equation 7.

$$M \pm 1.57 \times \frac{IQR}{\sqrt{n}}$$ **Equation 7**

where *M*, *IQR* and *n* are the median, interquartile range and number of observations respectively.

The notched box plots of number of submissions for compilation, number of wait durations from 1 to 2 seconds and number of backspace keys for frustration and non-frustration are shown in Figure 3-8, Figure 3-9 and Figure 3-10 respectively. The plots for the rest of the features are not shown as they are either badly skewed or the medians are not significantly different. The plots for number of submissions for compilation, number of wait durations from 1 to 2 seconds and number of backspace keys (with the exception of total number of keys) show that the difference in medians between the frustrated and non-frustrated students are significant at 0.05 level for these features.

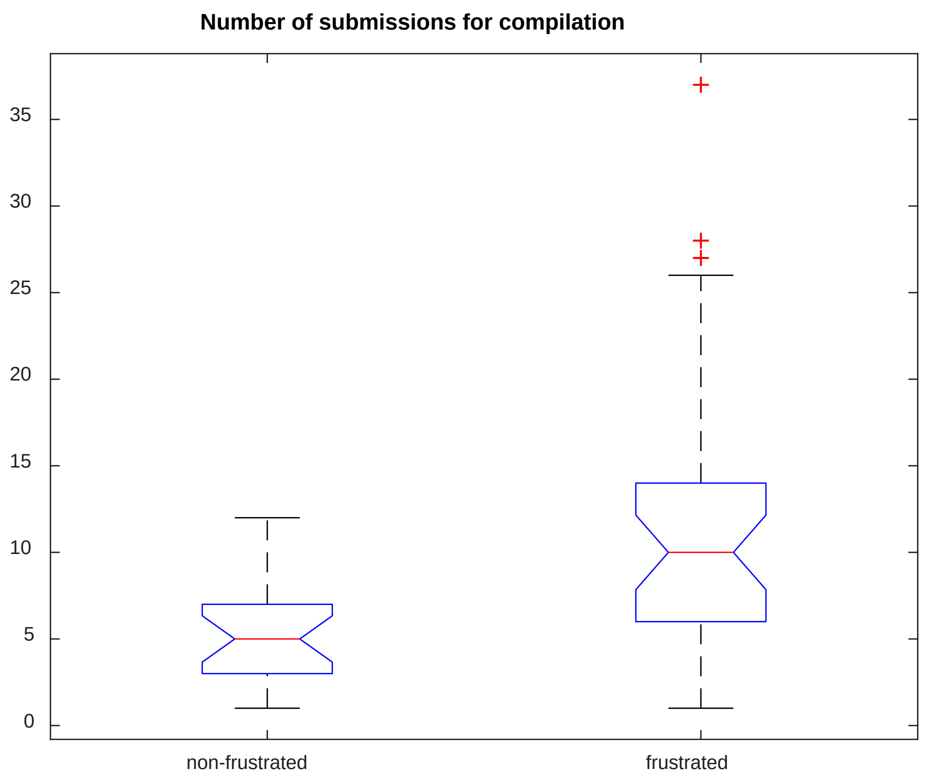


Figure 3-8: Notched box plot of number of submissions for compilation

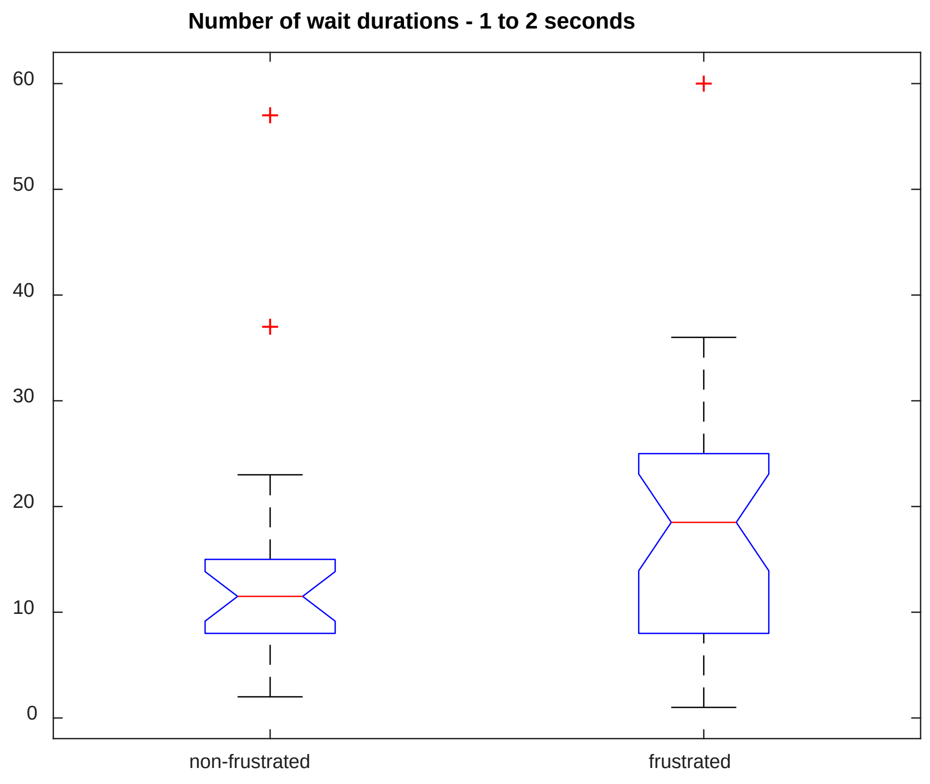


Figure 3-9: Notched box plot of number of wait durations from 1 to 2 seconds

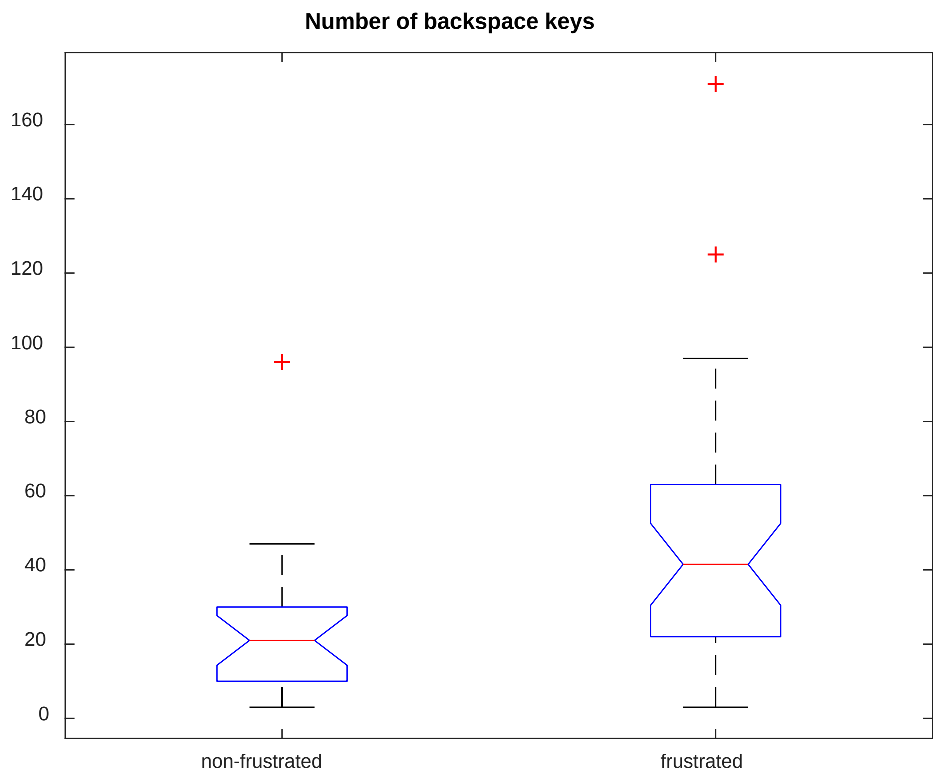


Figure 3-10: Notched box plot for number of backspace keys

The plot for the logarithm of the total number of keys in Figure 3-11 shows that the notches overlapped and thus the difference in median between the 2 groups is not significant at 0.05 level. Thus, both the lasso regularized logistic regression and the notched box plots confirmed that the significant features are the number of submissions for compilation, number of wait durations from 1 to 2 seconds and number of backspace keys. The only exceptions are the total number of keys and the effort.

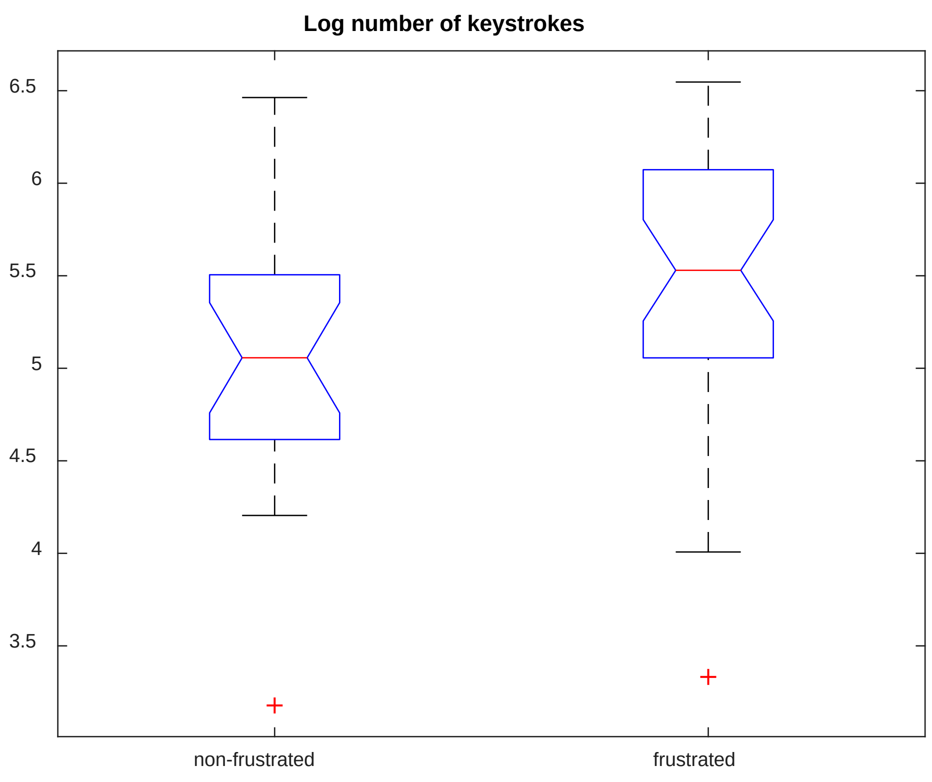


Figure 3-11: Notched box plot for logarithm of number of keystrokes

The accuracy, precision, recall, F-measure and Area Under the Receiver Operating Characteristics (ROC) Curve (AUC) were calculated and shown in Table 3-6. The AUC measures the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance (Fawcett, 2006). The formulas for precision, recall and F-measure are defined in Equation 8, Equation 9 and Equation 10 respectively.

$$precision = \frac{tp}{tp + fp}$$ **Equation 8**

$$recall = \frac{tp}{tp + fn}$$ **Equation 9**

where : true positive, $\boldsymbol{fp}$: false positive and $\boldsymbol{fn}$: false negative.

$$F\ measure = \frac{2\ \times precision\ \times recall}{precision + recall}$$ **Equation 10**

From the results in Table 3-7, the accuracy of the contextual model at 0.583 is slightly better than random chance at 0.5 but with the inclusion of the keystroke features, the accuracy increases to 0.667 (an improvement of 14.4% over the contextual features only model). Although the precision figure (0.625) is relatively low, the high recall performance measure (0.833) indicates that our model is able to identify most incidences of frustration but with a moderate number of non-frustration cases being wrongly classified as frustration.

This would be acceptable for a fail-soft scenario where for example, the repercussions of wrongly labeling students as frustrated (when they are not) is negligible. In any case, it is possible to change the threshold such that we can achieve a higher recall as the expense of a lower precision or a higher precision at the expense of a lower recall. The F-measure and AUC figures further confirm the adequacy of the performance of our model. Thus, the results substantiate the hypothesis that frustration of a student with the programming task on hand can be reliably detected through utilizing the contextual and keystroke features of the student collected within the tutoring system and the addition of keystroke features enhances the detection performance of the contextual features only model.

3.5.2. Study 2

The model was tested using k-fold cross validation methodology with k=5 folds (in each fold, 4 segments were used for training and 1 segment for testing). K-fold cross validation was employed to minimize over-fitting - the issue of the model having an excellent fit to the training data but yet not fitting well to future unseen data.

To evaluate the performance of the proposed model, it is compared against a baseline (Naïve Bayes) model. The Naïve Bayes model consists of only the class attribute (existence of frustration) as the parent node and assumes that all the other feature variables are conditionally independent given the class attribute. Table 3-8 compares the performance of Bayesian Networks against Naïve Bayes models for a 30 seconds time window width. The results show that Bayesian Networks can better discriminate the existence of frustration as compared to Naïve Bayes as both AUC and accuracy of Bayesian Networks are higher than that of Naïve Bayes by 27.12% and 27.78% respectively. This can be attributed to the fact that the constructed Bayesian Networks structure models the dependencies among the feature variables well.

Performance Measures	Models	
	Naive Bayes	Bayesian Network
AUC	0.59	0.75 (+27.12%)
Accuracy	0.54	0.69 (+27.78%)
Sensitivity	0.57	0.73 (+28.07%)

Table 3-8: Performance measures of Naïve Bayes and Bayesian Network models

The classification performance results for the various time window sizes are summarized in Table 3-9 and the Receiver Operator Characteristics (ROC) curves for the various time window sizes are in Figure 3-12. The ROC graph is a 2 dimensional graph in which sensitivity is plotted against (1-specificity) and it depicts relative trade-offs between benefits (true positive) and cost (false positives)(Fawcett, 2006).

Performance Measures	Time Window (in seconds)					
	30	60	90	120	150	180
AUC	0.75	0.81	0.86	0.89	0.89	0.91
Accuracy	0.69	0.71	0.76	0.78	0.79	0.81
Sensitivity	0.73	0.80	0.86	0.88	0.93	0.89
Specificity	0.69	0.69	0.72	0.73	0.69	0.74

Table 3-9: Performance measures for the various time window sizes

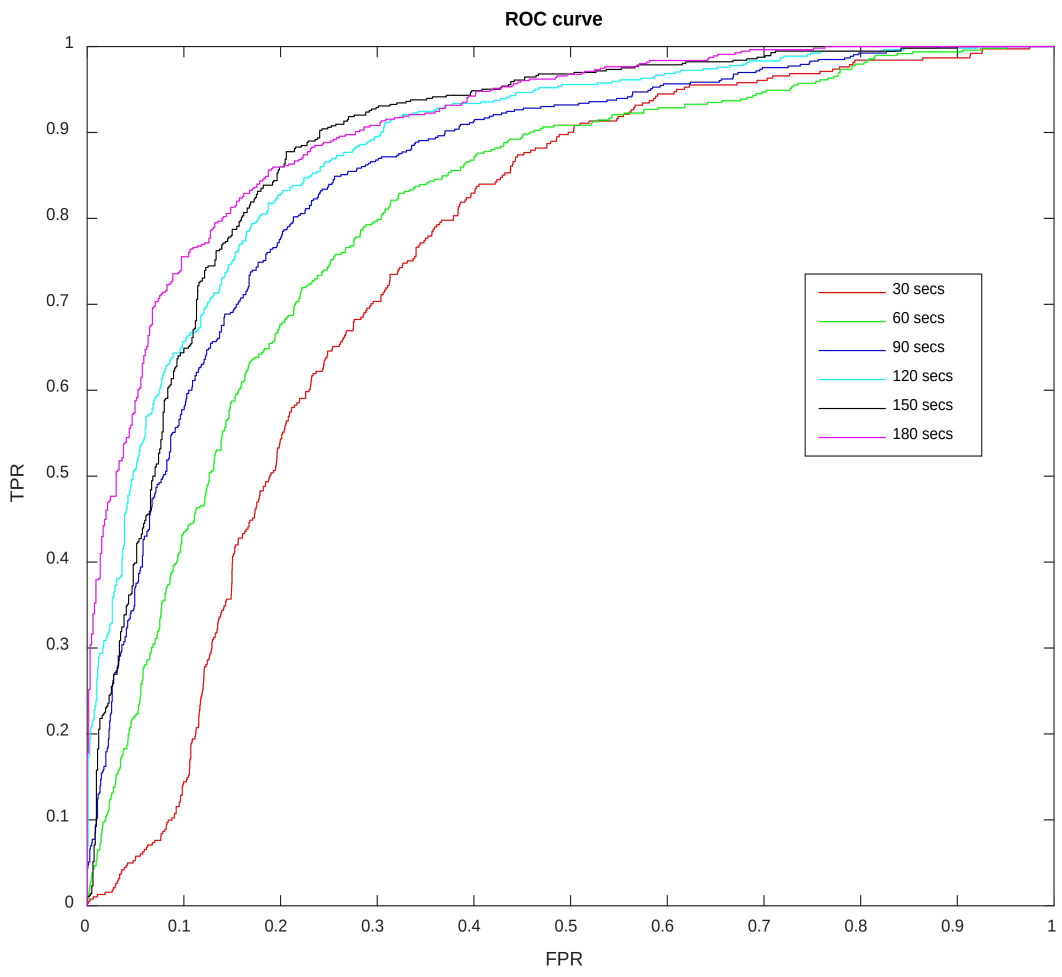


Figure 3-12: ROC curves for the various time window sizes

From Table 3-9, the Area Under the ROC Curve (AUC) and accuracy figures do not differ much across the different time window sizes. The accuracy is defined as the number of correctly identified instances divided by the total number of instances. In situations where the class distributions are highly skewed as they are in our case, accuracy is not a good performance metric for classification. AUC and sensitivity would be more adequate as the performance metric for model evaluation instead. For our model, the best AUC figure and accuracy figure is in the 180 seconds time window. For sensitivity, the 150 seconds time window offers the best performance. Sensitivity measures the true positive rate or the proportion of positives that are correctly identified as such while specificity measures the proportion of negatives that are correctly identified as such. A high sensitivity would mean that most of the instances when students are frustrated are detected by the model but at the expense of a lower specificity. A lower specificity is of a lesser concern as for our fail soft scenario of identifying students who are frustrated and responding with strategies to sustain and motivate them in their learning, the repercussions of wrongly labelling students as frustrated when they are not are negligible.

For the time window of 30 seconds, both the AUC of 0.75 and the sensitivity of 0.73 as well as the ROC characteristics depict reasonable performance for frustration detection. By changing the classifier's threshold, we can move to another point on the ROC with a higher sensitivity at the cost of a lower specificity or vice versa. One concern is that with the shortened time window, there will be more 'holes' – periods with no keystrokes which are eliminated from the data set. Thus, it may be necessary to make up for these periods of non-detection with other sensing modes.

3.6. Summary

For the first study, using the machine learning algorithm of Logistic Regression with Lasso Regularization, I formulated and compared the 2 models – one that uses the contextual features and another that uses both contextual and keystroke features collected from the protocol of a student's interaction within a tutoring system that tutors Java programming. The results indicated that although both models can discriminate between frustration and non-frustration at an adequate accuracy level, augmenting the contextual only model with keystrokes enhances the performance of the model. This study also reveals that the significant features are number of submissions for compilation, effort, number of wait durations from 1 to 2 seconds, number of backspace keys and total number of keys and these correspond to similar keystroke features in a previous study by Vizer et. al. Comparing to other studies that use physiological sensors which can possibly measure in higher fidelity (but which may not be ubiquitous in all environments and require substantial time, expertise and effort to set up), only the contextual and keystroke logs of the students are used for the construction of the model in this study. It is also one of the few limited works that models the state of frustration of novice programmers using keystroke characteristics.

The granularity of detection for study 1 (on a per exercise attempt level) may not be sufficiently fine-grained to initiate pedagogical intervention to tackle frustration at the moment when it occurs though it does offer the benefit of identifying which programming exercise or topic is frustrating to students so that corrective actions e.g. a re-design of the exercise can be undertaken by the instructor to maximize the learning efficacy of the students. A higher granularity or time resolution of detection will be useful for scaffolding the student in his or her learning e.g. with the provision of timely hints when system detects that student is frustrated. This leads on to study 2 which investigates into a finer granularity of affect detection.

In study 2, I proposed the novel use of keystrokes and mouse clicks as sensors and Bayesian Network as the model for naturalistic affect detection in the context of a Java tutoring system. The results showed that keystrokes and mouse clicks characteristics together with the contextual logs of students can be used in a Bayesian Network model to distinguish between instances of frustration and non-frustration with a high AUC of 0.91 and sensitivity of 0.89 for 180 seconds time window width and a reasonably high AUC of 0.75 and sensitivity of 0.73 for 30 seconds time window width. Comparing the proposed Bayesian Network model to the baseline model using Naïve Bayes, the Bayesian Network model achieved a differential of 27.12% over Naïve Bayes. This confirms the classification performance of the proposed model as compared to baseline. In addition, the risk of overfitting the data is mitigated with the use of cross validation.

Establishing the appropriate granularity of affect detection is critical as it informs on the affective state of students on a timely fashion so that appropriate remedial actions can be initiated by the tutoring system when students are frustrated with a learning task. In this study, various detection time window widths were also investigated with the formulation of an overlapped sliding window mechanism.

The elimination of time window with no keystrokes constrains the detection of frustration to only periods where keys are depressed. A possible extension to this study would be to investigate the use of other sensing modes such as facial expressions and eye gaze during the period of time when students are thinking but are not yet ready to attempt the exercises. With the incorporation of other sensing modes, it would be possible to detect frustration at a higher overall availability as compared to using only a single mode as it would be rare for all sensors to be unavailable at the same time. In addition, it may also be possible to better discriminate between incidences of frustration from non-frustration with the use of multiple sensing modes. The use of multiple sensing modes would allow us to triangulate between them to determine instances of frustration and non-frustration, resulting in fewer false instances of false detection. This will be the focus of our investigation in the next chapter.

Chapter 4. Multimodal Techniques for Frustration Detection

4.1. Introduction

Most of the past studies on affect detection have focused on a unimodal approach – use of a single modality (e.g. facial features) when in fact, a multimodal approach - the use of multiple modalities for affect detection would more closely emulate human affect expression. This is further affirmed by Picard who postulated that various bodily responses are activated during an emotional episode (Rosalind W Picard, 2000). Despite the recognition that multimodal affect detection potentially offers higher fidelity over unimodal affect detection, the former is widely advocated but rarely implemented (Jaimes & Sebe, 2007). This has been attributed to the inherent challenges in unimodal affect detection which increases exponentially in multimodal affect detection. In addition, unimodal affect detection also suffers from the missing data issue. For example, facial features are not captured by web camera occasionally due to occlusion illustrating the case where data for a single sensor may not be continuously available. In contrast, a multimodal approach leverages on the available data channels for detection to mitigate the missing data issue. These led to my investigation into whether a multimodal approach can overcome the deficiencies of unimodal affect sensing by increasing both the robustness and accuracy of frustration detection of learners in a naturalistic learning environment.

In the previous chapter, I mentioned that there exists the issue of ‘holes’ in detection when one or more sensing modes are not available e.g. there may be no keystrokes and mouse clicks activity during the time period when the student is thinking on how to tackle a problem. I evaded this problem in the previous chapter by eliminating those time windows when keystrokes are not available but this constrained the detection window i.e. there are periods where no affect detection is possible. By incorporating other sensing modes, the affect detection will not only be continuous but possibly more accurate than with a single mode, especially if the sensors complement each other in the sensing process. Thus in this chapter, I seek to evaluate between the unimodal and multimodal approaches using both feature level and decision level fusion by analysing the classification performance of the fusion of keystrokes, mouse clicks, facial features, head pose and contextual logs. Lastly, to test the generalizability of the model across subjects for the classification models, the student-level nested-cross validation was performed for all the models.

4.2. Background

The sections below detail the background information for the use of the various modes of sensing that will be utilized in this study. A sub-section on multimodal fusion techniques is included.

4.2.1. Facial

A strong link has been established between facial features and affective states (McDaniel et al., 2007). Facial features can also be extracted from web camera facial recordings of participants and web cameras are ubiquitous in most learning environment. This has resulted in many researches which focus on facial features for affect detection.

The Facial Action Coding System (FACS) developed by Ekman and Friesen (P Ekman & Friesen, 1978) is a human-observer-based system to describe subtle changes in facial features. FACS consists of 44 Action Units (AUs) which are a contraction or relaxation of one or more facial muscles. The manual coding of AUs is a tedious affair which requires an hour of coding by personnel trained in FACS coding for every minute of video.

The Computer Expression Recognition Toolbox (CERT) (M. Bartlett, Littlewort, Wu, & Movellan, 2008) is a computer vision tool used to automatically detect AUs as well as head pose and head positions and which eliminates the manual AU coding effort. CERT uses Gabor filters and Support Vector Machines (SVMs) to detect the presence and absence of facial actions. In the study by Littlewort et al. (2011), CERT was evaluated on the Extended Cohn-Kanade Dataset (CK+) with an average accuracy of 90.1% across 18 AUs (Littlewort et al., 2011). In addition, Grafsgaard (Grafsgaard, Wiggins, Boyer, Wiebe, & Lester, 2013) used CERT to predict frustration with r=0.29 from facial AUs for an entire tutoring episode and thus validated that facial features are correlated with the learning affect - frustration.

4.2.2. Keystrokes and mouse clicks

Keystrokes and mouse clicks have traditionally been used in the security domain as a form of biometrics for authentication (M. Brown & Rogers, 1993; P. Dowland et al., 2002). Most of the researches utilize features such as time duration between keystrokes, dwell time (the time a key is held down) and frequency of depressing backspace keys. In the domain of affect detection, keystrokes and mouse clicks are rarely explored, leaving many opportunities for new contributions (Vizer et al., 2009). Vizer et al. (2009) further demonstrated that it is possible to detect induced cognitive and physical stress through continuous monitoring of keystroke interactions. They extracted timing, keystroke and linguistic pattern from free text and compared classification performances of raw versus normalized data.

4.2.3. Head pose

The key benefit of using head pose is that it is less prone to social masking and that explains why it was used in a number of studies for detection of fatigue and attention. In their study, Vural et al. (2008) used a combination of head pose and facial expressions to detect driver's fatigue conditions. Another study by Asteriadis et al. (2009) documented the use of head, eye and hand movements to gauge children's level of interest and attention in the context of reading an electronic document. An accuracy level of 87.7% was achieved for the detection of attention albeit with a small sample size of 20 children (who were rated in the lower quartile for their text decoding ability).

4.2.4. Multimodal Fusion

Multimodal fusion refers to the process of integrating information from multiple input modalities and combining them into a complete command (D’Ulizia, 2009). The techniques of multimodal fusion include both early fusion (feature level fusion) and late fusion (decision level fusion). Feature level fusion consists of concatenating the features of the various modalities, making up 1 large feature vector for the learning stage and only 1 single output is produced. On the other hand, in decision level fusion, the features of the various modalities are not combined and instead fed individually into their individual learning stages. The decisions of the individual subsystems are later combined via specific techniques to produce the final decision. An argument in favour of feature level fusion is that humans express emotion through multiple channels e.g. both face and voice in a complementary and redundant manner and feature level fusion (with the combination of input signals in a joint feature space) emulates the human expression of emotion via multiple channels. In contrast, decision level fusion ignores the information of mutual correlation among the various modalities by combining at the score or decision level. A deficiency of feature level fusion however is that it requires data to be available for all channels as it is not capable of handling missing data (Konar & Chakraborty, 2014).

4.3. Modelling

This section details the modelling of the system for multimodal detection of frustration.

4.3.1. Facial Features

The facial features are captured from web cameras installed on the top of the desktop monitors. The iMotions FACET software (https://imotions.com/software/add-on-modules/attention-tool-facet-module-facial-action-coding-system-facs), a commercial version of the Computer Expression Recognition Tool (CERT) was used for extracting likelihood estimates of 17 Action Units (AUs) from the captured videos. These 17 AUs denote facial muscles movements of the brow, eye lid, nose and lip and details of these 17 AUs are listed in Table 4-1. From these 17 AUs, we derived additional features by calculating the median, maximum and standard deviation of each, making a total feature dimension size of 51. These 51 features are temporally aligned with frustration observations in the 30 seconds time window. As we have established the viability of 30 seconds time window for frustration detection, we will just focus on the use of 30 seconds time window for this study.

AU Number	**FACS Name**
1	Inner Brow Raiser
2	Outer Brow Raiser
4	Brow Lowerer
5	Upper Lid Raiser
6	Cheek Raiser
7	Lid Tightener
9	Nose Wrinkler
10	Upper Lip Raiser
12	Lip Corner Puller
14	Dimpler
15	Lip Corner Depressor
17	Chin Raiser
18	Lip Pucker
20	Lip Stretcher
23	Lip Tightener

24	Lip Pressor
25	Lips Part

Table 4-1: List of Action Units (AUs) and FACS names

4.3.2. Keystrokes, mouse clicks and contextual

Keystrokes and mouse clicks were captured only when the students were working on the exercises. As per the previous studies, the raw keystroke files were processed to derive details such as the keystroke, timestamp, duration of the key and whether modifier keys such as shift or control keys are depressed. The contextual features used is listed in Table 3-1 and the keystrokes and mouse click features is listed inTable 3-5 with a total feature dimension size of 36 (4 from contextual features and 32 from keystrokes and mouse click features).

4.3.3. Head pose

The head pose features were captured from web cameras installed on the top of the desktop monitors. An eye tracking software from xLabs (xlabsgaze.com) was used for extracting raw head pose information such as horizontal and vertical head position, head roll, pitch and yaw from the captured videos. These were further processed to derive the median, standard deviation and maximum positional, pitch, roll and yaw velocities, constituting a total feature dimension size of 11. Table 4-2 shows the details of the raw head pose and the processed head pose information while Figure 4-1 shows how the roll, pitch and yaw angles of head pose rotation are computed. The list of head pose features (with feature dimension size of 11) is listed in Table 4-3.

File type	File format	Description
Raw head pose file	Time stamp, x position, y position, yaw, pitch, roll	x position denotes the horizontal position of the head. y position denotes the vertical position of the head. yaw is the rotational angle around the y axis. pitch is the rotational angle around the x axis. roll is the rotational angle around the z axis.
Processed head pose on server	From time, to time, median pitch, median roll, median yaw, median translation, SD pitch, SD roll, SD yaw, SD translation	From time is the start date and time for the current time window. To time is end date and time for the current time window. Median pitch is the median of pitch for the time window. Median roll is the median of roll for the time window. Median yaw is the median of yaw for the time window. Median translation is the median of horizontal translation for the time window. SD pitch is the standard deviation of pitch for the time window. SD roll is the standard deviation of roll for the time window. SD yaw is the standard deviation of yaw for the time window. SD translation is the standard deviation of horizontal translation for the time window.

Table 4-2: Raw and processed head pose files format

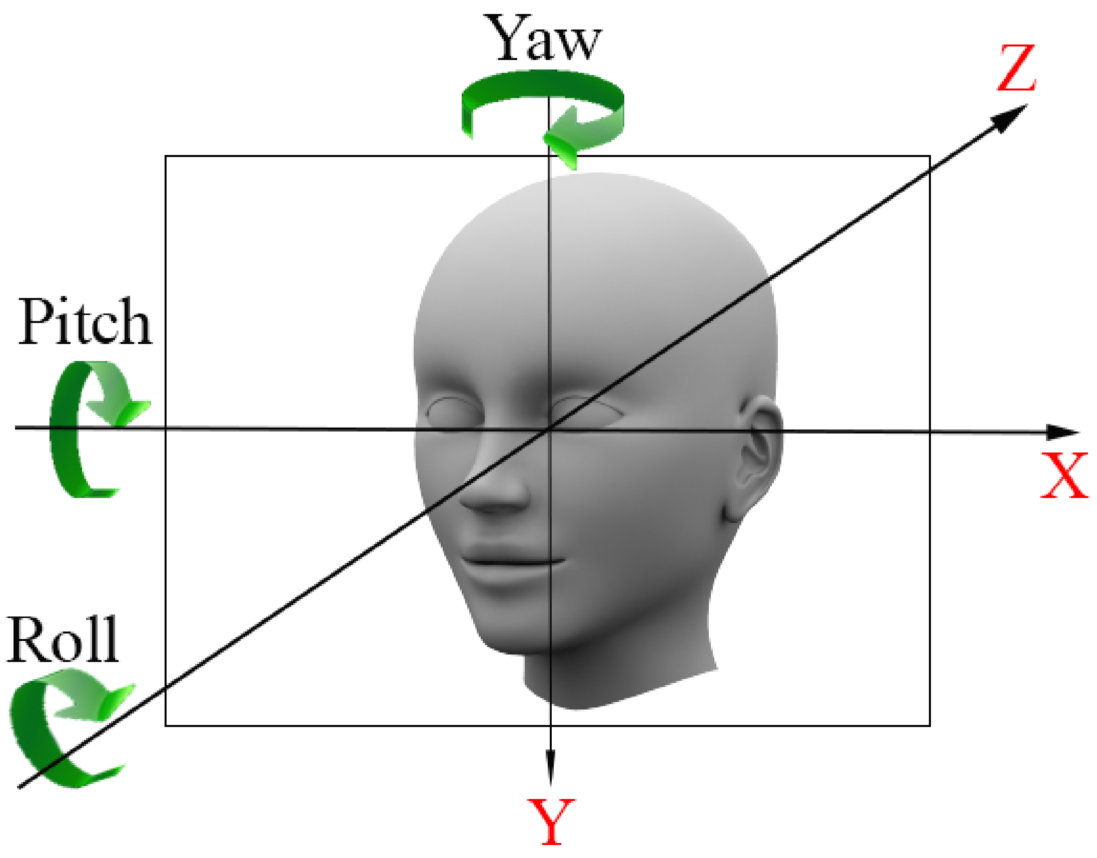


Figure 4-1: Yaw, Pitch and Roll angles of head pose

Head Pose Features	Description
med_horiz_pos_vel	Median horizontal velocity
std_horiz_pos_vel	Standard deviation of horizontal velocity
med_roll_vel	Median roll velocity
max_roll_vel	Maximum roll velocity
std_roll_vel	Standard deviation of roll velocity
med_yaw_vel	Median yaw velocity
max_yaw_vel	Maximum yaw velocity
std_yaw_vel	Standard deviation of yaw velocity
med_pitch_vel	Median pitch velocity
max_pitch_vel	Maximum pitch velocity

std_pitch_vel	Standard deviation of pitch velocity

Table 4-3: List of head pose features

4.3.4. Random Forests

Random Forests (RF), a classifier technique that was first proposed by Breiman (2001), is used as the base classifier in this study. It is an ensemble of bagged decision trees constructed with a different bootstrap sample of the data for each repetition and uses the best subset of predictors which are randomly picked to split a node. A prediction is then generated by an aggregation of votes from all the trees. An alternative to RF is the Support Vector Machine (SVM) but SVM's performance was tested to be below par for this research as compared to RF.

RF works by combining the concepts of bagging with random selection of a sub-set of features to split a node during the construction of a tree. Part of the algorithm of RF requires the use of bagging or bootstrap aggregating where random samples are selected from the training set X and Y (where X : feature vector and Y: class variable) with replacement and used to fit a decision tree repetitively for k times. The algorithm for bagging is given below.

1. Repeat the steps below for k times
   a. Sample, with replacement, n training examples from $\{X, Y\}$ – bootstrapped training set.
   b. Train a decision tree on the sampled or bootstrapped training set in step a.
2. Prediction for test data set made by either averaging predictions of the individual decision trees or a majority vote among the trees.

It turns out that RF is able to achieve comparable classification performance as compared to other classifiers e.g. Support Vector Machine and is robust against overfitting (Breiman, 2001). For this study, the TreeBagger function in Matlab is used for the creation of the RF model. The number of features to sample for the splitting of nodes is set to $\sqrt{m}$, where m is the total number of features and the maximum number of trees is arbitrarily set to 100.

4.3.5. Decision Fusion and Feature Fusion Techniques

Four types of decision fusion techniques are evaluated for combination of the classification results (existence of frustration) from the base classifiers – average vote, majority vote, product vote and Decision Template fusion (Kuncheva, Bezdek, & Duin, 2001). There are a total of 3 base classifiers built using the Random Forest classifier comprising of the facial features base classifier, the keystrokes, mouse click and contextual logs base classifier and the head pose base classifier. The architecture of the Decision Template fusion is shown in Figure 4-2. The benefits of Decision Template fusion are that the computations are simple and it gives a good general overall performance over other fusion techniques.

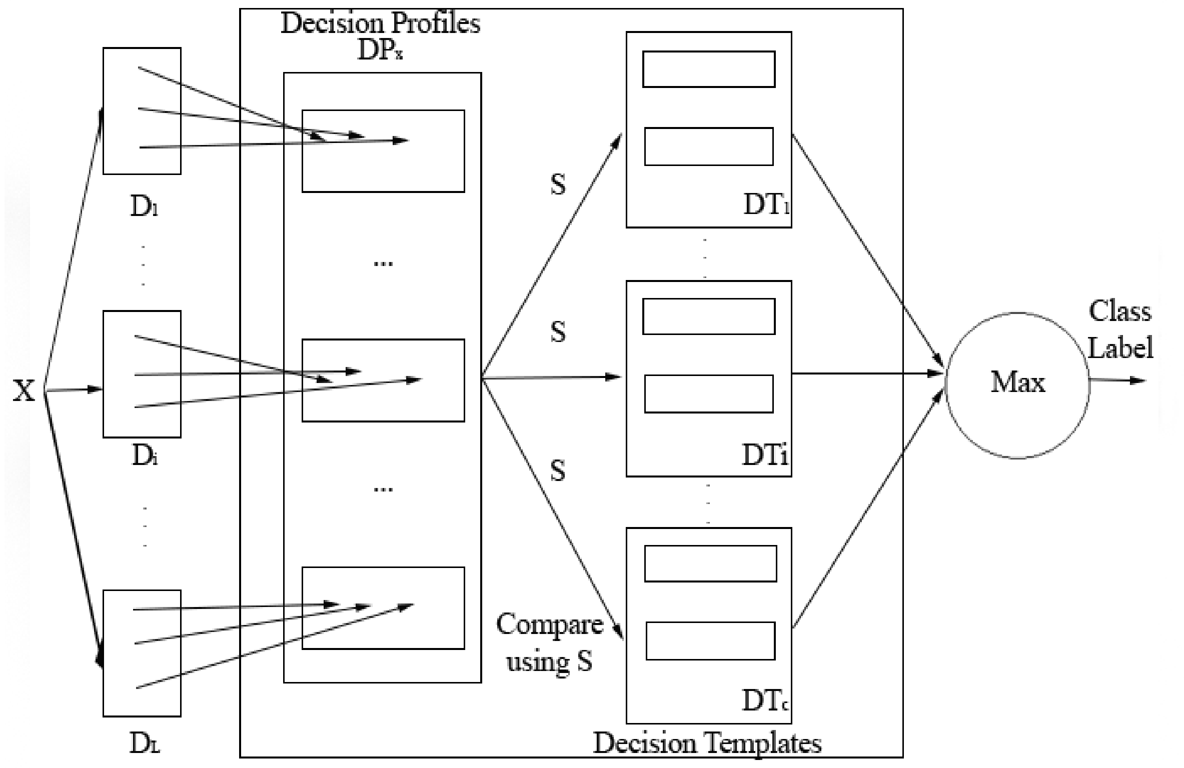


Figure 4-2: Decision Template fusion architecure

Average Vote – This combines the classification results of the base classifiers by averaging the classifiers' probabilities. The predicted class would be the class with the maximum averaged probability as given by

$$\overline{pred} = \arg max_j \sum_{i=1}^{N} \frac{P_{i,j}}{N}$$ **Equation 11**

where $P_{i,j}$ refers to the probability given by classifier i for class j and N is the number of classifiers. $\overline{pred}$ refers to the predicted class.

Majority Vote – This combines the classification results of the base classifiers by selecting the base classifier with the best probability out of all the classifiers' probabilities.

$$\overline{pred} = \arg\max x_j \; max_i \, P_{i,j}$$ **Equation 12**

where $P_{i,j}$ refers to the probability given by classifier i for class j. $\overline{pred}$ refers to the predicted class.

Product Vote – This combines the classification results of the base classifiers by multiplying the classifiers' probabilities. The predicted class would be the class with the maximum product of probabilities as given by

$$\overline{pred} = \arg max_j \prod_{i=1}^{N} P_{i,j}$$ **Equation 13**

where $P_{i,j}$ refers to the probability given by classifier i for class j and N is the number of classifiers. $\overline{pred}$ refers to the predicted class.

Decision Template – The classifier outputs are organized in a matrix as the decision profile.

$$\begin{bmatrix} d_{1,1}(x)\cdots & d_{1,j}(x)\cdots & d_{1,c}(x) \\ \vdots & \vdots & \vdots \\ d_{i,1}(x)\cdots & d_{i,j}(x)\cdots & d_{i,c}(x) \\ \vdots & \vdots & \vdots \\ d_{L,1}(x)\ldots & d_{L,j}(x)\ldots & d_{L,c}(x) \end{bmatrix}$$

The decision template *DT$_i$ (Z)* is the L x C matrix whose (k, s)th element $dt_i(k,s)(Z)$ is computed by

$$dt_i(k,s)(Z) = \frac{\sum_{j=1}^{N} Ind(z_j,i) d_{k,s}(z_j)}{\sum_{j=1}^{N} Ind(z_j,i)},$$ **Equation 14**

$$k = 1..L, s = 1..c$$

The decision template *DT$_i$* for class *i* is the average of the decision profiles of the elements of the training set Z labeled in class *i*.

When a test set $x \in R^n$ is submitted for classification, the similarity measure (μ_d^i) below is evaluated where a higher similarity measure equates to a higher support for that class.

$$\mu_d^i = 1 - \frac{1}{L\,C}\sum_{k=1}^{L}\sum_{s=1}^{C}(dt_i(k,s) - d_{k,s}(x))^2 \qquad \textbf{Equation 15}$$

The Euclidean distance $(dt_i(k,s) - d_{k,s}(x))^2$ is used as the similarity measure in this study.

Feature fusion – The facial, keystrokes, mouse clicks, head pose and contextual logs features are combined into a large feature vector and fed into a Random Forest classifier for classification. Although the performance of feature level fusion or early fusion is generally regarded to be superior to that of decision level fusion as the latter results in loss of information on the mutual correlation between the various sensor modes, the choice of fusion model is still an open research issue (Zeng, Pantic, Roisman, & Huang, 2007a).

4.4. Results and Discussion

4.4.1. K-fold cross validation

The data sets were randomly shuffled. Each of the models were tested using k-fold cross validation methodology with k=10 folds (in each fold, 9 segments were used for training and 1 segment for testing). K-fold cross validation is a common technique employed to minimize over-fitting - the issue of the model having an excellent fit to the training data but yet not fitting well to future unseen data (Hastie et al., 2009). Table 4-4 shows the comparison of the area under the ROC curve (AUC) for the various models. Our data set is heavily skewed with frustration accounting for only 792 out of 9502 time window periods (8.34%) across all the students. Accuracy of the model will not be a useful measure here as a naive classifier that always outputs non-frustration label will have a high accuracy figure of 91.7% but yet, it is not able to identify when the student is frustrated. Thus, the AUC is used as a performance measure to compare between the various models as it is useful for domains with skewed class distribution and unequal classification error costs (Fawcett, 2006).

Model	**AUC**	**% difference over best unimodal**
Facial features	0.850	-
Keystroke and contextual logs	0.749	-11.90%
Head motion	0.756	-11.10%
Average	0.914	7.50%
Product	0.906	6.60%
Majority	0.895	5.30%
Decision Template	0.914	7.50%
Feature Fusion	0.907	6.70%

Table 4-4: AUC measures of unimodal and multimodal models

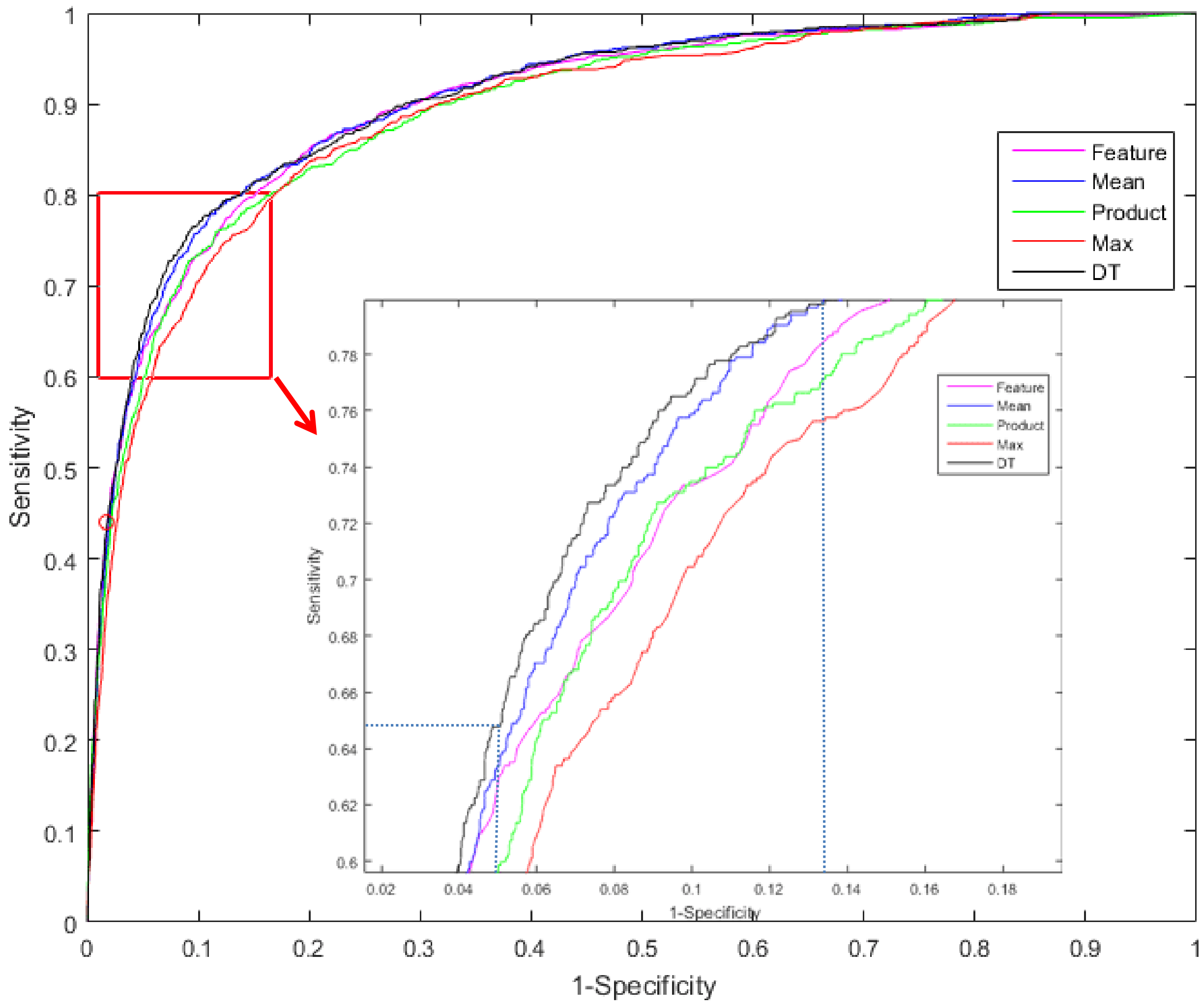


Figure 4-3: ROC curves of the various fusion techniques

As can be observed from Table 4-3, the best unimodal mode is facial features with an AUC of 0.85 and the multimodal model of feature fusion outperforms it by 6.7%. The variance of the various models' AUCs are calculated using the Mann-Witney two-sample statistic and the theory of U-statistics (DeLong, DeLong, & Clarke-Pearson, 1988). A one-tailed t-test is then used to test for whether AUC of one model is significantly greater from the other.

The Mann-Whitney two-sample statistic is shown to be equal to the area under an empirical ROC curve (calculated using the trapezoidal rule). It estimates the probability, θ, that a randomly chosen observation from the population represented by C_2 will be less than or equals to a randomly selected observation from the population represented by C_1. The estimated value of θ, $\hat{\theta}$ is given by

$$\hat{\theta} = \frac{1}{mn} \sum_{j=1}^{n} \sum_{i=1}^{m} \psi(X_i, Y_j), \qquad \textbf{Equation 16}$$

$$E(\hat{\theta}) = \theta = P(Y < X) + \frac{1}{2} P(Y = X) \qquad \textbf{Equation 17}$$

where $X_i, i = 1,2, \dots m$ is the prediction scores for class label 1 and $Y_j, j = 1,2, \dots n$ is the prediction scores for class label 0.

and

$$\psi(X, Y) = \begin{cases} 1 & Y < X \\ 0.5 & Y = X \\ 0 & Y > X \end{cases} \qquad \textbf{Equation 18}$$

By theory of generalized U-statistics by Hoeffding (1948),

$$\xi_{10} = E[\psi(X_i, Y_j)\psi(X_i, Y_k)] - \theta^2, j \neq k;$$

$$\xi_{01} = E[\psi(X_i, Y_j)\psi(X_k, Y_j) - \theta^2, i \neq k; \qquad \textbf{Equation 19}$$

$$\xi_{11} = E[\psi(X_i, Y_j)\psi(X_i, Y_j)] - \theta^2.$$

With that,

$$var(\hat{\theta}) = \frac{(n-1)\xi_{10} + (m-1)\xi_{01}}{mn} + \frac{\xi_{11}}{mn}$$ **Equation 20**

With the variance given in Equation 20, a one-tailed t-test is then performed for the comparison of AUC of facial feature model with that of feature fusion model and it is validated that the discrimination performance of the feature fusion model is different from that of the facial feature model at a significance level of 0.05.

Performing a one-tailed t-test between the AUC of average vote model with that of the feature fusion model, the AUC of the average vote model is not significantly greater than that of the feature fusion model. However, from the ROC curves in Figure 4-3, both the average vote and decision template models give better sensitivity (between sensitivity values of 0.65 to 0.8 as can be seen by the zoomed in figure for portion of the ROC curve within the red enclosed rectangle) for the same specificity values over the feature fusion model, denoting a better discrimination ability over these ranges.

To determine which are the significant features that best contribute to the accuracy of the model, we used a measure of prediction error calculated for every feature used in the RF model. This measure for any feature is the increase in prediction error if the values of that feature are permuted across the out-of-bag observations. It is computed for every tree, then averaged over the entire ensemble and divided by the standard deviation over the entire ensemble. The top 10 features are the head roll, pitch and the facial action units 4 (Brow Lowerer), 5 (Upper Lid Raiser), 7 (Lid Tightener), 10 (Upper Lip Raiser) and 23 (Lip Tightener). The keystroke features are not among the list of top significant features. However, it is still relevant to include keystroke features in the ensemble model as the facial features are unavailable for an average of 17% of the total sessions across all students. This is an already optimistic availability figure as we conducted the trials in a semi-controlled environment where lighting is adequate and students are advised to remain in their seats. In a naturalistic classroom environment where lighting and occlusion issues are more prevalent, the availability of the facial channel will be reduced further. In such cases, the keystrokes can make up for the periods where the facial or head postures (both of which are captured from web cameras) are not available.

4.4.2. Student-level nested cross validation

The data for 22 students with a total of 9502 instances were used for the student-level nested cross validation. By using a student-level nested cross validation, we train our models using the dataset for p or 15 students and use the dataset for the remaining (n – p) or 7 students for testing the models, where n refers to the total number of students and p is the number of students selected for training the model for each iteration. With this technique, we can test whether our models are generalizable across new test subjects.

In the outer loop of the student-level nested cross validation, the data set for a randomly selected 66.7% (15 students) of the students are used as the training set with the data set for the remaining 33.3% (7 students) of the students as the test set. In the inner loop, a further 66.7% of the student's data within the outer training set (44.5% of the whole data set) is then used for feature selection. The feature selection results are then averaged over 10 runs of the inner loop. The classification results for each of the models are averaged over 30 runs of the outer loop to derive the final classification results.

For feature selection, the RELIEF-F technique (Kononenko, 1994) is used to select the top 30, 40, 50 and 60 features. RELIEF-F is used here as it can deal with incomplete and noisy data set. The selected features are then fed into the Random Forest, Logistic Regression and Naïve Bayes classifiers trained for each of the top 30, 40, 50 and 60 features as shown in Figure 4-4.

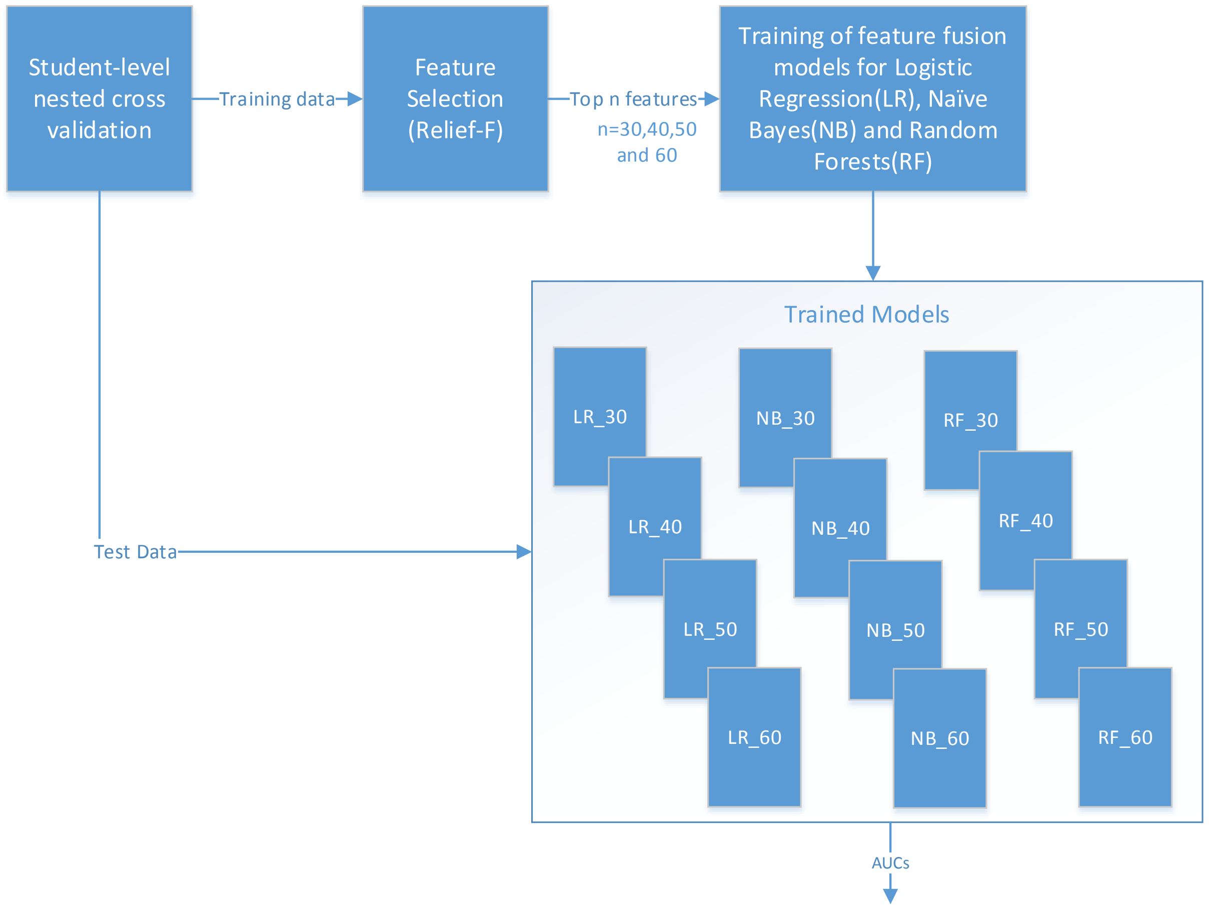

Student-level nested cross validation
Training data
Feature Selection (Relief-F)
Top n features
n=30,40,50 and 60
Training of feature fusion models for Logistic Regression(LR), Naïve Bayes(NB) and Random Forests(RF)
Trained Models
LR_30
LR_40
LR_50
LR_60
NB_30
NB_40
NB_50
NB_60
RF_30
RF_40
RF_50
RF_60
Test Data
AUCs

Figure 4-4: Data pre-processing, features selection and training of features fusion classification models

The AUCs for the unimodal models - facial channel (FC), head pose channel (HPC) and keystrokes, mouse clicks and contextual channel (KMC) using the Random Forest, Logistic Regression and Naive Bayes classifiers are as shown in Table 4-5.

Channels	Classifiers			No. of features
	Random Forest	Logistic Regression	Naive Bayes	
Facial (FC)	0.552	**0.58**	0.555	25
Head Pose (HPC)	0.51	0.542	**0.553**	5
Keystrokes, Mouse clicks	0.539	**0.575**	0.5	20

and Contextual (KMC)	

Table 4-5: AUCs of unimodal models (Random Forests, Logistic Regression and Naive Bayes)

From the results, the facial channel offers the best performance (AUC=0.58) followed by the keystrokes, mouse clicks and contextual logs channel (AUC= 0.575). 25 facial features, 5 head pose features and 20 combined keystrokes, mouse clicks and contextual features are extracted for FC, HPC and KMC respectively using the RELIEF-F feature selection algorithm. The classifiers for each of the 3 channels are better than the random model with AUC=0.5, thus providing evidence that each of the 3 classifiers can discriminate between instances of frustration from non-frustration better than chance. It can also be seen from Table 4-4 that the logistic regression classifier offers the best performance among the 3 classifiers for FC and KMC.

A range of features from 30, 40, 50 and 60 features are extracted using the RELIEF-F algorithm for the feature fusion models. The classification result for the various feature fusion models using the base classifiers of Logistic Regression, Random Forests and Naive Bayes is shown in Table 4-6. The best performing classifier is Logistic Regression for the top 30 selected features.

Random Forest is an ensemble classification technique which should give better performance. However, in this case, our dataset is an imbalanced one with frustration consisting of only 8.34% of the entire dataset. This affects the performance of Random Forest as it is constructed to minimize the overall error rate. This results in the maximizing of the majority class accuracy, resulting in poor accuracy for the minority class (Chen, Liaw, & Breiman, 2004).

Top selected features	AUC (Logistic Regression)	AUC (Random Forests)	AUC (Naive Bayes)
30	**0.636**	0.601	0.6
40	0.621	0.608	0.597
50	0.582	0.603	0.599

60	0.568	0.606	0.6

Table 4-6: AUCs for feature fusion models across the classifiers

The 3 channels feature fusion model combines the features for the FC, HPC and KMC into a large feature vector for classification while the 2 channels feature fusion model combines the features for the FC and HPC into a large feature vector for classification. The AUCs for the 2-channels and 3-channels feature fusion models using Logistic Regression as the base classifier is shown in Table 4-7.

Fusion Models	No. of features	AUC (Logistic Regression)
3 channels feature fusion	30	0.636
2 channels feature fusion	30	0.631
3 channels feature fusion	40	0.621
2 channels feature fusion	40	0.607
3 channels feature fusion	50	0.582
2 channels feature fusion	50	0.58
3 channels feature fusion	60	0.568
2 channels feature fusion	60	0.568

Table 4-7: AUCs for 2-channels and 3-channels feature fusion models

The results show that the 3 channels feature fusion model using the top 30 selected features has the best performance (AUC=0.636) among the fusion models. The AUC of 0.636 of the feature fusion model is 9.7% higher than the best unimodal channel (facial channel) with an AUC of 0.58, verifying that multimodal fusion leads to higher detection accuracy over unimodal model. In general, the feature fusion models perform better than the decision fusion models (for those feature fused models with 50 or lesser features).

Although the AUC of the 2 channels feature fusion model which combines the facial and head pose channels is only marginally lower than that of the 3 channels feature fusion model, it is still relevant to include keystroke, mouse clicks and contextual features in the fusion model as the facial and head pose features are unavailable for 17% of the total sessions across all students. In addition, the keystroke features of backspace latency and keystrokes wait interval are among the top 10 features out of the 30 features selected. Thus, the addition of keystrokes, mouse clicks and contextual channel features does complement the detection using facial and head pose channel features.

4.5. Conclusion

In this study, various multimodal fusion techniques have been investigated, ranging from feature fusion to various decision fusion techniques such as the Decision Template fusion. From the k-fold cross validated results, it has been validated empirically via a 1 tailed t-test that a multimodal fusion model offers higher discrimination ability over the best unimodal model that utilizes facial features in our context of frustration detection of learners in a naturalistic learning environment. Although a 1 tailed t-test reveals that the difference between the AUC of average vote model and feature fusion model is not significant, we can observe from the ROC curves that both the average vote and Decision Template fusion model offers better sensitivity values over the same specificity ranges as compared to the feature fusion model.

The student-level nested cross validation is next used to verify the generalizability of our models across new test subjects. Features selection is performed to select the top 30, 40, 50 and 60 features and then passed into 3 different base classifiers for comparison of the models' performance. From the results, the multimodal model outperforms the best unimodal model by 9.7%, verifying that multimodal fusion leads to higher detection accuracy over unimodal model.

Comparing between the various feature fusion models, the top performing model of a 3 channels feature fusion model using the top 30 selected features and Logistic Regression as base classifier (AUC=0.636) performs only marginally above the 2 channels model. However, for this study, the keystrokes, mouse clicks and contextual logs complement the facial and head pose channel which is unavailable for 17% of the total sessions across all the students. This unavailability figure is already an optimistic figure considering that we instructed the students to stay in their seats during the trial. In a naturalistic classroom environment, the availability of the facial and head pose channel is projected to worsen. From the features that are selected, backspace latency and wait interval are among the top 10 out of the 30 features selected and this confirms the significance of keystrokes in our proposed multimodal affect detection model.

The degradation of the model's performance when it is tested for generalizability across new subjects did not come as a surprise as affect expression and detection is distinctive among individuals. The best performing model's AUC of 0.636 also compares favourably among recent studies relating to educational affect detection. For example, the study by Bosch et al. (2016) achieves an AUC of 0.631 for detection of frustration with the use of down-sampling and synthetic over-sampling techniques to equalize base rates in the training data.

Although the addition of keystrokes, mouse clicks and contextual logs did not significantly improve the performance of our proposed multimodal affect detection model, it is also plausible in my view that other sources of information relating to keystrokes and text that is not investigated in this study e.g. semantics of the input text or keys may further enhance the accuracy of affect detection using keystrokes especially context such as essay writing where the use of different words may reflect the affective states of the student. This is supported by Vizer et al. (2009)'s study which has shown that semantics such as the language diversity, complexity and expressivity may be significant factors for the sensing affective states in domains such as essay writing. Computer programming languages, however follow a fixed structural format and adheres to fixed syntaxes and rules and thus may not be sufficiently diverse for the use of semantic features in affect sensing.

To conclude, this study has addressed a gap in the literature investigating unobtrusive techniques comprising of the combination of facial, keystrokes, mouse clicks and head posture for affect detection.

Chapter 5. Automatic Inference of Engagement

5.1. Introduction

Most would agree that engagement is critical to students' learning and achievement. This correlation between engagement and learning achievement is evidenced by many studies (Astin, 1975, 1993; Kuh, Hu, & Vesper, 2000). Consistently, high levels of motivation and engagement have been linked to higher probabilities of students' successes (Blank & Harwell, 1997; Dev, 1997). In realization of the significance of engagement, an increasing number of higher educational institutions routinely conduct engagement surveys so that they can take immediate rectification actions when the areas of student engagement which require attention are identified. Some examples of such surveys include the UK National Student Survey ("The National Student Survey," 2017), National Survey of Student Engagement (NSSE) (Kuh, 2001) and Australasian Survey of Student Engagement (AUSSE) (Coates, 2010), all of which are nation-wide surveys of college students in United Kingdom, United States and Australia respectively. The studies which linked student engagement to desirable learning outcomes and the regular barometric testing of students' engagement in institutions point undeniably to the value of engagement and its role in influencing educational outcomes.

Engagement just like any other affect is a complex and multi-faceted construct which makes it a challenge to measure it. In the literature, one approach for the measurement of engagement is the behavioral perspective. Within this perspective, student engagement is defined as 'the time and effort students devote to educationally purposeful activities' (Radloff & Coates, 2010) where time on-task or effort is the amount of time that students actually spent on learning (Slavin & Davis, 2006).

Keller (1987) proposed the ARCS model that encapsulates the behavioral perspective of learning motivation. It defines that the four conditions of Attention, Relevance, Confidence and Satisfaction (ARCS) are prerequisites of motivation. The condition of Attention refers to the gaining and sustaining of students' attention in learning. Relevance points not only to the material need for students to know the relevance of what they are learning but possibly their perceived relevance as well. Perceived relevance in turn comes from the feeling of enjoyment that results when people in need of affiliation work in groups. Alternatively, it can be from the enjoyment that people who are high in need of achievement derived when they take responsibility for achieving a moderately challenging task that they set on their own. Differences in confidence influences students' persistence and accomplishment while satisfaction results from the intrinsic and extrinsic rewards from learning. Confidence builds up in situations where students experience some level of success when they exert an adequate amount of effort. Thus, from the above studies, one can conclude that effort, relevance, satisfaction, confidence and attention are important measures of learning engagement.

In previous chapters, I employed the technique of supervised learning for mapping students' actions into classifications of whether students are frustrated or not. The model learned from the data can then be used to predict whether new unseen students are frustrated from their actions. This approach however requires the labeling of the data beforehand by subject matter experts. As this is done manually, the labeling process is tedious and prone to human error. Unsupervised learning, on the other hand, does not require labeled data. The provided data set consists of just the input features without the corresponding labels or classes. Using unsupervised learning, one can either discover groups of similar examples within the data, also known as clustering, or infers a function that approximates the distribution of data also known as density estimation (Bishop, 2007). To circumvent the need for tedious human labeling effort, unsupervised learning is used in this study to discover and group students with similar learning behaviors and also to associate different levels of engagement to these behavioral groups.

There were previous studies (Beal, Mitra, & Cohen, 2007; Shih, Koedinger, & Scheines, 2010) that use unsupervised mining to uncover engagement of students in tutoring systems through the use of only students' action logs within the tutoring systems. The key difference between this study and the previous studies is that this study tracks the engagement of students not in the domain of a mathematics tutoring system (middle school mathematics for the 2 previous studies) but in the domain of a computer programming tutoring system.

Another difference is this study uses not only action logs but also sensor logs such as keystroke logs and head posture logs from web cameras. Although the use of sensors for educational data mining is often associated with issues such as obtrusiveness and difficulty of scaling due to their fragility, high cost and long setup time, the sensors proposed in this study (web cameras and keyboards) are unobtrusive, available at a low cost and easy to set up. With the modeling of students' engagement on a moment by moment basis in a computer programming tutoring system, intervention measures can be initiated automatically by the system when necessary to optimize the learning of students.

5.2. Setup

The data set used in this study comprises of 14 students who were recruited in the year 2015 and 2016. The setup was similar to the one described in Chapter 3 with the exception that the students' head motions and eye gazes were captured. The software that was used to track head motion was developed by xLabs ("xLabs eye, gaze and head tracking via webcam," 2016). It is used in this study for extracting raw head pose information such as horizontal and vertical head position, head roll, pitch and yaw from the web camera. The facial videos of the students were not captured in this study as only one web camera was mounted and it was used by xLabs software for eye and head motion tracking.

The students worked on the exercises in the tutoring software while their eye gaze, keystrokes and head motion were logged. The eye gaze and head motion were captured by the xLabs software for translating into the raw head pose and eye gaze information. A Javascript function was developed to send this raw head pose and eye gaze information for logging to physical files on the web server.

5.3. Modeling

This section details the modeling for the inference of student's task engagement.

5.3.1. Head Pose

The captured head pose information is used for gauging the attention of students. Head poses can be used to approximate the eye gaze direction as proposed by Stiefelhagen and Zhu (2002). The head yaw angle is used to track whether a student is looking at the screen or not. If the head yaw angle is more than ten degrees to the left or right, the student is deemed to be not looking at the screen and probably inattentive. In our setup, the web camera is mounted at an average distance of 60 cm from the student and a length of 10.5 cm to the left or right on the computer monitor is considered to be off screen. This is shown in Figure 5-1. Using the formula in **Equation 21** below, the yaw angle limit is calculated to be 10 degrees or 0.1745 radians.

$$\text{yaw angle} = \tan^{-1}\frac{10.5}{60}$$ **Equation 21**

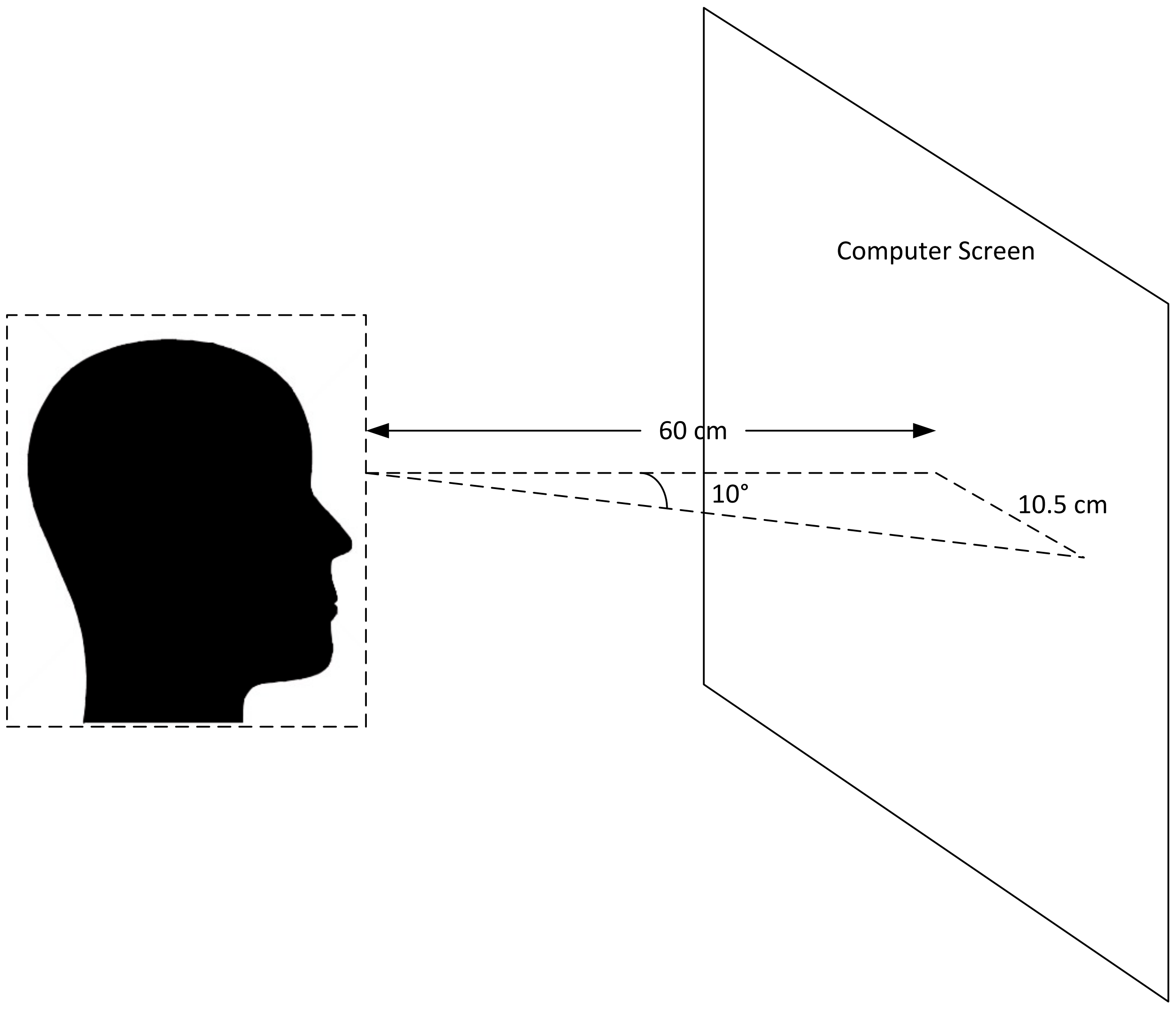


Figure 5-1: Head position and screen setup

The head pitch angle measures the vertical angle of the head motion. Although a student who is looking down and not at the screen is considered to be off screen as well, the head pitch angle is not used (only head yaw angle is used). The reason is that from observations, a number of student participants were looking down at the keyboard while typing. A student who is typing programming codes and looking down at the keyboard while typing should not be considered as being inattentive. As such, using the pitch angle as part of the criteria for determining whether the student is looking off screen and possibly inattentive would not be accurate. Thus, only the yaw angle is used for determining whether the student is looking off screen.

5.3.2. Features

Other than the head motion logs, keystrokes, compilation error details and number of exercises completed are logged as well. The features that are derived from these logs include the number of keystrokes, the maximum keystroke pause duration, the number of compilation attempts, the number of compilation errors and the off screen duration from head motion. These features are then aggregated into a time window slice with a width of 60 seconds. The features are then accumulated into a sequence of feature vectors. These feature vectors are then normalized using the formula $\frac{X-\bar{X}}{\sigma}$ where $\bar{X}$ is the mean and σ is the standard deviation. The number of keystrokes, maximum keystroke pause duration and the number of compilation attempts serve as a surrogate for the effort put in by the student; the number of compilation errors as a surrogate for the obstacles encountered and confidence in tackling these obstacles; the off screen duration as a surrogate for the attention level of the student. The hypothesis is that if the student is confident in resolving the compilation errors, it will be reflected as a decrease in compilation errors in the next time period.

The system will log each instance in which a key is depressed as well as the time at which it is depressed. The pause duration between each key is calculated as the time difference between the depressing of 2 consecutive keys and the maximum keystroke pause duration is the maximum of these pause durations within an entire time window slice.

5.3.3. Hidden Markov Model (HMM)

The temporal-sequential aspect of learning data is an important aspect in the modeling of learning (Shih et al., 2010). The actions of students in a tutoring system are usually time ordered and the sequence of these actions does offer a valuable trove of information into modeling the learning process. For example, comparing two students who solved a programming exercise in the same amount of time, the learning mileage will differ depending on the trajectory of resolving the problem. One student may ask for hints for every failed compilation attempt with no effort put in to decipher the compilation errors while another may solve the problem by carefully deciphering the compilation errors and resolving it without the use of hints. This temporal sequential aspect can be modeled by the Hidden Markov Model (HMM). HMM, which is used for this study, was first popularized by L. Rabiner and Juang (1986). It has since been used in diverse applications such as speech recognition (L. R. Rabiner, 1989), financial stock market prediction (Hassan & Nath, 2005) and human action recognition (Yamato, Ohya, & Ishii, 1992). HMMs consist of stochastic Markov chains based on a set of hidden states whose values cannot be directly observed with the relationship between a hidden state and the actual observations being modeled with a probability distribution. HMMs adhere to the Markov property which states that the state of a model at time t is only dependent on the state of the model at time t-1 and not on other prior states such that $P\left(S_j^{t+1} \middle| S_i^t\right) = P\left(S_j^{t+1} \middle| S_i^t, S_m^{t-1} \dots S_n^0\right)$.

A HMM is described by the tuple $\{S, O, A, B, \pi\}$ where

N : number of hidden states S

$S = \{S_1, S_2, \dots, S_N\}$,

M: number of observation symbols O

$O = \{O_1, O_2, \dots, O_M\}$

$A = \{a_{ij}\}$, where $a_{ij} = P\left(S_j^{t+1}, S_i^t\right); i, j = 1, \dots, N$

$B = \{b_i(k)\}$, where $b_i(k) = P(O_k | S_i)$

The model parameters are valid probabilities that must satisfy the following constraints:

$$\sum_j^N a_{ij} = 1, \sum_k^M b_j(k) = 1$$

The probability of being in state i at time t=0 is given by $\pi = \{\pi_i\}$, where $\pi_{i=}P(S_i^0)$.

In supervised learning, the forward backward algorithm is then used to compute the likelihood that a given training sequence O^s is generated by HMM H.

The forward probability,

$$\alpha_t(i) = \sum_{i}^{N} a_{ij} b_j(O_{t-1}^s)\alpha_{t-1}(i), \qquad (t = 1, \dots, T)$$

Equation 22

The backward probability,

$$\beta_t(j) = \sum_{j}^{N} a_{ij} b_j\,(O_{t+1}^s)\beta_{t+1}(j), \qquad (t = T-1, \dots, 0)$$

Equation 23

The likelihood that the given training sequence O^s is generated by HMM H is

$$P(O^s|H) = \sum\nolimits_{i=1}^{N} \alpha_{T-1}(i)$$

Equation 24

The probability that the training sequence O^s is generated by HMM and that the state at time *t* is *i* is

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O^s|H)}$$

Equation 25

$\gamma_t(i)$ can also be described as the number of transitions from S_i.

The probability that the training sequence O^s is generated by HMM, the state at time *t* is *i* and the state at time *t+1* is *j* is

$$\gamma_{t+1}(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1}^s)\beta_{t+1}(j)}{P(O_s|H)}$$

Equation 26

$\gamma_{t+1}(i,j)$ can also be described as the number of transitions from S_i to S_j.

The model parameters are then calculated as

$$a_{ij} = \frac{\text{expected number of transitions from state i to state j}}{\text{expected number of transitions from state i}}$$

$$a_{ij} = \frac{\sum_{t=0}^{T-2} \gamma_t(i,j)}{\sum_{t=0}^{T-2} \gamma_t(i)}$$

Equation 27

$$\pi_i = \gamma_0(i)$$ **Equation 28**

The Viterbi algorithm (Forney, 1973) can be used to find the most probable path of states given a sequence of observables where $\delta_t(i)$ the highest probability path across all states which end in state i at time t is given by Equation 29. The Viterbi algorithm finds the maximum value of $\delta_t(i)$ iteratively, and then uses a backtracking process to find the sequence of hidden states taken along the path of maximum likelihood.

$$\delta_t(i) = \max_{S_i^1 S_j^2 .. S_k^{t-1}} [P(S_l^t = i, O_i^1 O_j^2 \dots O_k^T \,|H]$$ **Equation 29**

HMMs can be used in unsupervised learning as well. In unsupervised learning, the model parameters are learned by maximizing $\sum_S \log(P(O^s|H))$, the sum of the posterior log-likelihoods of each training sequence O^s using expectation maximization (EM) or the Baum-Welch algorithm (Dempster, Laird, & Rubin, 1977). In this study, the engagement levels of students are modeled as hidden states of the HMM using the HMM toolbox ("Hidden Markov Model (HMM) Toolbox for Matlab," 2016). The HMM is then fitted to the action and sensor logs of students.

5.3.4. HMM with continuous observations

The observations in this study are continuous. Although it is possible to discretize the continuous valued features, it will result in a loss of information. In this model, the model parameter B cannot be represented as a matrix of point probabilities but will have to be represented as a probability distribution function (pdf) over the continuous observation space for each state. The pdf chosen in this study is a Gaussian Mixture Model (GMM) (Rose & Reynolds, 1990)where

$$P(x|z_k\phi) = \sum_{m=1}^{M} P(v_{km}|z_k\phi)P(x|v_{km}\phi) = \sum_{m=1}^{M} P_{km}\mathcal{N}(x|\psi_{km,}\Sigma_{km})$$

where $P_{km} = P(v_{km}|z_k)$ is the probability of m-th mixture under k-th state; μ_{km}, Σ_{km} is mean and covariance of the k-th state, m-mixture respectively.

The Baum-Welch (EM) method can be used to find the model parameters given the supplied data. In the E-step, Equation 25 can be used to calculate $\gamma_t(i)$ and Equation 26 changes to

$$\gamma_t(i,j) = \frac{\alpha_t(i)a_{ij}[\sum_{m=1}^{M} P_{jm}\mathcal{N}(\mathrm{x(t+1)}, \mu_{jm}, \Sigma_{jm)}]\beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_{T-1}(i)}$$ **Equation 30**

Equation 30 is the posterior probability of state i at time t and state j at time t+1 of the model H.

And the joint posterior probability of state i and the mth Gaussian mixture component pdf model of state i as time t is

$$\zeta_t(i,m) = \frac{\sum_{j=1}^{N} \alpha_{t-1(j)} a_{ij} P_{im}\mathcal{N}(\mathrm{x(t)}, \mu_{im}, \Sigma_{im})\beta_t(i)}{\sum_{j=1}^{N} \alpha_{T-1}(j)}$$ **Equation 31**

The re-estimation formulas for the mixture coefficients $\hat{P}_{im}$, mean vector $\hat{\mu}_{im}$ and covariance matrices $\hat{\Sigma}_{im}$ of the model are

$$\hat{P}_{im} = \frac{\sum_{t=0}^{T-1} \zeta_t(i,m)}{\sum_{t=0}^{T-1} \gamma_t(i)}$$ **Equation 32**

$$\hat{\mu}_{im} = \frac{\sum_{t=0}^{T-1} \zeta_t(i,m)x(t)}{\sum_{t=0}^{T-1} \zeta_t(i,m)}$$ **Equation 33**

$$\hat{\Sigma}_{im} = \frac{\sum_{t=0}^{T-1} \zeta_t(i,m)[x(t)-\mu_{im}][x(t)-\mu_{im}]^T}{\sum_{0}^{T-1} \zeta_t(i,m)}$$ **Equation 34**

The re-estimation equations used in the M step for the Continuous Density Hidden Markov Model (CDHMM) are

$$\hat{\pi}_i = \frac{1}{L}\sum_{l=0}^{L-1} \gamma_0^l(i)$$ **Equation 35**

$$\hat{a}_{ij} = \frac{\sum_{l=0}^{L-1}\sum_{t=0}^{T_1-2} \gamma_t^l(i,j)}{\sum_{l=0}^{L-1}\sum_{t=0}^{T_1-2} \gamma_t^l(i)}$$ **Equation 36**

$$\hat{P}_i(\mu_{im}) = \frac{\sum_{l=0}^{L-1}\sum_{t\epsilon x(t)\to\mu_{im}}^{T_1-1} \gamma_t^l(i)}{\sum_{l=0}^{L-1}\sum_{t=0}^{T_1-1} \gamma_t^l(i)}$$ **Equation 37**

where l ranging from 0 to L denotes the data sequences.

5.3.5. Unsupervised Learning

In supervised learning mode of HMM, the hidden states will need to be annotated. The annotated states will serve as the ground truth for inferring the parameters of the model. As such, using HMM in supervised learning mode requires substantial human effort to annotate the data set. In addition, in cases where it is infeasible to infer the hidden states through human observation, supervised learning will not be possible. In consideration of these, the unsupervised learning mode of HMM is used in this study to model the engagement levels of students in this study.

Each student observation is made up of the number of keystrokes, the maximum keystroke pause duration, the number of compilation attempts, the number of compilation errors and the off screen duration.

The unsupervised learning method requires that one determines the number of hidden states that the model should have through model selection. One of the standard model selection technique is the Leave-One-Out Cross-Validation (LOOCV) (Arlot & Celisse, 2010). The algorithm to determine the ideal number of hidden states is listed below.

```
Algorithm NoOfStatesFromCV
begin
    For states = 1 to 5 do
    begin
        For students=1 to total_num_students do
        begin
            testing_data=data_set(students);
            training_data=data_set(~students); //~students refers to students not in
        testing data
            For runs=1 to 10 do
            begin
                hmm_parameters_set = hmm_em_learn(training_data);
                log_likelihood_run(states,students,runs)  =
                hmm_log_probability(test_data);
            end
            log_likelihood_student(states,
            students)=max_over_runs(log_likelihood_run(states,students,:));
        end
```

```
                log_likelihood_state(states) =
                **mean_over_students**(log_likelihood_student(states,:));
        **end**
        **return** log_likelihood_state;
**end**
```

1. **hmm_em_learn** is the procedure used to learn the hmm model parameters using Baum_Welch learning technique.
2. **hmm_log_probability** is the procedure that calculates the log probability of the data given the hmm model.
3. **max_over_runs** is the procedure that calculates the maximum of log likelihoods over the various runs.
4. **mean_over_students** is the procedure that calculates the mean of log likelihood over all the students.

Models ranging from 1 to 5 hidden states were trained over 30 runs using the Baum Welch algorithm with different random initial model parameters. The outer loop performs the LOOCV by leaving a different student's data out of training each time. The left out student's data set was then used as the testing data set for evaluation of the model's log likelihood. The results from the **NoOfRunsFromCV** procedure which gives the Akaike Information Criteria (Akaike, 1998) figures for the various number of states is plotted as shown in Figure 5-2.

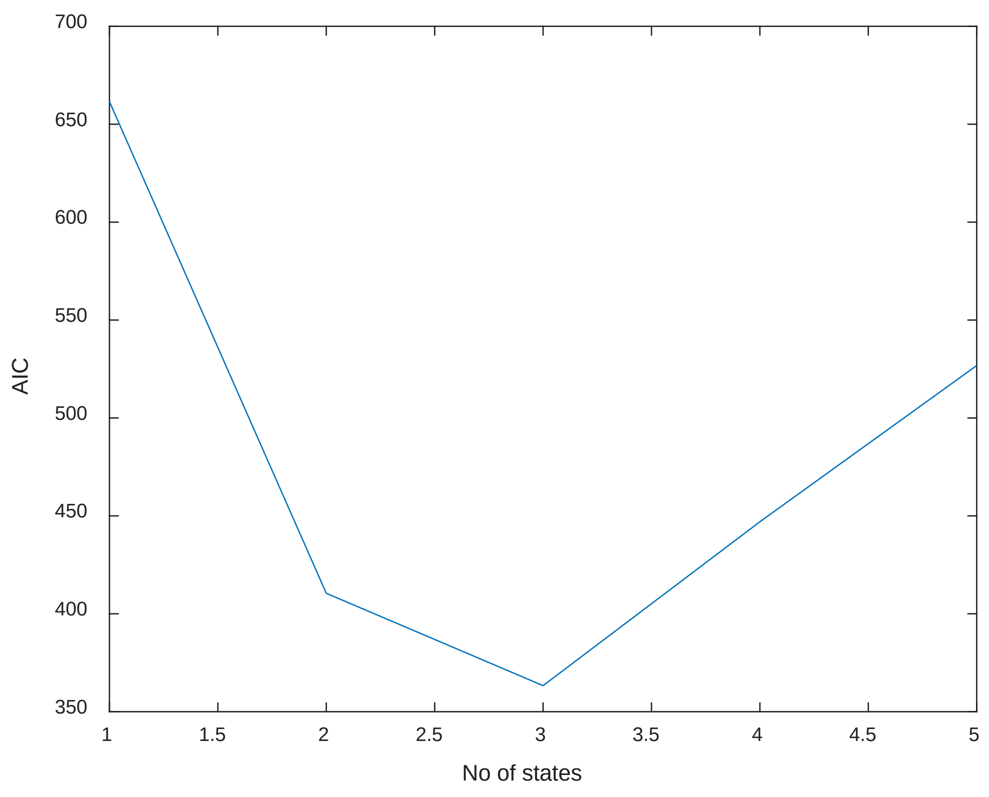


Figure 5-2: AIC versus the number of states for HMM models from leave-one-out cross-validation

From Figure 5-2, one can see that the minimum Akaike Information Criteria (AIC) value is at the number of states value of 3. Thus, the value of 3 is selected as the number of states for our HMM model.

5.4. Results and Discussion

The Gaussian mixture weights and means of the learned model's features for the 2 mixtures are shown in Table 5-1, Table 5-2 and Table 5-3 respectively.

Mixture Weights	**Hidden States**		
	State 1	**State 2**	**State 3**
Mixture 1 weights	0.48	0.91	0.68
Mixture 2 weights	0.52	0.09	0.32

Table 5-1: Learned Gaussian Mixture Weights

Gaussian Mixture 1 (GM1)		**Hidden States**		
		State 1	**State 2**	**State 3**
Mean	No. of compilations	2.81	0.00	1.75
	No. of keystrokes	22.08	0.04	0.17
	Max keystroke pause duration	11.50	60.00	59.95
	No. of errors	0.87	0.00	0.78
	Off screen duration (in milliseconds)	270.66	291.55	2967.38

Table 5-2: Mean values of features for Gaussian Mixture 1

Gaussian Mixture 2 (GM2)		**Hidden States**		
		State 1	**State 2**	**State 3**
Mean	No. of compilations	0.00	3.13	0.88
	No. of keystrokes	41.75	19.94	22.36
	Max keystroke pause duration	9.71	19.72	19.05
	No. of errors	0.00	9.84	0.38
	Off screen duration (in milliseconds)	299.96	198.67	19246.76

Table 5-3: Mean values of features for Gaussian Mixture 2

Combined means from GM1 and GM2		**Hidden States**		
		State 1	**State 2**	**State 3**
Mean	No. of compilations	1.35	0.28	1.47
	No. of keystrokes	32.31	1.83	7.27
	Max keystroke pause duration	10.57	56.37	46.86
	No. of errors	0.42	0.89	0.65
	Off screen duration (in milliseconds)	285.90	283.19	8176.78

Table 5-4: Combined means values of features from GM1 and GM2

From Table 5-4, as compared to other states, State 1 is characterized by a higher number of compilations, high number of keystrokes and a low off screen duration. This suggests that students in State 1 are engaged as they are actively trying to compile and solve the programming exercises (high effort as evidenced by the high number of compilations and keystrokes) and are maintaining a high level of attention on the screen (as evidenced by the low off screen duration). Students in State 2 are characterized by close to zero number of compilations, high maximum keystroke pause duration and very low number of keystrokes but yet low off screen duration. It can thus be inferred that students in State 2 are just starting out working on the exercises. It could also be probable that students in State 2 are clueless and do not know how to start on the exercises if they have been working on the exercises for a while. State 3 is characterized by a low number of keystrokes, high maximum keystroke pause duration, some compilations and high off screen duration. It seems that there is students in State 3 are disengaged and not actively working on the exercises as evidenced by their high off screen duration (low attention), low number of keystrokes and high maximum wait duration (low effort).

Transition Matrix	State1	State 2	State 3
State 1	0.72	0.12	0.16
State 2	0.23	0.68	0.09
State 3	0.40	0.29	0.31

Table 5-5: Probabilities of state transitions

The state transition matrix is shown in Table 5-5. From the state transition matrix, students in State 1 (the engaged state) and 2 (the starting out state) are more likely to persist in their current states than to transit to other states. This can be seen from the higher probabilities of persisting in the original state (e.g. students in States 1 and 2 will tend to persist in States 1 and 2 with a probability of 0.72 and 0.68 respectively). This suggests that a student who is engaged will likely stay engaged. A student who is in State 3 (the disengaged state) is slightly more likely to transit to State 1 (with a probability of 0.40) than to transit to State 2 (with a probability of 0.29). The student in State 3 is also less likely to persist in the disengaged state as seen from the probability of 0.31. The high probability of a student in the starting out state persisting in the same state for a substantial duration of time as well as the one-third chance that a disengaged student will stay disengaged suggest opportunities for pedagogical interventions to shift the student into the more desirable engaged state.

Student identifier	**No. of exercises completed**	**Longest disengagement duration (in periods of 60 seconds)**
1	2	**14**
2	2	**12**
3	3	1
4	1	**12**
5	2	**19**
6	2	**6**
7	5	2
8	7	3
9	4	2
10	4	2
11	2	2
12	1	3
13	2	**9**
14	4	4

Table 5-6: Number of exercises completed and disengagement duration by students

Using the Viterbi algorithm, the temporal sequences of engagement for each individual student was derived from the HMM model. Table 5-6 shows the number of exercises completed and the longest disengagement (hidden state 2) duration for each student with disengagement duration of values 5 or more shown in bold. The longest disengagement duration was derived from the maximum continuous duration of disengagement from the temporal engagement sequences output by the HMM model. To illustrate, for the sample engagement sequence of "122222312213", the longest continuous disengagement duration will be 5. From Table 5-6, students who completed 2 or less exercises experienced longer disengagement episodes as compared to students who completed 3 or more exercises. These long disengagement episodes will likely lead to the student becoming bored and eventually quitting learning altogether. If the tutoring system can intervene and provide support to "unstick" the student analogous to a human tutor providing support and hints to a student who is stuck, the student will likely be re-engaged and gain confidence in continuing to tackle and solve the remaining exercises (Vygotsky, 1978).

5.5. Conclusion

In this study, unsupervised learning using HMM was employed to infer the engagement states of students from both students' actions within the tutoring system as well as from sensors such as web camera (for head postures) and keyboard (for keystrokes). As engagement is critical to students' learning and achievement, the inference of the engagement states allow us to intervene at opportune moments to sustain the students' engagement. The use of HMM allowed for the time sequenced students' responses comprising of their actions and head postures to be translated into hidden engagement states and their transition probabilities.

The results showed that the engagement states of students can be clustered into 3 distinct states – the starting out or disengaged state, the engaged but possibly slipping state and the engaged and confident state. In addition, from the transition matrix, students in the 3 states experience inertia in transiting to other states with the students in the disengaged state having the greatest inertia. The students who did not complete as many exercises also experienced the longest continuous period of disengagement. These suggest the potential for the use of learning interventions such as hints or scaffolding to shift the students into a more engaged state. This hypothesis will however require further experiments to validate and is not covered in this thesis.

Chapter 6. Architecture of Affective Programming Tutoring System

6.1. Introduction

This chapter presents the architectural design and implementation of the Affective Programming Tutoring System (APTS). Augmenting ITS with the ability to infer the affect of students is the first step in the construction of an ATS. However, the next challenge is how do we then design the ATS such that it can respond appropriately to the detected emotions of the students to increase engagement, task persistence and sustain their motivation to learn. The goal is to regulate or ease the negative emotions such as frustration so that it does not degenerate into hopelessness. It would be ideal if a recipe that lists the pedagogic strategies to be applied for every tutoring situation that may arise in an ATS exists. However, a search through the literature reveals that there is a lack of research addressing how ATS or its constituents should adapt to the detected students' emotions. Furthermore, in spite of the critical role that affect plays in learning, few ATSs are actually implemented. Thus, I will address these gaps and discuss within this chapter, the proposed architecture of the APTS, the incorporation of some pedagogic considerations into its design as well as the design of a context sensitive hint module. I will first review some of the background literature on the design of the tutoring response in some ATSs and the design of a hints module for a programming tutor before proceeding to discuss the architectural design of APTS and the design of the hint module in APTS.

6.2. Background

In psychology, coping refers to efforts to master, reduce or tolerate the demands created by stress (Weiten, Dunn, & Hammer, 2011). Weiten et al. (2011) categorized the coping strategies into 3 main coping strategies – appraisal-focused, emotional-focused and problem-focused.

The appraisal-focused coping strategy involves changing the way one thinks about the problem. One can possibly dispute or challenge one’s irrational assumptions. The key is to replace the negative thinking with rational analyses, potentially reducing stressful situations to less threatening ones. Another technique is to de-stress with humour. Humour buffers the effect of stress, allowing people to bounce back from negative situations faster and also helps to promote social interactions and support.

The emotional focused strategy involves releasing one’s pent up emotions through expression of the emotions. The techniques in emotional focused strategy include managing hostility towards others and being more forgiving and the use of relaxation techniques such as mediation.

Lastly, the problem-focused strategy entails having a systematic problem resolution process, seeking help and acquiring new skills for dealing with the problem e.g. time management skills. By exercising self-control through techniques such as operant conditioning (which involves use of rewards and punishments), one can also better cope with the stress. A number of ATSs in the literature respond to detected affect with tutoring actions that can be broadly categorized into these 3 coping strategies and are described in the paragraphs below.

As discussed in Chapter 2, AutoTutor is an ITS that helps students learn Newtonian Physics and computer literacy topics. Affective AutoTutor is one of the few ATSs that detects and responds to students’ affective states (S. K. D'Mello & Graesser, 2010). The tutoring actions are then derived from a set of production rules that dynamically assessed the cognitive and affective states of students to address negative emotions such as frustration and boredom. The tutoring responses of Affective AutoTutor were designed based on the attribution theory of emotions (Weiner, 1972) and the theory of cognitive disequilibrium (see Chapter 2). The attribution theory attributes an individual’s achievement to ability, effort, task difficulty and luck. Affective AutoTutor evaluates the appropriate tutoring response based on the inferred emotion and the student’s attributes (such as ability). The responses inferred from a set of production rules include short feedback, affective statement, dialog sentence, emotional display on embodied agent and modulated audio response of embodied agent. Short feedbacks are responses to the student's action e.g. "Good Job" or "Well Done". Affective statements include empathetic and motivational responses that were formulated with the aim of attributing the source of the emotion or the blame to itself. To surmise, the tutoring responses of Affective AutoTutor are predominantly emotional-focused and appraisal-focused.

Wayang is an ATS proposed by B. Woolf et al. (2009) that tutors students in mathematical concepts. It responds to students' negative emotions (e.g. frustration and boredom) by employing the use of unsupervised offline machine learning algorithm to learn optimal policies from students' data. The tutor's responses include voice or gesture change of embodied agent or learning companion, empathetic responses, graphs and hints, attribution of failure to external sources and scenario change. These responses are largely similar to Affective AutoTutor's with the exception of graphs and hints which are problem-focused responses. Quantitative evaluation of Wayang by B. P. Woolf et al. (2010) on 108 ninth- and tenth-grade students demonstrated that although learning companions did not affect students' learning directly, it did result in the learning companion group spending more time on hinted problems than the non-learning companion group.

Easy with Eve is an ATS that was built by Massey University, New Zealand (S. Alexander et al., 2006). It uses a case-based tutoring strategy for the mapping of students' affective states and interaction histories into tutoring actions. The tutoring strategy module outputs a set of tutoring actions, each with weighted scores and one of the actions out of the list will be selected randomly (with the inclusion of the weighted score of each as a criteria for the random selection). The tutoring actions include a set of agent animation videos and hints which can be categorized into emotional-focused and problem-focused responses. Both pre-test and post-tests and questionnaire were used to evaluate the effectiveness of the ATS on a group of 59 students (S. T. V. Alexander, 2008). The results showed no significant effect of any improvement on the learning of the students which can be attributed to the presence of the affective agent. The results of the questionnaire further revealed that the students did not rate the version of the tutoring system with the animated affective agent to be more enjoyable than the version with only a text based agent. The author attributed this to the deficiency in the facial expression detection module which resulted in the animated agent not being able to respond empathetically in many interactions with the students, thus reducing the enjoyment factor.

Both Wayang and Easy with Eve used hints to address the students' knowledge gaps and alleviate the negative emotions of students. Hinting is a form of support to help novices learn skills or concepts involved in that task (Aleven, Stahl, Schworm, Fischer, & Wallace, 2003). Hints provide information that elaborates on the error and the relevant conceptual knowledge that addresses the gap in knowledge. In fact, hinting is a tactic that is frequently used by human tutors to assist students in comprehending domain concepts and accomplishing learning tasks. When students are frustrated, it is crucial to respond to them adequately and timely with the provision of hints. It is imperative that the provided hints are both relevant and contextual in order to aid the students in overcoming the hurdle or resolving their misconceptions. Without rendering appropriate assistance, students will be stuck in the Zone of Proximal Development (ZPD) which is described as the zone of learning outside of students' capability if they are left on their own but yet would be attainable with additional assistance from their peers or tutors (Vygotsky, 1978). ZPD is typically characterized with feelings of frustration for a prolonged period. This frustration will degenerate into hopelessness after some time and many will give up on the learning eventually if no appropriate help is received.

Both Wayang and Easy with Eve tutor students in the domain of mathematics which is different from our domain of computer programming. Intrinsically, computer programming requires the student to interpret and resolve syntax and logic errors which is different from the solving of mathematical exercises. For generation of hints, no studies can be found on the design of the hint module in Wayang while in Easy with Eve, the student's response is mapped into concept scores and the hint that relates to the highest scored concept will be dispensed to the student (S. T. V. Alexander, 2008). The design of hints for a programming tutor is more complex in relation to that for a mathematics tutor owing to the large permutations of programming constructs in computer programming.

In the teaching of computer programming, students usually key in their programming codes into an Integrated Development Environment (IDE). Modern IDEs offer benefits over a text editor such as code completion, easy navigation, class explorers and even source code formatting and version control. However, in spite of the conveniences offered by IDEs, compilation messages and errors are still notoriously cryptic to novice programmers (Brandt, Guo, Lewenstein, Dontcheva, & Klemmer, 2009). On many occasions, the students would still require additional help from teachers or more experienced peers to interpret and explain the exact meanings and causes of these compilation messages. This necessitates the need for an automatic means of generating easy to understand feedback and help messages to students for interpreting the exact source of errors in their programs.

The primary purpose of hinting is to correct errors or misconceptions that are made by students. It is thus critical that the provided hints be sensitive to the context of the errors or misconceptions for the prompting of appropriate remedial actions to be taken by students. Inappropriate hints on the other hand, will confuse or frustrate the students. A detailed analysis of human tutoring transcripts and interviews with tutors by Hume, Michael, Rovick, and Evens (1996) revealed that hinting is frequently employed by human tutors in their tutoring sessions. The automatic generation of hints by computerized tutor however is not an easy task as the solution space is likely large and the hints generated must be matched to the students' intentions. Invariably, even for a simple programming problem, there are multiple solution sets and infinite permutations of intermediate program states. A brute force solution to generating hints for the infinite permutations of possible states would be to generate hints for each of the state. This, though not impossible, would be too onerous and resource intensive.

There are a variety of possible approaches to the automatic generation of hints. Constraints Based Modeling (CBM) is an approach proposed by (Ohlsson, 1992). It is predicated on domain principles that every good or correct solution has to adhere to. Le and Menzel (2009) further extended this with the concept of weighted constraints. The weighted constraints can be violated and each violation generates diagnostic information that is then communicated to the students via feedback messages. The incorporation of weights in the constraints was used to resolve conflicting error explanations. The LISP tutor is another variant that uses extensive production rules that are executed in sequence (John R. Anderson & Skwarecki, 1986) to second-guess the student's intentions based on their previous actions. This process of simulating the list of sequences of future steps and then matching the observed next steps with the simulated set of next steps for the selection of appropriate instruction is also referred to as model tracing. Rivers and Koedinger (2013) use a transformation approach where students' programs are canonicalized and then converted into an Abstract Syntax Tree (AST). Canonicalization is a process of simplifying, anonymizing and ordering the program's syntax without changing its semantics. The students' programs are then subsequently parsed into AST and compared to a dataset of correct student solutions – a data driven approach.

Given the above background on hint generation, I will discuss in detail my proposed hint generation technique and the reason for its adoption in the section on hint generation design below.

6.3. Overall Architectural Design

Figure 6-1 shows the overall architecture of APTS. APTS consists of the sensor subsystem, student subsystem and the tutoring subsystem. The entire system is developed using Enterprise Java technologies for the back end and uses Java Server Pages, JQuery and JavaScript for the front end. It is hosted on a Tomcat 7.1 web server. Although the tutoring system currently only supports tutoring in Java programming language, it can be adapted for tutoring in other programming languages. In fact, work is currently underway to adapt it for Python programming tutoring. The proposed architecture in which each component within the tutoring system is designed to be loosely coupled also ensures that the tutoring system is generalizable and can be adapted for use in other tutoring domains as well. The sub-sections below detail the function of the various subsystems within the tutoring system.

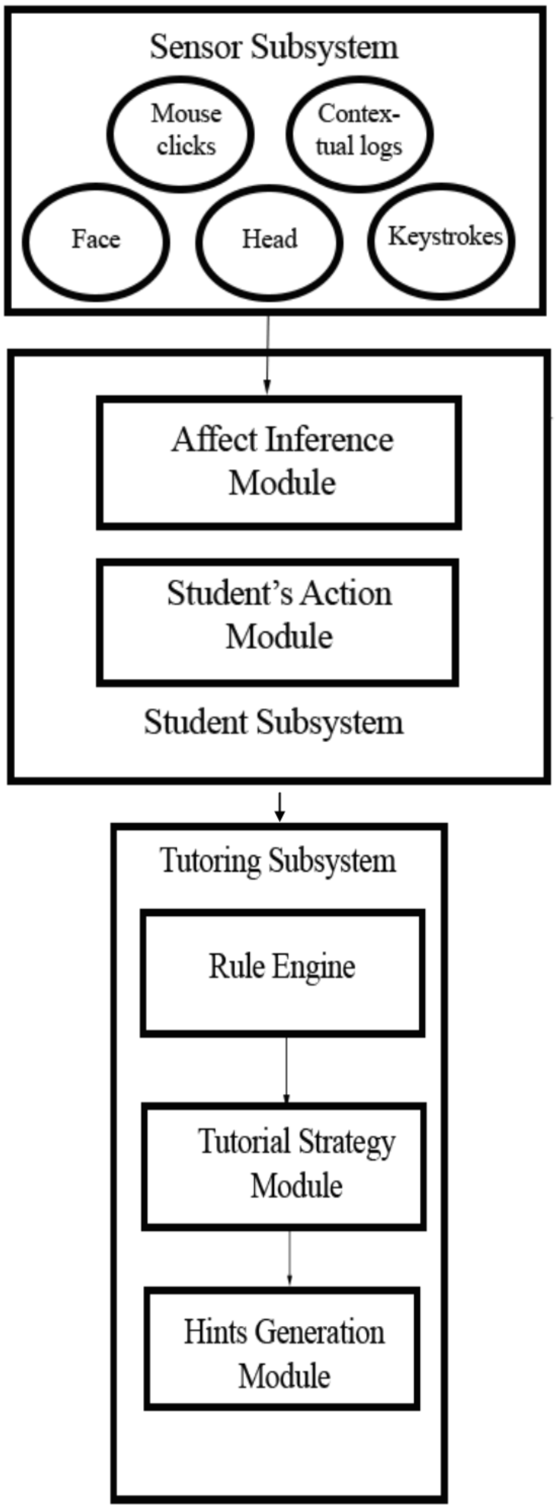


Figure 6-1: Architecture of ATPS

6.3.1. Sensor Subsystem

This subsystem acquires and processes the raw data from the various sensors e.g. web camera for both facial cues and head pose. The facial expressions (AUs' intensity values) and head pose (pitch, yaw and roll angles) are captured by the iMotions FACET software. The keystrokes and mouse clicks are captured using JavaScript at the client PC and these will be transmitted to the server at a configurable interval (currently set to 10 seconds). All the sensor logs will be consolidated and processed at the server end to derive the list of features as detailed in the earlier chapters. The processed features will be passed to the Affect Inference module for inference of the student's affective states.

6.3.2. Student Subsystem

This subsystem consists of both the Affect Inference and the Student's Action modules. The Affect Inference module will be in charge of inferring the affect of the student using an ensemble of machine learning algorithms as described in Chapter 4. From the consolidated features list that is passed in by the Sensor subsystem, the Affect Inference module will then output the inferred affect of the student. The inferred affect will be passed to the Tutorial Strategy module for formulation of an appropriate tutorial strategy that regulates the inferred affect.

The design of the Affect Inference module accounts for the unavailability of any of the sensor modes. A Decision Template fusion ensemble model is used to fuse the outputs from the Random Forest base classifiers of each of the sensor modes. In the event that data from one or more sensor mode is not available at any one time window, the remaining data from the other sensor modes would be fed into the ensemble for the inference of affect. This ensures that the affect inference is continuous albeit at a degraded accuracy level. The inferred affect will then be passed to the Tutoring subsystem.

The Student's Action module logs the history of student's interactions with the tutoring system. The details of the interactions that are logged includes the duration of time spent on an exercise, the topic and exercise the student is working on, the date and time of compilation, the compilation error details and the number of lines of errors for each compilation attempt. Other details that are logged include the details on the hints provided, the date and time the hint is provided and the number of hints provided to each student. The interaction logs are in turn passed as input to the Affect Inference module for inference of the student's affect.

6.3.3. Tutoring Subsystem

The inferred affect of the student obtained from the Affect Inference module and the history of interactions of the student obtained from the Student's Action module are passed on to the Tutorial Strategy module within the Tutoring Subsystem. An instance of the Drools Rule Engine ("Drools - Business Rules Management System," 2017) is embedded within the Tutorial Strategy module. The tutoring strategy rules are encoded within the rule engine and are shown in Figure 6-2. The rules will be detailed in the section on the design of the hint generation module.

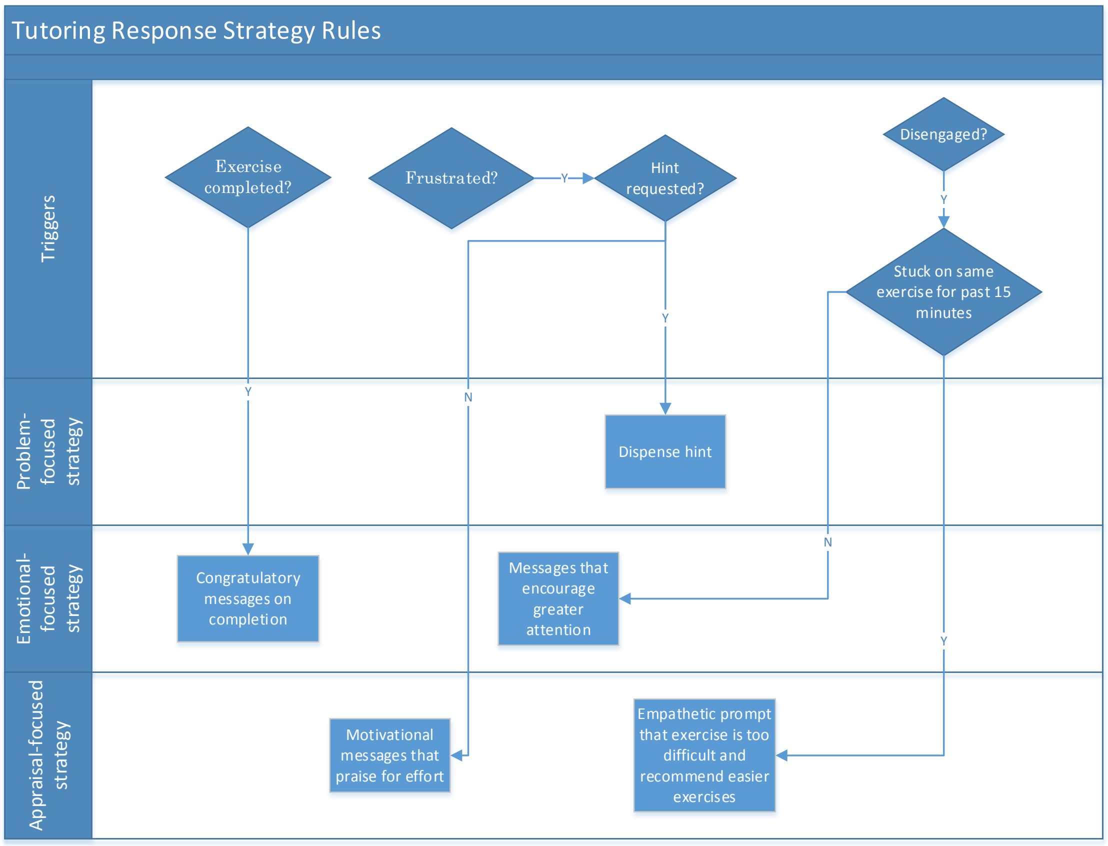

Tutoring Response Strategy Rules
Triggers
Problem-focused strategy
Emotional-focused strategy
Appraisal-focused strategy
Exercise completed?
Frustrated?
Hint requested?
Disengaged?
Stuck on same exercise for past 15 minutes
Y
N
Dispense hint
Congratulatory messages on completion
Messages that encourage greater attention
Motivational messages that praise for effort
Empathetic prompt that exercise is too difficult and recommend easier exercises

Figure 6-2: Tutoring Response Strategy Rules of Tutoring System

A consideration in our design is that it is difficult to establish individual student's ZPD and automatically dispense hints to the students whenever the system detects that they are stuck or frustrated. If the system detects wrongly that the student is frustrated when it is actually not the case, the automatic popping up of the hint may end up frustrating the student instead. Thus, for our tutoring architecture, students are left to decide when they require help and at what level of help in a tutoring session. The system will just display a slide in gentle reminder that help is available if required. By providing help on demand, the tutoring system reduces the probability that the help provided is at an inappropriate time (which can possibly lead to students getting frustrated that they are getting help which they do not need).

A disadvantage of putting the request for hint under the students' control is that some students may game or misuse it by simply requesting for bottom out hints without putting in effort to think about the problem on their own first. To mitigate the gaming issue, I have put in a simple timing based mechanism by ensuring that the hint dialog box stays open for at least 5 seconds. The duration of 5 seconds is configurable within the tutoring system. Students would not be able to dismiss the hint dialog box before 5 seconds is up and a message encouraging them to spend some time to read the hint would be displayed. In addition, students will not be able to request for the next hint unless 30 seconds have elapsed since the request of the previous hint. An empathetic message emphasizing that retention will be higher if they try to resolve the problems on their own first would be shown if students try to request for hints before 30 seconds is up. The 5 seconds hint duration where the hint dialog box stays open encourages students to spend the 5 seconds to read and understand the hint and the 30 seconds delay between consecutive requests for hints encourages students to put in effort to try and tackle the problems on their own before requesting for the next hint to prevent them from misusing the hints. However, if the system detects that the student is frustrated, the 30 seconds delay will not apply and the student will be able to request for hints even if the previous hint request is within the last 30 seconds.

One principle of design of the help system provided by the tutor is that it must be contingent on the students' needs (Wood & Wood, 1999).When students make a mistake, it is difficult for the system to ascertain whether it is due to slips, misconceptions or a lack of knowledge. If we offer only hint that points at a generic concept or area to look into, weaker students would most likely be still stuck. On the other hand, if we provide hint that is so detailed that it almost gives the answer away, the more proficient students who slip occasionally would not be sufficiently challenged and may also be bored or frustrated by the need to go through an elaborate explanation text when they already know the concept. This justifies the need for multiple levels of hints (from generic to specific) so that students who slip would not have to go through the more detailed explanations that are meant for students with a genuine lack of knowledge. For students who are genuinely stuck due to knowledge gaps, the help provided would have to be progressively more explicit until the student is able to resolve the problem. The first instance when students request for hint, a generic hint relating to the topic would be displayed. For subsequent help requests, detailed hints would be provided to the students. If the system detects that the student is frustrated, the generic hint would be skipped and a detailed hint would be provided to the student straight. The detailed hints are bottom out hints that provide an explanation of the error and the steps to be taken to rectify the error.

The flow of the problem solving process is shown in Figure 6-3. Students will first key in codes into the placeholders within exercises in the tutoring system. The students will then compile the codes and when errors are encountered, students will try to debug and rectify the errors on their own. Some students may experience frustration after repeated attempts to rectify the errors. The tutoring system will detect that the students are frustrated and a message will slide in on screen, reminding them that they can request for hints to resolve the errors. The students can also request for hints on their own without the reminder message by clicking on the request for hint button. If students signal that they do not want to have hints at this point, the system will display an empathetic message encouraging the students to continue resolving the errors on their own. This would be an emotional-focused response. The time at which the system detects that the student is frustrated is also logged into the database. By correlating this with the other action logs, we would be able to infer the line of code that the student is working on before they are frustrated and from there, the concepts in which the student is facing difficulties with. By aggregating this across the entire population of students, we would be able to decipher which are the programming concepts that most students struggle with.

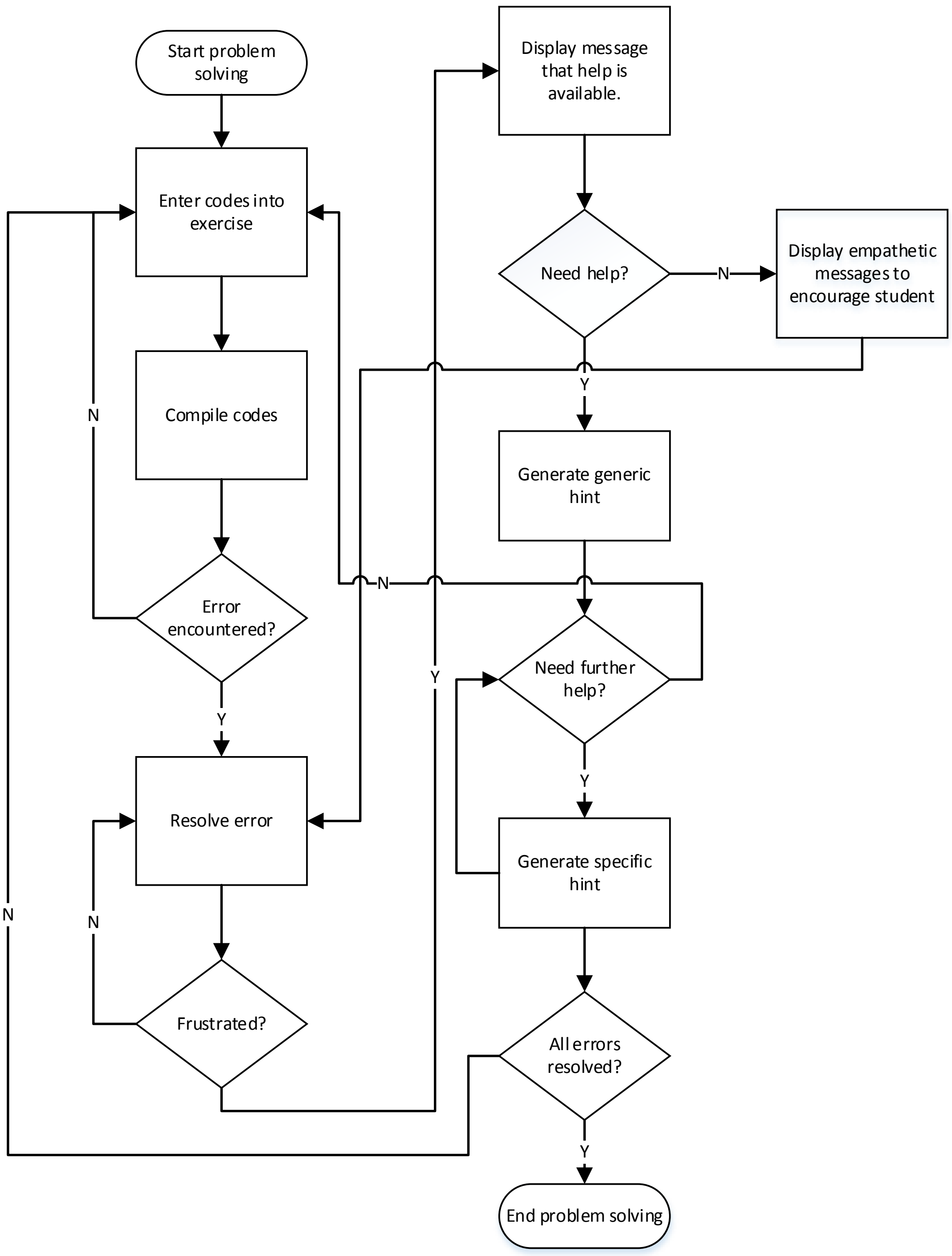


Figure 6-3: Student's problem solving process

6.4. Design of Hint Generation Module

The programming exercises in APTS require students to fill in missing lines of codes. The codes with missing lines to be filled in and the model solutions for the missing lines in the programming exercises are stored in the database. Tutors are required to annotate the hints by providing the possible model solutions for the missing lines of codes.

The generation of contextual hint entails the identification of the syntax or logical error. There are a number of possible approaches for the identification of errors. One way is to represent both the student's program and the model program as Abstract Syntax Trees (ASTs) and Augmented Object-oriented Program Dependence Graphs (AOPDGs) as proposed by Xu and Chee (2003). Transformations are applied to standardize or canonicalize the ASTs and to represent both the student's program and model program in AOPDGs. A comparison is then performed and unresolved differences are reported as errors while unmatched statements with no unresolved difference are reported as controlling errors. This transformation approach is also used by Rivers and Koedinger (2013) in their study. A similar canonicalization approach is taken in this study where both the student's program and the model program are parsed as ASTs and then canonicalized.

Barnes, Stamper, Lehmann, and Croy (2008) uses a machine learning technique, the Markov Decision Process (MDP) that is automatically created from past students' data to generate contextualized hints. An advantage of this approach is that minimal annotation of possible solutions by the tutors is required. Only a one time wording of hints have to be performed. Historical student data is used to generate the MDP model that contains the problem states and student actions. Reinforcement learning is then used to train or generate the optimal solution for the MDP. Using the MDP, a hint file consisting of all problem states in the MDP with available hints is then created. This approach requires an MDP to be built off-line for each problem using historical student solutions. The authors have managed to build up several semesters of data to create a large MDP for each problem by using it in a computer aided instructional tool for a number of years. Thus, a sufficient amount of data consisting of student solutions is required for training the MDP. A sufficiently large student solutions set is required for adequately accurate hints to be generated. If the problem changes, the student solutions and the MDP model for the new problem will have to be respectively collected and trained again. Thus, a shortcoming of this technique is that a relatively large training data set consisting of past students' solutions for the specific problem must be collected before hints can be accurately generated.

Another possible contextualized hint generating solution is the intention matching approach where the student's program is matched to the intended program. An intention (Johnson, 1990) or reference programs are correct solutions to a given problem. An adaptation of the Levenshtein distance algorithm (further elaborated in 6.4.3), similar to that proposed by Vee, Meyer, and Mannock (2006) is used in this study. A benefit of the intention matching approach is that both syntactical and logical programming errors can be detected. In contrast, a shortcoming of modern compilers is that they cannot detect logical programming errors. An added benefit of this approach is unlike the MDP technique, it does not require a large historical database of past students' solutions for every problem to seed the model. Although it was stated in Vee et al. (2006)'s study that they would extend it to incorporate a hint or feedback mechanism for students, there was no subsequent study published by the authors on the proposed extension. In this study, I have implemented a full-fledged hint generation module using a combination of program canonicalization, intention matching and rule based generation techniques to generate contextual and easy to interpret hint messages for students.

6.4.1. Flow of hint generation

To summarize the steps taken to generate the hint, the flow of hint generation is shown in Figure 6-4.

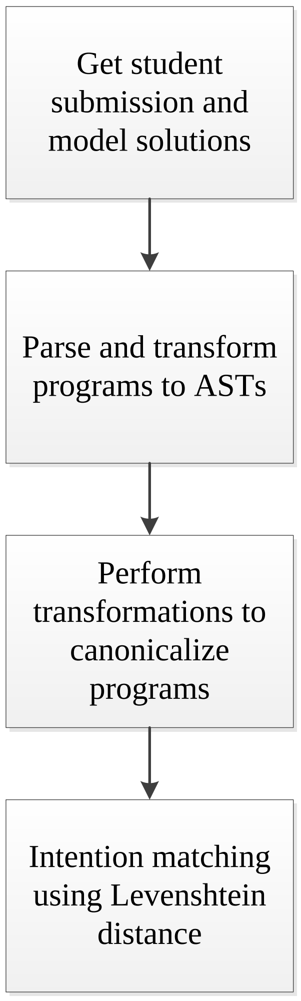

Get student submission and model solutions
Parse and transform programs to ASTs
Perform transformations to canonicalize programs
Intention matching using Levenshtein distance

Figure 6-4: Flow of hint generation

The list of model programs for the current exercise is retrieved from the database. The student’s program and list of model programs are next parsed into tokens and transformed into ASTs. The list of transformations as listed in 6.4.2 is then performed on both the model programs and student’s program. In the final step, the Levenshtein distance algorithm is then applied to both find the best match model program and to derive the list of transformations for rectification of the errors in the student’s program.

6.4.2. Program Canonicalization

Although the students are required to just fill in missing lines of codes within the exercises, variations such as use of different variable names and ordering of variables would still exist among the students' codes and even between the students' codes and the provided model solution. As an example, the two code lines "sumOfNum = firstNum + secondNum;" and "sumOfNum = secondNum + firstNum;" are both syntactically correct and produce the same output. Thus, the students' codes would need to be canonicalized or normalized before they can be matched to the model solution for the generation of hints. Xu and Chee (2003) defined a list of 13 Semantics Preserving Variations (SPVs) for canonicalizing students' programs. The list of 13 SPVs is shown in Table 6-1.

SPV1	Different algorithms e.g. different sorting algorithms.
SPV2	Different source code formats e.g. whitespaces and comments.
SPV3	Different syntax forms e.g. The same logic can be expressed in a single statement or in multiple statements.
SPV4	Different variable declarations
SPV5	Different algebraic expression forms
SPV6	Different control structures
SPV7	Different boolean expression forms
SPV8	Different temporary variables
SPV9	Different redundant statements
SPV10	Different statement orders
SPV11	Different variable names
SPV12	Different program logical structures
SPV13	Different statements

Table 6-1: List of Semantic Preserving Variations

For this study, in consideration of the fact that students are not required to write an entire program but just fill in missing lines of codes for short and simple programming exercises within the tutoring system, only SPVs 2, 5, 7 and 11 are implemented. Contingent on future needs, we would consider the implementation of other SPVs. The students' codes are first parsed into tokens and then formulated into ASTs using the Another Tool for Language Recognition (ANTLR) library ("ANTLR," 2017). The whitespaces and comments are eliminated during the transformation to ASTs.

To handle SPV 5, commutative, associative and distributive operations are normalized. Commutative operations are algebraic operations where the operands can be switched e.g. a+b=b+a or a*b=b*a. To normalize commutative operations, the operands are sorted in alphabetical order e.g. if the algebraic expression is "totalMark = testMark + examMark;", it will be normalized to "totalMark = examMark + testMark;" as the operand "examMark" is before "testMark" by alphabetical order. This is done so that the order of the operands within a commutative operation does not matter.

For SPV 7 – different boolean expression forms, expressions of the form "age>16 && age <65" will be normalized to the form "age<65 && age>16". The expression "age<65" is assigned a weight that is less than that of "age>16" and thus "age<65" is placed before "age>16".

To handle SPV 11 – different variable names, the variables are replaced with their normalized variable names. When a variable declaration is encountered for a missing line, it will be replaced with normalized variable name of the form "var_sequenceNumber" where sequenceNumber is a number that is incremented for each new variable declaration. All instances of the same variable will be replaced with the normalized variable name.

Other transformations that are applied include the normalization of shortcut operation such as "+=". To illustrate, the expression "i+=2" will be normalized to "i=i+2" and the expression "i%=2" will be normalized to "i=i%2".

6.4.3. Intention matching

Using the ANTLR library, the codes are first tokenized and then matched to the intentions. The ANTLR library is used to generate a Java tokenizer. The Levenshtein distance (Levenshtein, 1966) or Minimum Edit distance algorithm is used to compute the difference between the student's code and the intended code line by line. The Levenshtein distance between 2 lines of code yields the minimum number of insertions, deletions or substitutions required to change one line of code to another. Mathematically, the Levenshtein distance is given by

$$lev_{sc,mc}(i,j) = min\begin{cases} lev_{sc,mc}(i-1,j)+1 \\ lev_{sc,mc}(i,j-1)+1 \\ lev_{sc,mc}(i-1,j-1)+1_{(sc_i \neq mc_j)} \end{cases}$$

Equation 38

where sc is the line of student code and mc is the line of model code,

i is the number of tokens in sc and j is the number of tokens in mc.

$1_{(sc_i \neq mc_j)}$ is the indicator function and it is 0 when sc is not equal to mc.

For each line of student code to be keyed in, multiple lines of model code will have to be input by a human tutor. Each line of student code is then matched against a number of model code lines to derive the model code line with the lowest Levenshtein distance. The model code line with the lowest Levenshtein distance when compared to the student code line will be the solution that has the closest match to the student's code.

A by-product of Levenshtein distance other than yielding the minimum number of insertions, deletions or substitutions is the list of changes to transform the student code to the model code. To illustrate, suppose the line of student code is

int taxRate = 0.2

and the line of model code is

float taxRate =0.2;

The list of transformations generated would be to *replace int with float* and to *insert ; at the end of the line*. I shall refer to these transformations as the list of raw hints.

As mentioned earlier, an adapted version of the intention matching algorithm similar to that proposed by Vee et al. (2006) is used in this study. This can be explained with the use of an example below.

Suppose that the student's line of code is

int deduction = 2000

whereas the model line of code is

static final int DEDUCTION=2000;

		int	**deduction**	**=**	**2000**
	0	1	2	3	4
static	1	$\underline{1}$	2	3	4
final	2	$\underline{2}$	2	3	4
int	3	$\underline{2}$	3	3	4
DEDUCTION	4	3	$\underline{3}$	4	4
=	5	4	4	$\underline{3}$	4
2000	6	5	5	4	$\underline{3}$
;	7	6	6	5	$\underline{4}$

Table 6-2: Levenshtein edit distance matrix of student code and model code

Using the adapted Levenshtein's distance algorithm for intention matching, the edit distance matrix shown in Table 6-2 is generated. The underlined costs are the minimum cost for each row. A by-product of this algorithm is the generation of the list of transformations below.

```
INSERT STATIC static at position 1
INSERT FINAL final at position 1
REPLACE IDENTIFIER deduction with IDENTIFIER DEDUCTION at position 2
INSERT SEMI ; at position 4
```

There are a number of model solutions for each line of student code. Each line of student code is matched against each of the possible model solutions using the algorithm below. The algorithm generates the edit distance matrix for each pair of student code and model solution and outputs the Levenshtein distance or cost.

1. Repeat the below steps for each pair of student code and model codes.
2. Generate the Levenshtein distance matrix
3. Find the minimum cost for each row by comparing the costs across the columns for that row.
4. If there is only 1 minimum for that row, mark the column as taken and store the cell value and cell location as the minimum cell value and location for that row.
5. If there is more than 1 minimum for that row, check that the column is not already marked as taken; For the first column that is not already marked as taken, store the cell value and cell location as the minimum cell value and location for that row.
6. The Levenshtein distance is the sum of all the stored minimum cell values.
7. The line of model code that is closest to the line of student code is the one with the least Levenshtein distance.

6.4.4. Generation of meaningful and explanatory hint messages

The list of transformations or raw line hints generated using the algorithm in the above section only instructs the student to either insert, delete or replace a token. As the objective is to generate meaningful hints with explanations on why a particular step is necessary, this is not adequate.

The Drools rule engine ("Drools - Business Rules Management System," 2017) is used for the formulation of rules to map the raw line hints into detailed hints with explanatory text. Drools is a Business Rule Management System (BRMS) with a forward and backward chaining inference based rules engine optionally also called a production rule system. The production rule system comes with an implementation of the Rete algorithm, a pattern matching algorithm for rule-based system created by Forgy (1979). A production rule system is used here as it allows the teachers to modify the production rules for the generation of hints on their own if required. The rules can be modified even by a non-technical person. In addition, other than the effort to key in the rules in the first instance, effort to maintain the rules is minimal as most of the rules can be re-used.

An alternative is to use a stochastic approach such as the use of a Bayesian network. The study by Conati, Gertner, VanLehn, and Druzdzel (1997) uses a Bayesian network to generate help messages for students in the Andes, a physics tutoring system. In Andes, Bayesian network is used for plan recognition (inference of the most likely strategy the students will follow) and prediction of student's goals and actions. Bayesian network is not used in our context for generation of hints as it is computationally complex especially for a network with large number of nodes. The computational complexity of propagation of evidence through the network will result in a delayed response to the student. An added consideration is that the Bayesian network will have to be re-designed and re-learned if we swap to a different programming language for our tutoring context. In contrast, for our proposed solution using the rule engine, the rules can be re-coded easily by someone who is not well versed in the Java programming language.

A list of frequent novice Java programming errors is referenced from previous research. N. C. Brown and Altadmri (2014) collected a list of top 18 novice programmers' errors after collecting java codes and error messages from around 110,000 users. After discussion of colleagues who is teaching Java programming, we decided to reduce this list of top 18 novice programmers' errors to a list of 15 instead. The list comprising of 15 student mistakes include

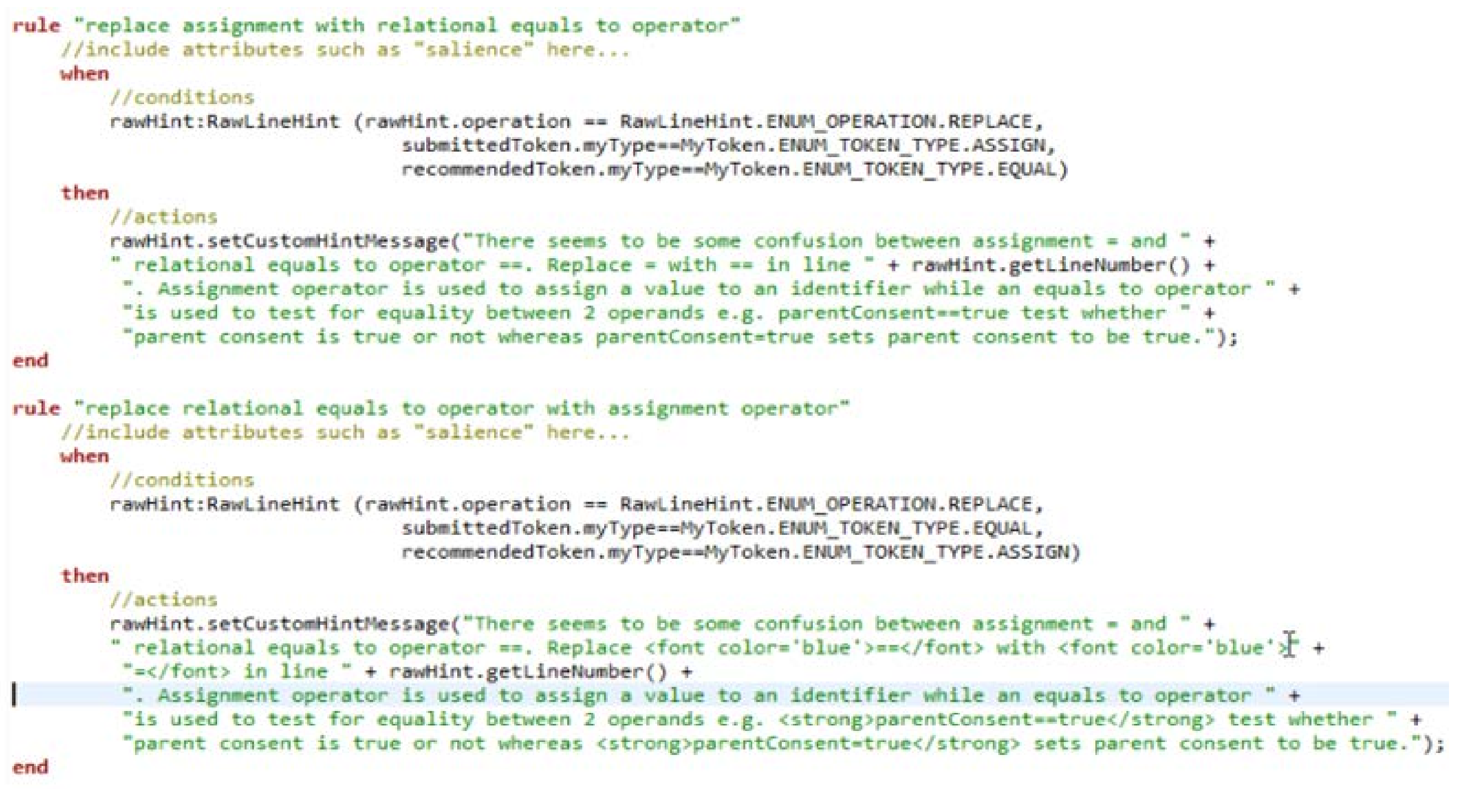
1	Confusing assignment operation (=) with the comparison operator (==).
2	Use of == instead of .equals to compare strings.

3	Unbalanced parenthesis, curly brackets, square brackets and quotation marks, or using these different symbols interchangeably.
4	Confusing "short-circuit" evaluators (&& and ||) with conventional logical operators (& and |).
5	Incorrect semi-colon after an if-selection structure before the if statement or after the for or while repetition structure before the respective for or while loop.
6	Wrong separators in for loops (using commas instead of semi-colons).
7	Inserting the condition of an if-statement within curly brackets instead of parentheses.
8	Using keywords as method names or variable names.
9	Invoking methods with wrong arguments (e.g. wrong types).
10	Forgetting parentheses after a method call.
11	Incorrect semicolon at the end of a method header.
12	Getting greater than or equal/less than or equal wrong.
13	Trying to invoke a non-static method as if it was static.
14	Including the types of parameters when invoking a method.
15	Incompatible types between method return and type of variable that the value is assigned to.

Table 6-3: List of top novice programmers' errors

The list of top novice programmers' errors listed in Table 6-3 is used to seed the formulation of rules for the generation of detailed hints. An example of the rule for resolving the confusion of assignment (=) with comparison operator (==) is shown in Figure 6-5.

In the above rule, each time when a raw line hint to replace an assignment token with a relational equal token was generated by the intention matching algorithm, it will be translated by the rule into a detailed hint message stating that there is a confusion between the assignment (=) and relational equals to (==) operator and an explanation is presented on the use of each operator to correct the student's misconception.



```
rule "replace assignment with relational equals to operator"
    //include attributes such as "salience" here...
    when
        //conditions
        rawHint:RawLineHint (rawHint.operation == RawLineHint.ENUM_OPERATION.REPLACE,
                            submittedToken.myType==MyToken.ENUM_TOKEN_TYPE.ASSIGN,
                            recommendedToken.myType==MyToken.ENUM_TOKEN_TYPE.EQUAL)
    then
        //actions
        rawHint.setCustomHintMessage("There seems to be some confusion between assignment = and " +
        " relational equals to operator ==. Replace = with == in line " + rawHint.getLineNumber() +
         ". Assignment operator is used to assign a value to an identifier while an equals to operator " +
         "is used to test for equality between 2 operands e.g. parentConsent==true test whether " +
         "parent consent is true or not whereas parentConsent=true sets parent consent to be true.");
end

rule "replace relational equals to operator with assignment operator"
    //include attributes such as "salience" here...
    when
        //conditions
        rawHint:RawLineHint (rawHint.operation == RawLineHint.ENUM_OPERATION.REPLACE,
                            submittedToken.myType==MyToken.ENUM_TOKEN_TYPE.EQUAL,
                            recommendedToken.myType==MyToken.ENUM_TOKEN_TYPE.ASSIGN)
    then
        //actions
        rawHint.setCustomHintMessage("There seems to be some confusion between assignment = and " +
        " relational equals to operator ==. Replace <font color='blue'>==</font> with <font color='blue'>" +
         "=</font> in line " + rawHint.getLineNumber() +
         ". Assignment operator is used to assign a value to an identifier while an equals to operator " +
         "is used to test for equality between 2 operands e.g. <strong>parentConsent==true</strong> test whether " +
         "parent consent is true or not whereas <strong>parentConsent=true</strong> sets parent consent to be true.");
end
```

Figure 6-5: Rule to generate detailed hint message for use of assignment and relational equals to operator

6.5. Drools Rule

A Drools rule starts with the keyword ‘rule’ followed by the rule name. It is followed by a ‘when’ section and a ‘then’ section. The ‘when’ section is used for specifying the condition for the matching of the rule and the ‘then’ section is used for specifying the actions to be performed when the condition in the ‘when’ section is matched. The condition is specified based on the output from the Intention Matching algorithm. To illustrate, for the example rule in Figure 6-5, the condition for matching this rule is the ‘OPERATION’ is ‘REPLACE’, the token to be replaced (i.e. the wrong token) is an ‘ASSIGNMENT’ (=) token and the correct token is an ‘EQUAL’ (==) token. In this case, if the student mistakenly used an assignment operator instead of a relational equals operator (e.g. if (a=b) when it should be if (a==b) instead), the action would be to display the message encoded in the ‘when’ section.

6.6. Summary

The implementation of an ATS is rare in the literature and even less so, an implementation of one for the domain of computer programming. In this study, I have proposed an architectural design for APTS that predicates on some of the pedagogic and psychological considerations to address an existing gap on how to respond appropriately to the detected emotions of the students to increase their engagement and task persistence and to sustain their motivation to learn.

I have also discussed the significance of hinting as a tactic used frequently by human tutors and as a form of support for the learning of concepts and skills. In my tutoring system, gradated levels of hints are used to alleviate the negative affect of students in the learning process. Finally, I also discussed the various techniques to automatically generate contextualized hints. Out of all these approaches, the abstract syntax tree transformation combined with intention matching approach was found to be the most adequate for our context where students are required to fill in missing lines of codes within each programming exercise. The methodology which involves program canonicalization, adaptation of the

Levenshtein's distance algorithm for intention matching and the use of a rule engine to formulate the rules for the elaboration of hints were detailed as well. The rules are formulated based on the common programming misconceptions documented in the literature which will result in hints that rectify the majority of the common misconceptions and errors made by the students. These summarize the approach taken in the design and implementation of the APTS.

Chapter 7. Evaluation of system

7.1. Introduction

This chapter details the use of both an experimental study and focus group discussions for both quantitative and qualitative evaluation of the effectiveness, usability and acceptance of the Affective Programming Tutoring System (APTS). The conduct of the experimental study and the focus group discussions and an analysis of the results are elaborated in the subsequent sections.

7.2. Experimental Study

For the experimental study, the effectiveness of the APTS for the students' learning was evaluated. As the proposed APTS is to be used as a Java programming tutor, to evaluate its effectiveness, I seek to measure the learning achievements of the students during the trial. Specifically, the learning achievement for this study is defined as the number of exercises completed, the duration of time taken to complete each exercise, the number of exercises attempted and the number of compilations required to complete each exercise. These can be derived from the students' action logs that are captured within the Student's Action Module elaborated in the section on Proposed Architecture. The argument is that if the APTS is effective, it should motivate the students to attempt and complete more programming exercises (as compared to students who do not have access to the affective functions of APTS) and also to complete them in a shorter amount of time with fewer compilations. Our hypothesis for this study is thus the full affective version of the APTS results in more effective tutoring as compared to the version with the affective function disabled.

This measurement of learning achievement through tracking the number of programming exercises completed and the time taken to complete them was also adopted in the study by Corbett and Anderson (2001). In his study, they further concluded that test achievement was strongly related to the set of problems the students successfully solved and the solution paths that the students follow had little or no impact (Corbett & Anderson, 2001), thus validating our approach to focus on the outcomes and not the process of problem resolution for measurement of learning achievement.

7.3. Focus group discussions

Focus group discussion is a group interviewing technique relying on interaction within the group based on topics that are selected by the researcher (Morgan, 1996b). The participants within the focus group are also usually selected by the researcher. In a focus group, the interaction among the members of the group allows the researcher to extract the attitudes, feelings, beliefs and experiences of the members which may not be feasible with the use of other research methods. Focus groups work best in situations in which the respondents feel free and comfortable to express their opinions without fear of reprisal, allowing the researcher to know what their respondents' genuine feelings are (Morgan, 1996a).

In this research, the developed APTS will likely be deployed in a classroom or computer laboratory to complement a human tutor in the teaching of computer programming. The use of focus group discussions facilitates the retrieval of a range of students' perceptions on the effectiveness of the system for their learning. In addition, it allows for the extraction of genuine feelings and opinions of the students on the condition that the environment is comfortable and conducive for them to voice their views. It is also envisaged that a group's capability would be more than the sum of its parts, thus generating more opinions than what can be obtained from the sum of opinions from each individual interview.

Focus group discussions are frequently used as a qualitative evaluation instrument to triangulate information and data that are gathered both for informing the early design of as well as for the evaluation of tutoring systems' effectiveness. The Writing Pal is one ITS that employed the use of focus group discussion to inform its early content and design (Roscoe, Allen, Weston, Crossley, & McNamara, 2014). COSMO (Towns, FitzGerald, & Lester, 1998) and a PHP ITS built by Queensland University of Technology (Weragama & Reye, 2014) are ITSs that employed the use of focus group discussions to evaluate its tutoring effectiveness.

Focus group discussions with expert teachers were used to feed into the design of its writing strategy modules. The writing strategy modules teach strategies for prewriting, drafting and revision for essay writing. The focus group discussions were found to be beneficial in assessing the users' interactions with the system and informing on the functional enhancements to the system.

7.4. Conduct of Evaluations

7.4.1. Experimental Study

In an experimental study, a researcher actively manipulates which group receives the designated treatment in a randomized control trial and then records the outcome. A benefit of experimental study is that it is less susceptible to confounding as each participant is randomly assigned to either the test or control group (Oehlert, 2010).

In our context, a total of 39 students participated in the experimental study for evaluation of the effectiveness of the APTS on their learning. The 39 participants were 18 years of age on average and were undertaking an Information Technology diploma in Nanyang Polytechnic. They had undertaken one semester of Java programming a semester ago. All the participants were first requested to sign consent forms indicating their consent to participate in the trial and for collection and publication of data captured in the trial. They were also briefed on the basic functions of the APTS and that they will be required to complete as many Java exercises as possible in the APTS for an hour in the trial. Each of them was assigned a user id and password to login to the APTS. The participants were next randomly assigned to either the control (n=21) or the test group (n=18). Participants in the control group worked on a version of the APTS without the affect related functions while participants in the test group worked on the full affective version.

7.4.2. Focus Group Discussions

In our context, a total of 4 focus group discussions were conducted with the objective of evaluating the usability and acceptance of the APTS. 28 students comprising of 12 females and 16 males in total (different from the ones in the experimental study) were recruited for the focus groups with 6, 7, 8 and 7 students for each of the 4 focus groups. The student participants are 18 years old and were then undertaking an Information Technology diploma in Nanyang Polytechnic. They had undertaken a basic course in Java for a semester about a year ago.

The participants were asked to trial the APTS for an average of 30 minutes before the start of the focus group discussion to familiarize them with the functions of the system. The focus group discussions were next held in classrooms with the participants seated in a circle. Each focus group discussion lasted for an average of about 20 minutes. The participants were asked to self-introduce themselves at the start of the session. All the focus group sessions were conducted by a moderator and were tape-recorded. After each session, the tape recordings were then transcribed. A set of questions was formulated with the objective of evaluating the effectiveness of the APTS on the students' learning. These questions serve as a guide to the moderator and the moderator is free to rephrase and re-sequence the questions when he or she conducts the discussion session. Two moderators with an average teaching experience of 7 years each conducted the focus group discussion sessions.

The set of guiding questions for the focus group discussions are

1. How was your learning experience with the tutoring system?
2. Why do you feel that way about the tutoring system?
3. What do you like about the tutoring system?
4. What do you think can be improved for the tutoring system?
5. What do you think if the system can sense your frustration and engagement and respond accordingly e.g. by providing you with hints if it senses that you are frustrated?
6. What else would you like to say about the tutoring experience or about the system?

During the conduct of the session, the moderator tried to ensure that every participant had an equal opportunity of expressing his or her view on the topic and that the session is not dominated by any individual participant (by tracking the time taken by each participant and prompting for the next participant to start). After the session, the moderator thanked each participant for their participation and assured the participants that their views are valued and will contribute towards the enhancement of the APTS which will in turn benefit future users of the APTS. An observation from the focus group discussions is that the student participants, who are all Asians, were generally reserved in expressing their opinions and the moderators had to actively probe them for further comments during the sessions.

7.5. Results

7.5.1. Experimental Study

The descriptive statistics for the number of exercises completed, time taken to complete exercise, number of compilations and number of exercises attempted for the non-affective (control) and affective (test) groups are shown in Table 7-1.

The time taken to complete each exercise was compiled for each exercise completed by individual students, resulting in a total of 65 records for the non-affective group and 60 for the affective group. The number of compilations was recorded for each compilation attempt by each student, resulting in a total of 101 and 118 compilation attempts by the students in the non-affective and affective group respectively.

Groups	**No of exercises completed**		**Time to complete exercise**		**No of compilations**		**No of exercises attempted**	
	Median	**n**	**Median**	**n**	**Median**	**n**	**Median**	**n**
Non-affective	3	21	831	65	7	101	5	21
Affective	3.5	18	470.5	60	13	118	7.5	18

Table 7-1: Descriptive Statistics for determinants of learning achievement

As the data from the experimental trial did not fulfill the assumption of normality, the Mann-Whitney U test (Mann & Whitney, 1947) was used to determine whether the students are able to achieve more with the use of the full APTS functionality. The Mann-Whitney U test also known as the Wilcoxon rank sum test, tests for differences between 2 groups on a single ordinal variable with no assumption of a specific distribution.

The Mann-Whitney test results showed that the time taken to complete each exercise ($U=1406$, $n1=60$, $n2=65$, $z=-2.69$, $p<0.05$ two-tailed) and the number of exercises attempted ($U=83$, $n1=18$, $n2=21$, $p<0.05$ two-tailed) were significantly different for the non-affective and affective group. On the other hand, the number of exercises completed ($U=174$, $n1=18$, $n2=21$, $p<0.05$ two tailed) and the number of compilations ($U=5907$, $n1=101$, $n2=118$, $z=-0.112$, $p<0.05$ two tailed) were not significantly different for the non-affective and affective groups.

7.5.2. Focus Group Discussions

General feelings on the tutoring system

Most of the students expressed that they were favorably disposed towards the use of the APTS for e-learning or self-directed learning or as a complement to the face-to-face lessons. Generally, their experience on the 30 minutes or so that they spent working on the tutoring system was positive.

Design of programming exercises

Cognitive Load Theory (Sweller, 1988) is a theory that provides the framework for presentation of instructional information to optimize the cognitive load which will in turn lead to enhanced intellectual performance. It theorizes that if the total cognitive load exceeds the available working memory capacity, learning would not be able to occur. An effective instructional design reduces the cognitive load which puts less strain on the working memory requirements. The freed working memory in turn allows the learner to acquire other more advanced schemas which leads to improved learning when repeated over many cycles.

The students reflected that the programming exercises were beneficial to their learning as on the one hand, it allowed them to apply whatever they have learnt from the lessons and on the other, to assess the adequacy of their knowledge for the various topics. They also felt that filling in missing lines of codes in the exercises was less daunting than writing an entire program on their own. For some of the students, they recalled that frequently when writing a computer program from scratch, they did not know how to start. To top it up, the cognitive load of having to interpret long lists of errors made it difficult for them to identify the exact cause and location of the errors. They felt that the filling in of missing lines design of the exercises within the tutoring system substantially lessened the cognitive load and made it easier for them to pick up the relevant programming concepts.

Hint function

The top function of the tutoring system that most of the participants found most beneficial to their learning was the hint function. They found that the hints were especially useful for students who are weaker in Java programming as the bottom-out hints told them exactly how to correct the related errors. As the student aptly puts it, "The hint gives the answer e.g. put in the static keyword in line XX… the hint is the top feature and it can remind me of stuff that I missed out".

In the trial, most of the students used the hint function although some of the students also reflected that they did not understand the hints. They expressed that the hints could be in a form that was easier to understand. This is surprising given that the students are bi-lingual in both English and Chinese. The hints were formulated such that they not only offered the student steps on how to fix the relevant error but also a detailed explanation of the syntax or logical error. This appeared to put some of the students off due to the length of the hint text. In the trials, the duration of time in which the students kept the hint message box opened was tracked and it was found that most of the students closed the hint message box within a few seconds, suggesting that they might be scanning the hints quickly without spending much effort to read and understand the accompanying explanation text. Some mechanisms may have to be built into the hint module to encourage the students to spend more time reading and contemplating about the provided hint and explanation to further enhance their understanding.

Usability of tutoring system

On the usability of the tutoring system, most of the students reflected that they were able to navigate around the various functions of the system without much help. Some of the students felt though that they needed some time to familiarize themselves with the various functions of the tutoring system. This was despite the fact that they were given a brief introduction to the various functions of the tutoring system before the trial. These students felt that the user interface of the tutoring system can be more intuitive and that the colors used for the user interface can be more vibrant especially if it was to be used for their self-directed learning. One suggestion was to incorporate a pre-recorded video tutorial that introduces the students to the functions of the tutoring system.

To solve the tutorial exercises, the students would have to first key in the missing lines of codes in the code window, submit their completed codes for compilation, check for compilation errors in the output window and then activate the verification of the output in sequence. This resulted in some students mixing up the sequence of these steps. They reflected that it would be good if the system can provide guidance messages advising them to for example submit their changed codes for compilation and checking that there are no errors in the output window before allowing the students to verify their output against the expected output.

Sensing of emotions

The students were briefed at the beginning of the trial that the tutoring system will be equipped with a web camera that will sense their frustration and engagement by capturing their facial expression, keystrokes, mouse clicks and head postures, and contextual logs. Most of the students in the focus group discussions, when told that web cameras will be installed to monitor their facial expressions and head postures, were surprised. However, when the moderator explained that this was necessary to sense their frustration and engagement so that the tutoring system can respond with empathetically sensitive tutorial response, the students were able to accept the presence of the web cameras. 3 out of the 28 students however, expressed that they are still uneasy about the presence of the web cameras. As one student says it: "It's like being watched… Why do you need to watch me?"

Some students were also skeptical of the ability of the tutoring system to sense their emotions. These were the words of one of the student - "If it senses well, then it's good …… but what happens if it didn't sense well ……. Some people can code even when they are frustrated …… the question is how does it sense". It thus seemed that some students were not confident that the tutoring system could sense their emotions most probably because they did not understand how this emotion sensing mechanism works. They were also concerned with the consequences on their learning should the tutoring system sense their emotions wrongly.

7.6. Discussion of results

7.6.1. Experimental Study

The results of the experimental study showed that the time taken to complete each exercise and the number of exercises attempted by the students differed significantly between the non-affective and affective groups. The number of exercises completed and the number of compilations were however not significantly different between the two groups. This implies that the use of the full affective version of the APTS enhances the efficiency (solving exercises in a shorter time) and persistence (expending effort in attempting more exercises) of the students.

It is surprising though that the number of exercises completed was not significantly different. Closer examination of the data further revealed that the students using the full affective version were able to solve the easier exercises in a shorter time but they were unable to resolve the more difficult exercises covering more advanced concepts such as arrays and functions. One possible explanation for this is that these difficult exercises were scheduled towards the end of the one hour session and students were encouraged to solve the exercises in sequence from the fundamental to the more advanced level. It is plausible the students ran out of time for the resolution of the more difficult exercises which were attempted only towards the end of the session.

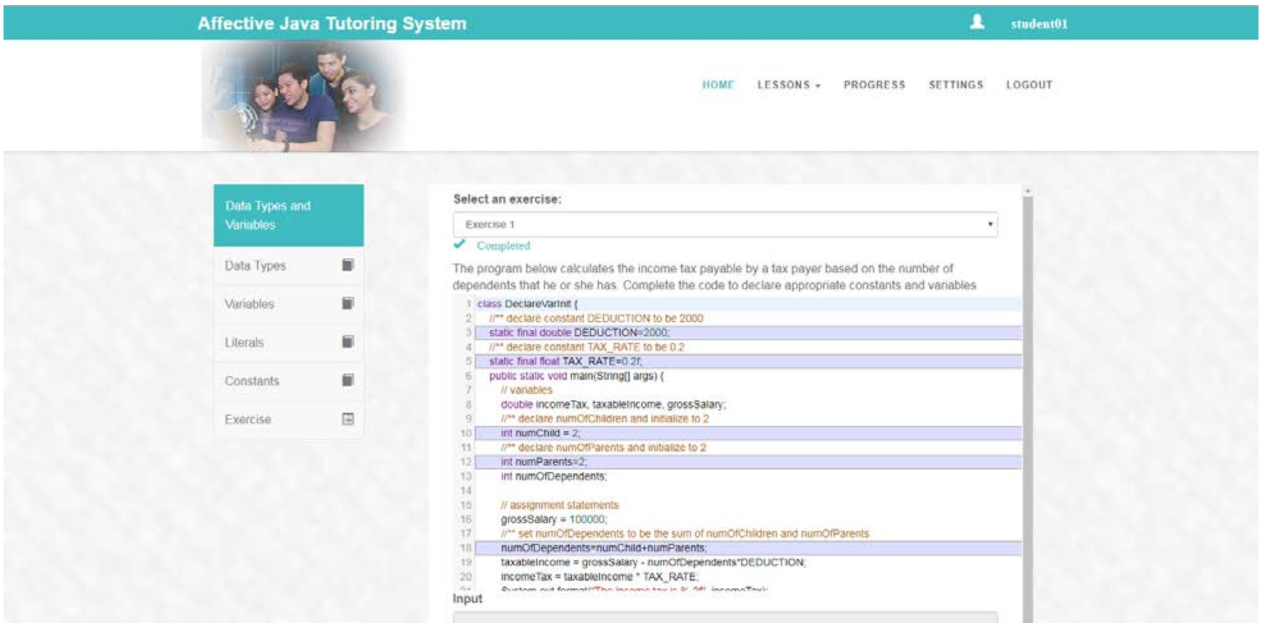
The number of compilations was not significantly different for the affective and non-affective groups. It is possible that more sophisticated measures that examine the compilation errors and the compilation behaviour of the students in resolving the errors between compilation attempts e.g. the error quotient as proposed by Jadud (2006) may be required. As this version of APTS did not capture the details of the compilation errors encountered by the students, this was not further investigated.

It is thus reasonable to support the hypothesis that the full affective version of the APTS results in more effective tutoring as compared to the version with the affective function disabled as the affective APTS does enhance the efficiency and persistence of the students in solving the programming exercises.

7.6.2. Focus Group Discussions

From the summary, the students were generally positive on their learning experience with the tutoring system and they were willing to use the tutoring system as a complement to face-to-face lessons conducted by their tutors and also as a form of e-learning. The top function that the students found most useful was the hint function as they felt that the hints provided helped them to resolve compilation errors that they were unable to solve. However, some students felt that the explanation provided in the hint messages could be further enhanced to be more comprehensible, readable and attractive to entice them to read it. These students suggested that audio messages might help. Other than audio, animations and videos may also help in encouraging the students to spend more effort on comprehending the provided hints. This can be implemented in future versions of the tutoring system and a research can then be initiated to investigate whether this would lengthen the reading time of the hints by the students and whether this would subsequently result in deep enhanced learning.

With regards to the usability of the tutoring system, the student suggested changing the color scheme of the tutoring system to be more vibrant. The color scheme of the tutoring system had been changed since and the new design is attached in Figure 7-1. A video tutorial that guides students who are new to the system on the features and functions of the tutoring system has also been completed. As for the last suggestion on the guidance messages to guide students on the compilation and checking of the output, I have instead implemented informational messages that point the students to what they should do next. For example, when the students are filling in the last missing line of code, an informational message box would appear which reminds them to click on the "submit code" button to submit their code for compilation. After successful compilation i.e. no compilation errors, a confirmation message box would appear automatically requesting the students to confirm whether they want to verify their code output against the expected output.



The program below calculates the income tax payable by a tax payer based on the number of dependents that he or she has. Complete the code to declare appropriate constants and variables

```
class DeclareVarInit {
   //** declare constant DEDUCTION to be 2000
   static final double DEDUCTION=2000;
   //** declare constant TAX_RATE to be 0.2
   static final float TAX_RATE=0.2f;
   public static void main(String[] args) {
      // variables
      double incomeTax, taxableIncome, grossSalary;
      //** declare numOfChildren and initialize to 2
      int numChild = 2;
      //** declare numOfParents and initialize to 2
      int numParents=2;
      int numOfDependents;

      // assignment statements
      grossSalary = 100000;
      //** set numOfDependents to be the sum of numOfChildren and numOfParents
      numOfDependents=numChild+numParents;
      taxableIncome = grossSalary - numOfDependents*DEDUCTION;
      incomeTax = taxableIncome * TAX_RATE;
```

Figure 7-1: New design of tutoring system

Some students were also skeptical of the ability of the tutoring system to sense their emotions accurately and the consequences on learning should the tutoring system sense their emotions wrongly. The tutoring system currently senses both frustration and disengagement of students and uses both prompts and hints to mitigate frustration and sustain the students in their learning. In a tutoring session, the sensed frustration and disengagement will also be aggregated across the students and presented in the form of an adequate visualization. This visualization when overlaid with the topic or sub-topic that the student is working on at the time when the frustration or disengagement was detected, offers insight to the human tutor on which topics or sub-topics the majority of the students are facing difficulty with. Should the sensing of the emotions be incorrect, the prompts and hints (that are designed to be displayed in a subtle manner), can simply be ignored by the students. This ensures that the tutoring response for incorrect emotion sensing will cause minimal disturbance to the students' learning. In addition, the aggregated visualization is for more informational purpose for analysis and action by the human tutor. Thus, it is reasonable to state that the consequences of wrong emotion detection for our applications are not dire. This can be better communicated to the students to address their concerns.

As for the privacy concerns of some students with regards to the monitoring through web cameras during the tutorial session, I believe that this can be mitigated to a certain extent with increased awareness and education. By convincing the students of the benefits of the tutoring system and assuring them that the real-time logs recorded would not be used for any other purposes than for that of optimizing their learning, I am optimistic that fewer students would be reluctant to use the tutoring system. This can also be affirmed by the growing popularity of wearables and mobile applications where users' personal data are collected and stored (with the consent of the users most of the time) by companies. The caveat is that these users would need to be assured that the data collected would be stored securely and not be misused for monetary purposes. Although I am conjecturing that with proper disclosure on the use of data, secure storage and user education on the technology used, the privacy concerns of users can be effectively mitigated, admittedly this is a broad issue that requires a separate study for in-depth analysis.

7.7. Conclusion

The evaluation of the APTS for its effectiveness, usability and acceptance by the students using both the quantitative technique of experimental study and the qualitative technique of focus group discussions is described in this chapter.

The results for the experimental study support the hypothesis that the full affective version of the APTS results in more effective tutoring as compared to the version with the affective function disabled as the time taken to complete each exercise and the number of exercises attempted by the students significantly differs between the 2 groups. Although the number of exercises completed was found to not significantly differ between the groups, this can be attributed to the time constraint imposed for the experiment and the sequence in which the students attempted the exercises.

Focus group discussions were used to evaluate usability and acceptance of the APTS by the students. The analysis of the transcripts for the focus group discussions further revealed that the students were generally positive on their learning experience with the APTS. The top rated functions as revealed by the students were the fill in the gap programming exercise and the hint function which offers explanation on top of the rectification steps. Arising from the concerns highlighted by the students in relation to the usability of the APTS, I had implemented functions that enhanced the APTS's usability.

This research also highlights privacy concerns of some students with regards to the monitoring of students' behaviors and actions via sensors in the APTS. One suggestion is for options can be provided for students in an APTS to opt in or out of the monitoring but opting out will also mean that the empathetic tutoring response will be disabled as well. This privacy and usability conflict is an issue that will have to be resolved for widespread deployment of APTSs in classrooms and would be a potential area for future research.

Chapter 8. Conclusion

8.1. Introduction

This chapter closes the thesis with a summary of the work done in this research, the main findings and contributions to the body of knowledge and the proposed areas for future expansions to this research.

8.2. Summary of work

In this research, I started by reviewing the literature on the various emotional theories and the classification of emotions. In the classification section, I discussed that emotions are commonly measured by valence and arousal on a 2 dimensional scale. However, complex emotions such as frustration and engagement cannot be represented on a dimensional scale and hence, a discrete representation of emotions is used in this work. I then moved on to review the role of affect on learning, laying the ground for the need to mitigate negative emotions such as frustration to sustain the students' learning. The need for augmenting Intelligent Tutoring Systems with emotional awareness and empathy next leads to the discussion of the few Affective Tutoring Systems in the literature. Affect sensing is the first step in the building of an Affective Tutoring System and thus in the literature review, I also discussed the use of the different modalities in affect sensing, reviewing the state of the art in affect sensing studies as well as the merits and demerits of each modality. The need for unobtrusive and cost-effective sensors for affect detection especially in an educational setting was then highlighted. Lastly, multimodal affect detection was put forth as an affect sensing technique that resembles human affect expression (in multiple modalities) and thus comparatively more promising than unimodal affect sensing. This thus led to my investigation into whether multimodal affect detection offers higher accuracy over unimodal affect detection.

As the use of keystrokes is novel, I initiated a study to establish the viability of keystrokes for the sensing of frustration by using machine learning techniques. From the results obtained, it was established that keystrokes is viable for sensing of frustration. However, there was a need to further establish the granularity of detection as the tutoring response to the students experiencing negative affect would have to be on a moment-to-moment basis, failing which the perfect moment for intervention would be missed. The data was first preprocessed into the feature vectors using a sliding window technique. Various window widths were investigated in this study. The keystrokes data were then divided by the baseline keystroke rates and the data with no keystrokes were next eliminated before being input into a Bayesian Network, a probabilistic based approach for inferring the frustration of students.

Although a fine granularity of detection is possible, the elimination of periods where no keystrokes are available led to breaks where no detection was possible. A multimodal fusion where multiple sensor outputs were combined into a single coherent output was thus proposed to resolve this issue. The research intuition here is that the different sensors will complement one another and lead to higher accuracy of detection. Various multimodal fusion techniques (including feature fusion and decision fusion techniques) of keystrokes, mouse clicks, head pose, facial features and contextual logs data using the Random Forests as the base classifier were investigated.

The engagement states of the students were next modeled with the head postures, keystrokes, mouse clicks and contextual logs features being input into a Hidden Markov Model (HMM) in an unsupervised learning approach. To alleviate human effort in annotating the videos for engagement states, an unsupervised learning approach was adopted. In contrast, adopting a supervised learning approach would require manual labeling of the output by human annotators. A HMM was used for modeling the temporal sequential characteristics of the engagement trajectory of the students. Through cross validation, the most likely number of states is then determined and the features' characteristics for each state was examined. The identification of the engagement states allow for tutorial intervention for students in disengaged states and the state transitions allow for an understanding into how students traverse between the different engagement states.

I next presented the architectural design of the Affective Programming Tutoring System and the methodology for the generation of context sensitive hints for the programming exercises. Both emotional-focused and problem-focused responses were designed into the tutorial strategy module of the system. The hint was also made progressively more detailed contingent on the need to cater to both students who slip occasionally and students who require a deeper level of assistance. The design of the context sensitive hint system was next described. In the literature, there are various techniques to design a context sensitive hint system for the domain of computer programming. An important factor in my hint system design is that the hint dispensed by the system must not only list the rectifying steps but also include an explanation of why the step is necessary. This led to my proposed implementation combining an intention matching algorithm with a rule based engine for hint generation.

To round up the work, the Affective Programming Tutoring System was evaluated through both an experimental study (quantitative) and a series of focus group studies (qualitative). The experimental study was used to evaluate the hypothesis that the full affective version of the APTS results in more effective tutoring as compared to the version with the affective function disabled. The focus group discussions, on the other hand were used for evaluation of the acceptance and usability of the tutoring system by the students.

8.3. Findings and contributions

A summary of the findings and contributions in relation to the research aim of exploring the feasibility of building an ATS using unobtrusive sensors for the learning of computing programming is listed below.

1. To develop a method for sensing the emotions of students that are detrimental to learning on a moment-to-moment basis from unobtrusive and cost effective sensors.

The use of unobtrusive sensors (keyboards, mouse and web cameras) in affect detection for the context of a tutoring system that tutors students in computer programming has been established.

2. To investigate the use of multimodal affect sensing techniques for enhanced affect detection.

   A multimodal system of affect detection using keystrokes, mouse clicks, contextual logs, facial and head postures combined using various fusion techniques was proposed and investigated.

3. To develop a method for formulating and adapting the tutoring response in accordance with the sensed emotions of the students for optimizing their learning.

   A pedagogical based model of responding to the students expressing negative affect centered on a context sensitive and comprehensive hint system was proposed and implemented.

4. To design an architecture and implement an ATS for the learning of computer programming.

   An architecture was proposed and the ATS was implemented and its effectiveness and acceptance evaluated through an experimental study and a series of focus group discussions.

In this research, I have explored the use of unobtrusive sensors in affect detection for the context of a tutoring system that tutors students in computer programming. More specifically, I have established the viability of using keystrokes, mouse clicks and contextual logs for the detection of frustration on a level of granularity that is adequate for timely remedial intervention. Keystrokes and mouse clicks were traditionally used in computer security domain for authentication and user identification purposes and were rarely used for affect detection and hence, the significance of the use of keystrokes and mouse clicks for affect detection in this research.

Another area of significance is in multimodal affect detection – the fusing of multiple sensing channels' outputs. It is a fact that human emotion is expressed in various channels e.g. facial, vocal and bodily expressions but implementation of a multimodal affect detection system is still rare in occurrence (Jaimes & Sebe, 2007). In this research, a multimodal system of affect detection using keystrokes, mouse clicks, contextual logs, facial and head postures combined using various fusion techniques was proposed and investigated. It was further verified that a multimodal fusion of the proposed sensors (AUC=0.914) outperformed the best unimodal channel (the facial channel with an AUC of 0.85). Although the features that contributed most to the accuracy of the multimodal model were the ones derived from head postures and facial channels, the keystrokes and mouse clicks filled in for the periods of no detection when both the head postures and facial features were not available. To illustrate, facial features were missing for 17% of the instances across all students for my study which is a semi-naturalistic setup where students were urged to remain in their seats for the web camera to get a full frontal view of their face for the entire session. In a naturalistic environment with lesser constraints imposed on the students, the availability of the facial features would likely be further degraded and thus the need for the keystrokes and mouse clicks to complement the lack of detection of facial features.

The next area of significance relates to the implementation of an Affective Tutoring System which tutors students in the domain of computer programming. Despite the recognized benefits of incorporating affect into tutoring systems, Affective Tutoring Systems are seldom implemented (Thompson & McGill, 2012), let alone one that tutors in computer programming. In this research, a pedagogical based model of responding to the students expressing negative affect centered on a context sensitive and comprehensive hint system was proposed and implemented. In addition, the architecture of the tutoring system also caters for future extension into other tutoring domains, with loose coupling between the various subsystems and pluggable modules within each subsystem.

Lastly, the effectiveness of the Affective Programming Tutoring System was also evaluated through an experimental study and a series of focus group discussions. The results of the experimental study showed that the students who were tutored with the full affective version of the APTS were able to attempt more exercises and were able to complete the exercises within a shorter period of time as compared to those tutored with the non-affective version, thus supporting the hypothesis that the full affective version of the Affective

Programming Tutoring System resulted in more effective tutoring as compared to the non-affective version. Analysis of the transcripts of the focus group discussion revealed that the students in general rated the usability and their acceptance of the tutoring system favorably. In particular, the filling in of missing lines of codes feature and the hint function of the system were highlighted as the top two useful functions of the system. The students' general consensus was that the filling in of missing lines of codes feature substantially reduces their cognitive load in the computer programming learning process and that the hints function offers them context sensitive help to overcome the programming hurdle.

8.4. Possible areas for future research

Keystrokes are idiosyncratic in nature but are readily available, non-intrusive and can be harvested from a cheaply available device i.e. keyboard. In this research, we extracted latency and duration features from keystrokes and established the viability of keystrokes for affect detection. However, the detection accuracy from keystrokes alone can be further enhanced. A suggestion would be to investigate into the semantics of the keys. Semantical analysis, which is not explored here, may offer further information into the deciphering the affective states of students. For example, a student may type the keyword "for" and then pause for a while, not knowing how to proceed at that point as he or she may not know the syntax of a for loop or whether a for loop is appropriate for that context. Exploiting the semantics and the timing sequences would help in this scenario.

Although most physiological sensors are obtrusive and difficult to set up, recent developments in wearable technologies are making it promising to use physiological sensors in affect detection. With advances in nanotechnology, analytics and wireless data communications, wearables will progress to be main stream sensor technologies that allow for continuous, reliable physiological monitoring with fuss free configuration and at an affordable cost. It is thus conceivable that in the foreseeable future, wearables will enable physiological sensing to make the transition from its current fragile use in a controlled laboratory environment to a more robust use in a naturalistic real-life environment.

Another potential area of research for extension is in the area of learning on mobile platforms. The use of mobile devices affords learning to take place anytime and anywhere and this is alluring to modern learners due to their hectic and globe-trotting lifestyles. Smart phones and tablets are being equipped with more and better sensing capabilities and manufacturers of these smart mobile devices are competing to pack these devices with ever faster processing chips that can crunch more data and at a faster rate. These developments have made possible the capturing of interaction patterns of these mobile users for affect sensing. Some issues, however remain to be resolved with regards to the use of mobile devices for affect sensing. The constraint of battery power is still a limiting factor and this also explains the use of a wide range of sensors in affect sensing studies in the literature with the exception of cameras and microphones (Rana, Hume, Reilly, Jurdak, & Soar, 2015); the cameras and microphones consume high levels of power that will fast deplete the battery. Nevertheless, this is still a promising area to monitor especially with the advent of power conserving processing capabilities and higher battery capacity technologies.

Bibliography

Adams, A., Mahmoud, M., Baltrusaitis, T., & Robinson, P. (2015). *Decoupling facial expressions and head motions in complex emotions*. Paper presented at the 6th International Conference on Affective Computing and Intelligent Interaction (ACII'15).

Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle *Selected papers of hirotugu akaike* (pp. 199-213): Springer.

Aleven, V., Stahl, E., Schworm, S., Fischer, F., & Wallace, R. (2003). Help seeking and help design in interactive learning environments. *Review of educational research, 73*(3), 277-320.

Alexander, S., Sarrafzadeh, A., & Hill, S. (2006). *Easy with Eve: a functional affective tutoring system*. Paper presented at the Workshop on Motivational and Affective Issues in ITS. 8th International Conference on ITS.

Alexander, S. T. V. (2008). An affect-sensitive intelligent tutoring system with an animated pedagogical agent that adapts to student emotion like a human tutor: a thesis presented in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science at Massey University, Albany, New Zealand.

AlZoubi, O., Calvo, R. A., & Stevens, R. H. (2009). Classification of EEG for affect recognition: an adaptive approach *AI 2009: Advances in Artificial Intelligence* (pp. 52-61): Springer.

Ambady, N., & Rosenthal, R. (1992). Thin slices of expressive behavior as predictors of interpersonal consequences: A meta-analysis. *Psychological bulletin, 111*(2), 256.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences, 4*(2), 167-207.

Anderson, J. R., & Skwarecki, E. (1986). The automated tutoring of introductory computer programming. *Communications of the ACM, 29*(9), 842-849.

Ang, J., Dhillon, R., Krupski, A., Shriberg, E., & Stolcke, A. (2002). *Prosody-based automatic detection of annoyance and frustration in human-computer dialog.* Paper presented at the INTERSPEECH.

ANTLR. (2017). from http://www.antlr.org/

Arlot, S., & Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics surveys, 4*, 40-79.

Arroyo, I., Cooper, D. G., Burleson, W., Woolf, B. P., Muldner, K., & Christopherson, R. (2009). *Emotion Sensors Go To School.* Paper presented at the AIED.

Ashcraft, M. H., & Kirk, E. P. (2001). The relationships among working memory, math anxiety, and performance. *Journal of experimental psychology: General, 130*(2), 224. doi: http://dx.doi.org/10.1037/0096-3445.130.2.224

Asteriadis, S., Tzouveli, P., Karpouzis, K., & Kollias, S. (2009). Estimation of behavioral user state based on eye gaze and head pose—application in an e-learning environment. *Multimedia Tools and Applications, 41*(3), 469-493.

Astin, A. W. (1975). *Preventing students from dropping out*: San Francisco: Jossey-Bass.

Astin, A. W. (1993). *What matters in college?: Four critical years revisited* (Vol. 1): Jossey-Bass San Francisco.

Austermann, A., Esau, N., Kleinjohann, L., & Kleinjohann, B. (2005). *Prosody based emotion recognition for MEXI.* Paper presented at the Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on.

Azevedo, R., Johnson, A., Chauncey, A., & Burkett, C. (2010). Self-regulated learning with MetaTutor: Advancing the science of learning with MetaCognitive tools *New science of learning* (pp. 225-247): Springer.

Barnes, T., Stamper, J. C., Lehmann, L., & Croy, M. J. (2008). *A pilot study on logic proof tutoring using hints generated from historical student data.* Paper presented at the EDM.

Barthet, M., Fazekas, G., & Sandler, M. (2012). Music emotion recognition: From content-to context-based models *From sounds to music and emotions* (pp. 228-252): Springer.

Bartlett, M., Littlewort, G., Wu, T., & Movellan, J. (2008). *Computer expression recognition toolbox.* Paper presented at the 8th IEEE International Conference on Automatic Face & Gesture Recognition.

Bartlett, M. S., Littlewort, G., Frank, M., Lainscsek, C., Fasel, I., & Movellan, J. (2005). *Recognizing facial expression: machine learning and application to spontaneous behavior.* Paper presented at the Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on.

Bartlett, M. S., Littlewort, G., Frank, M., Lainscsek, C., Fasel, I., & Movellan, J. (2006a). *Fully automatic facial action recognition in spontaneous behavior.* Paper presented at the Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on.

Bartlett, M. S., Littlewort, G. C., Frank, M. G., Lainscsek, C., Fasel, I. R., & Movellan, J. R. (2006b). Automatic recognition of facial actions in spontaneous expressions. *Journal of multimedia, 1*(6), 22-35.

Beal, C., Mitra, S., & Cohen, P. R. (2007). *Modeling learning patterns of students with a tutoring system using Hidden Markov Models*. Paper presented at the Proceedings of the 2007 conference on Artificial Intelligence in Education: Building Technology Rich Learning Contexts That Work.

Beilock, S. L., & Carr, T. H. (2005). When high-powered people fail working memory and “choking under pressure” in math. *Psychological Science, 16*(2), 101-105. doi: DOI: http://dx.doi.org/10.1037/e537052012-380

Bergadano, F., Gunetti, D., & Picardi, C. (2003). Identity verification through dynamic keystroke analysis. *Intelligent Data Analysis, 7*(5), 469-496.

Bernstein, N. A. (1967). The co-ordination and regulation of movements.

Bishop, C. (2007). Pattern Recognition and Machine Learning (Information Science and Statistics), 1st edn. 2006. corr. 2nd printing edn: Springer, New York.

Bixler, R., & D'Mello, S. (2013). *Detecting boredom and engagement during writing with keystroke analysis, task appraisals, and stable traits*. Paper presented at the 18th International Conference on Intelligent User Interfaces (IUI'13).

Blank, W. E., & Harwell, S. (1997). Promising Practices for Connecting High School to the Real World. 15-21.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher,* 4-16. doi: http://dx.doi.org/10.3102/0013189x013006004

Bosch, N., D'Mello, S. K., Baker, R. S., Ocumpaugh, J., Shute, V., Ventura, M., . . . Zhao, W. (2016). *Detecting Student Emotions in Computer-Enabled Classrooms.* Paper presented at the IJCAI.

Brandt, J., Guo, P. J., Lewenstein, J., Dontcheva, M., & Klemmer, S. R. (2009). *Two studies of opportunistic programming: interleaving web foraging, learning, and writing code.* Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Breiman, L. (2001). Random forests. *Machine learning, 45*(1), 5-32.

Brown, M., & Rogers, S. J. (1993). User identification via keystroke characteristics of typed names using neural networks. *International Journal of Man-Machine Studies, 39*(6), 999-1014.

Brown, N. C., & Altadmri, A. (2014). *Investigating novice programming mistakes: Educator beliefs vs. student data.* Paper presented at the Proceedings of the tenth annual conference on International computing education research.

Calvo, R. A., & D'Mello, S. (2010). Affect detection: An interdisciplinary review of models, methods, and their applications. *Affective Computing, IEEE Transactions on, 1*(1), 18-37.

Cannon, W. B. (1927). The James-Lange theory of emotions: A critical examination and an alternative theory. *The American journal of psychology, 39*(1/4), 106-124.

Cetintas, S., Si, L., Xin, Y. P., & Hord, C. (2010). Automatic detection of off-task behaviors in intelligent tutoring systems with machine learning techniques. *IEEE Transactions on Learning Technologies, 3*(3), 228-236. doi: http://dx.doi.org/10.1109/tlt.2009.44

Chambers, J. M. (1983). Comparing Data Distributions *Graphical Methods for Data Analysis* (pp. 60-62): Belmont, California: Wadsworth International Group.

Chen, C., Liaw, A., & Breiman, L. (2004). Using random forest to learn imbalanced data. Statistics Department of University of California at Berkeley: Berkeley. Technical Report 666.

Coates, H. (2010). Development of the Australasian survey of student engagement (AUSSE). *Higher Education, 60*(1), 1-17.

Conati, C., Gertner, A. S., VanLehn, K., & Druzdzel, M. J. (1997). *On-line student modeling for coached problem solving using Bayesian networks.* Paper presented at the User Modeling.

Conati, C., & Maclaren, H. (2009). Empirically building and evaluating a probabilistic model of user affect. *User Modeling and User-Adapted Interaction, 19*(3), 267-303.

Corbett, A. T., & Anderson, J. R. (2001). *Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes.* Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.

Coulson, M. (2004). Attributing emotion to static body postures: Recognition accuracy, confusions, and viewpoint dependence. *Journal of nonverbal behavior, 28*(2), 117-139. doi: http://dx.doi.org/10.1023/b:jonb.0000023655.25550.be

D'Mello, S., & Graesser, A. (2009). Automatic detection of learner's affect from gross body language. *Applied Artificial Intelligence, 23*(2), 123-150.

D'Mello, S. K., & Graesser, A. (2010). Multimodal semi-automated affect detection from conversational cues, gross body language, and facial features. *User Modelling and User-Adapted Interaction, 20*(2), 147-187.

D'Mello, S., Jackson, T., Craig, S., Morgan, B., Chipman, P., White, H., . . . Picard, R. (2008). *AutoTutor detects and responds to learners affective and cognitive states.* Paper presented at the Workshop on emotional and cognitive issues at the international conference on intelligent tutoring systems.

D'Ulizia, A. (2009). Exploring Multimodal Input Fusion Strategies. *The Handbook of Research on Multimodal Human Computer Interaction and Pervasive Services: Evolutionary Techniques for Improving Accessibility*, 34-57. doi: http://dx.doi.org/10.4018/978-1-60566-386-9.ch003

Damasio, A. (1999). *The feeling of what happens*: New York: Harcourt, Brace.

Damasio, A. R. (1994). Descartes' error: Emotion, rationality and the human brain.

Darwin, C. (1969). *The Expression of the Emotions in Man and Animals: London, Murray, 1872*: Culture et civilisation.

De Vicente, A., & Pain, H. (2002). *Informing the detection of the students' motivational state: an empirical study.* Paper presented at the Intelligent tutoring systems.

DeLong, E. R., DeLong, D. M., & Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, 837-845. doi: http://dx.doi.org/10.2307/2531595

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1-38.

Dev, P. C. (1997). Intrinsic Motivation and Academic Achievement What Does Their Relationship Imply for the Classroom Teacher? *Remedial and Special Education, 18*(1), 12-19.

Dowland, P., Furnell, S., & Papadaki, M. (2002). Keystroke analysis as a method of advanced user authentication and response *Security in the Information Society* (pp. 215-226): Springer.

Dowland, P. S., & Furnell, S. M. (2004). A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies *Security and Protection in Information Processing Systems* (pp. 275-289): Springer.

Drools - Drools - Business Rules Management System. (2017). from http://www.drools.org/

Eckerdal, A., McCartney, R., Moström, J. E., Sanders, K., Thomas, L., & Zander, C. (2007). *From Limen to Lumen: computing students in liminal spaces*. Paper presented at the Proceedings of the third international workshop on Computing education research.

Efklides, A. (2006). Metacognition and affect: What can metacognitive experiences tell us about the learning process? *Educational Research Review, 1*(1), 3-14.

Ekman, P. (1984). Expression and the nature of emotion. *Approaches to emotion, 3*, 19-344.

Ekman, P. (1992). An argument for basic emotions. *Cognition & emotion, 6*(3-4), 169-200.

Ekman, P., & Friesen, W. (1978). The facial action coding system: A technique for the measurement of facial movement: Palo Alto, CA: Consulting Psychologists Press.

Ekman, P., & Friesen, W. V. (1969). Nonverbal leakage and clues to deception. *Psychiatry, 32*(1), 88-106.

Ekman, P., & Friesen, W. V. (2003). *Unmasking the face: A guide to recognizing emotions from facial clues*: Ishk.

Ekman, P., Friesen, W. V., O'Sullivan, M., Chan, A., Diacoyanni-Tarlatzis, I., Heider, K., . . . Ricci-Bitti, P. E. (1987). Universals and cultural differences in the judgments of facial expressions of emotion. *Journal of personality and social psychology, 53*(4), 712.

Epp, C., Lippold, M., & Mandryk, R. L. (2011). *Identifying emotional states using keystroke dynamics*. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters, 27*(8), 861-874. doi: http://dx.doi.org/10.1016/j.patrec.2005.10.010

Forgy, C. L. (1979). *On the efficient implementation of production systems*. Carnegie-Mellon University.

Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE, 61*(3), 268-278.

The Future of Jobs. (2017). from http://wef.ch/22CnI5C

Graesser, A., McDaniel, B., Chipman, P., Witherspoon, A., D'Mello, S., & Gholson, B. (2006). *Detection of emotions during learning with AutoTutor*. Paper presented at the Proceedings of the 28th Annual Meetings of the Cognitive Science Society.

Graesser, A. C., Chipman, P., Haynes, B. C., & Olney, A. (2005). AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *Education, IEEE Transactions on, 48*(4), 612-618.

Grafsgaard, J. F., Wiggins, J. B., Boyer, K. E., Wiebe, E. N., & Lester, J. C. (2013). *Automatically Recognizing Facial Expression: Predicting Engagement and Frustration*. Paper presented at the 6th International Conference on Educational Data Mining.

Greene, T. R., & Noice, H. (1988). Influence of positive affect upon creative thinking and problem solving in children. *Psychological reports, 63*(3), 895-898. doi: http://dx.doi.org/10.2466/pr0.1988.63.3.895

Hassan, M. R., & Nath, B. (2005). *Stock market forecasting using hidden Markov model: a new approach.* Paper presented at the 5th International Conference on Intelligent Systems Design and Applications (ISDA'05).

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: New York: Springer.

Heckerman, D. (2008). A tutorial on learning with Bayesian networks *Innovations in Bayesian networks* (pp. 33-82): Springer.

Hidden Markov Model (HMM) Toolbox for Matlab. (2016). from http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html

Hoeffding, W. (1948). A Class of Statistics with Asymptotically Normal Distribution. 293-325. doi: http://dx.doi.org/10.1007/978-1-4612-0919-5_20

Hume, G., Michael, J., Rovick, A., & Evens, M. (1996). Hinting as a tactic in one-on-one tutoring. *The Journal of the Learning Sciences, 5*(1), 23-47.

Isen, A. M. (2000). Some perspectives on positive affect and self-regulation. *Psychological Inquiry, 11*(3), 184-187.

Isen, A. M., Daubman, K. A., & Nowicki, G. P. (1987). Positive affect facilitates creative problem solving. *Journal of personality and social psychology, 52*(6), 1122.

Jadud, M. C. (2006). *Methods and tools for exploring novice compilation behaviour.* Paper presented at the Proceedings of the second international workshop on Computing education research.

Jaimes, A., & Sebe, N. (2007). Multimodal human-computer interaction: A survey. *Computer Vision and Image Understanding, 108*(1-2), 116-134.

James, W. (1884). II.—What is an emotion? *Mind*(34), 188-205.

Jaques, N., Conati, C., Harley, J. M., & Azevedo, R. (2014). *Predicting affect from gaze data during interaction with an intelligent tutoring system.* Paper presented at the International Conference on Intelligent Tutoring Systems.

Ji, Q., Lan, P., & Looney, C. (2006). A probabilistic framework for modeling and real-time monitoring human fatigue. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 36*(5), 862-875.

John, G. H., & Langley, P. (1995). *Estimating continuous distributions in Bayesian classifiers.* Paper presented at the Proceedings of the Eleventh conference on Uncertainty in artificial intelligence.

Johnson, W. L. (1990). Understanding and debugging novice programs. *Artificial Intelligence, 42*(1), 51-97.

Jordan, M. I. (1999). *Learning in Graphical Models* (Vol. 89). Cambridge, MA: MIT Press.

Joyce, R., & Gupta, G. (1990). Identity authentication based on keystroke latencies. *Communications of the ACM, 33*(2), 168-176.

Juslin, P. N., & Scherer, K. R. (2005). Vocal expression of affect. *The new handbook of methods in nonverbal behavior research*, 65-135.

Kaklauskas, A., Kuzminske, A., Zavadskas, E. K., Daniunas, A., Kaklauskas, G., Seniut, M., . . . Juozapaitis, A. (2015). Affective tutoring system for built environment management. *Computers & Education, 82*, 202-216.

Kanade, T., Cohn, J. F., & Tian, Y. (2000). *Comprehensive database for facial expression analysis.* Paper presented at the Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on.

Kapoor, A., Burleson, W., & Picard, R. W. (2007). Automatic prediction of frustration. *International Journal of Human-Computer Studies, 65*(8), 724-736. doi: http://dx.doi.org/10.1016/j.ijhcs.2007.02.003

Kapoor, A., & Picard, R. W. (2005). *Multimodal affect recognition in learning environments.* Paper presented at the Proceedings of the 13th annual ACM international conference on Multimedia.

Keller, J. M. (1987). Development and use of the ARCS model of instructional design. *Journal of instructional development, 10*(3), 2-10.

Khan, I. A., Hierons, R. M., & Brinkman, W. P. (2007). *Mood independent programming.* Paper presented at the Proceedings of the 14th European conference on Cognitive ergonomics: invent! explore!

Kim, J., & André, E. (2008). Emotion recognition based on physiological changes in music listening. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 30*(12), 2067-2083.

Konar, A., & Chakraborty, A. (2014). *Emotion recognition: A pattern analysis approach*: John Wiley & Sons.

Kononenko, I. (1994). *Estimating attributes: analysis and extensions of RELIEF.* Paper presented at the European conference on machine learning.

Kort, B., Reilly, R., & Picard, R. W. (2001). *An affective model of interplay between emotions and learning: Reengineering educational pedagogy-building a learning companion.* Paper presented at the Advanced Learning Technologies, IEEE International Conference on.

Kotsiantis, S., & Kanellopoulos, D. (2006). Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering, 32*(1), 47-58.

Kuh, G. D. (2001). Assessing what really matters to student learning inside the national survey of student engagement. *Change: The Magazine of Higher Learning, 33*(3), 10-17.

Kuh, G. D., Hu, S., & Vesper, N. (2000). "They shall be known by what they do": An activities-based typology of college students. *Journal of College Student Development, 41*(2), 228-244.

Kuncheva, L. I., Bezdek, J. C., & Duin, R. P. (2001). Decision templates for multiple classifier fusion: an experimental comparison. *Pattern recognition, 34*(2), 299-314. doi: http://dx.doi.org/10.1016/s0031-3203(99)00223-x

Kurgan, L., & Cios, K. J. (2004). CAIM discretization algorithm. *Knowledge and Data Engineering, IEEE Transactions on, 16*(2), 145-153.

Kwon, O.-W., Chan, K., Hao, J., & Lee, T.-W. (2003). *Emotion recognition by speech signals.* Paper presented at the INTERSPEECH.

Land, R., Cousin, G., Meyer, J. H., & Davies, P. (2006). Implications of threshold concepts for course design and evaluation. *Overcoming barriers to student understanding threshold concepts and troublesome knowledge*, 195-206.

Lange, C. G., & James, W. (1922). *The emotions* (Vol. 1): Williams & Wilkins.

Lazarus, R. S. (1991). Progress on a cognitive-motivational-relational theory of emotion. *American psychologist, 46*(8), 819.

Le, N.-T., & Menzel, W. (2009). Using Weighted Constraints to Diagnose Errors in Logic Programming–The Case of an Ill-defined Domain. *International Journal of Artificial Intelligence in Education, 19*(4), 381-400.

Levenshtein, V. I. (1966). *Binary codes capable of correcting deletions, insertions, and reversals*. Paper presented at the Soviet physics doklady.

Litman, D. J., & Forbes-Riley, K. (2004). *Predicting student emotions in computer-human tutoring dialogues*. Paper presented at the Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics.

Littlewort, G., Whitehill, J., Wu, T., Fasel, I., Frank, M., Movellan, J., & Bartlett, M. (2011). *The computer expression recognition toolbox (CERT)*. Paper presented at the Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on.

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 50-60.

Marsland, S. (2015). *Machine learning: an algorithmic perspective*. Boca Raton, FL: CRC press.

McDaniel, B., D'Mello, S., King, B., Chipman, P., Tapp, K., & Graesser, A. (2007). *Facial features for affective state detection in learning environments*. Paper presented at the 29th Annual Cognitive Science Society.

Mcquiggan, S. W., Lee, S., & Lester, J. C. (2007). Early prediction of student frustration *Affective Computing and Intelligent Interaction* (pp. 698-709): Springer.

Meyer, J., & Land, R. (2003). *Threshold concepts and troublesome knowledge: Linkages to ways of thinking and practising within the disciplines*: University of Edinburgh Edinburgh.

Montepare, J., Koff, E., Zaitchik, D., & Albert, M. (1999). The use of body movements and gestures as cues to emotions in younger and older adults. *Journal of nonverbal behavior, 23*(2), 133-152.

Morgan, D. L. (1996a). Focus groups. *Annual review of sociology, 22*(1), 129-152.

Morgan, D. L. (1996b). *Focus groups as qualitative research* (Vol. 16): Sage publications.

Mota, S., & Picard, R. W. (2003). *Automated posture analysis for detecting learner's interest level*. Paper presented at the Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'03).

Murphy, K. (2001). The bayes net toolbox for matlab. *Computing science and statistics, 33*(2), 1024-1034.

Nasoz, F., Alvarez, K., Lisetti, C. L., & Finkelstein, N. (2004). Emotion recognition from physiological signals using wireless sensors for presence technologies. *Cognition, Technology & Work, 6*(1), 4-14.

The National Student Survey. (2017). from http://www.thestudentsurvey.com/about.php

Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine learning, 39*(2-3), 103-134.

Oehlert, G. W. (2010). *A first course in design and analysis of experiments*. New York, USA: WH Freeman.

Ohlsson, S. (1992). Constraint-based student modelling. *Journal of Interactive Learning Research, 3*(4), 429.

Öhman, A., & Soares, J. J. (1994). " Unconscious anxiety": phobic responses to masked stimuli. *Journal of abnormal psychology, 103*(2), 231.

Pearl, J. (1988). Probabilistic reasoning in intelligent systems: Networks of plausible reasoning: Morgan Kaufmann Publishers, Los Altos.

Pekrun, R., Goetz, T., Titz, W., & Perry, R. P. (2002). Academic emotions in students' self-regulated learning and achievement: A program of qualitative and quantitative research. *Educational psychologist, 37*(2), 91-105.

Peng, C.-Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An introduction to logistic regression analysis and reporting. *The Journal of Educational Research, 96*(1), 3-14.

Picard, R. W. (1995). Affective computing.

Picard, R. W. (1997). *Affective computing* (Vol. 252): MIT press Cambridge.

Picard, R. W. (2000). *Affective computing*: MIT press.

Picard, R. W., Vyzas, E., & Healey, J. (2001). Toward machine emotional intelligence: Analysis of affective physiological state. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 23*(10), 1175-1191.

Prendinger, H., Dohi, H., Wang, H., Mayer, S., & Ishizuka, M. (2004). Empathic embodied interfaces: Addressing users' affective state *Affective Dialogue Systems* (pp. 53-64): Springer.

Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. *ieee assp magazine, 3*(1), 4-16.

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

Radloff, A., & Coates, H. (2010). Doing more for learning: enhancing engagement and outcomes: Australasian Survey of Student Engagement: Australasian Student Engagement Report.

Rana, R., Hume, M., Reilly, J., Jurdak, R., & Soar, J. (2015). Opportunistic and context-aware affect sensing on smartphones: the concept, challenges and opportunities. *arXiv preprint arXiv:1502.02796*.

Rivers, K., & Koedinger, K. R. (2013). *Automatic generation of programming feedback: A data-driven approach.* Paper presented at the The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013).

Rodrigo, M. M. T., & Baker, R. S. (2009). *Coarse-grained detection of student frustration in an introductory programming course*. Paper presented at the Proceedings of the fifth international workshop on Computing education research workshop.

Roscoe, R. D., Allen, L. K., Weston, J. L., Crossley, S. A., & McNamara, D. S. (2014). The Writing Pal intelligent tutoring system: Usability testing and development. *Computers and Composition, 34*, 39-59.

Rose, R. C., & Reynolds, D. A. (1990). *Text independent speaker identification using automatic acoustic segmentation.* Paper presented at the Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on.

Rosenberg, E. L., & Ekman, P. (1994). Coherence between expressive and experiential systems in emotion. *Cognition & emotion, 8*(3), 201-229.

Rowe, J., Mott, B., McQuiggan, S., Robison, J., Lee, S., & Lester, J. (2009). *Crystal island: A narrative-centered learning environment for eighth grade microbiology*. Paper presented at the workshop on intelligent educational games at the 14th international conference on artificial intelligence in education, Brighton, UK.

Russell, J. A. (2003). Core affect and the psychological construction of emotion. *Psychological review, 110*(1), 145.

Russell, J. A., & Pratt, G. (1980). A description of the affective quality attributed to environments. *Journal of personality and social psychology, 38*(2), 311.

Sabourin, J. L., Rowe, J. P., Mott, B. W., & Lester, J. C. (2013). Considering alternate futures to classify off-task behavior as emotion self-regulation: A supervised learning approach. *JEDM-Journal of Educational Data Mining, 5*(1), 9-38.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). *A Bayesian approach to filtering junk e-mail.* Paper presented at the Learning for Text Categorization: Papers from the 1998 workshop.

Scherer, K. R. (1984). Emotion as a multicomponent process: A model and some cross-cultural data. *Review of Personality & Social Psychology*.

Scherer, K. R., & Ellgring, H. (2007). Multimodal expression of emotion: Affect programs or componential appraisal patterns? *Emotion, 7*(1), 158.

Schuller, B., Lang, M., & Rigoll, G. (2002). Automatic emotion recognition by the speech signal. *Institute for Human-Machine-Communication, Technical University of Munich, 80290*.

Schuller, B., Villar, R. J., Rigoll, G., & Lang, M. K. (2005). *Meta-Classifiers in Acoustic and Linguistic Feature Fusion-Based Affect Recognition.* Paper presented at the ICASSP (1).

Sebe, N., Lew, M. S., Sun, Y., Cohen, I., Gevers, T., & Huang, T. S. (2007). Authentic facial expression analysis. *Image and Vision Computing, 25*(12), 1856-1863.

Shaikh, M. A. M., Prendinger, H., & Ishizuka, M. (2008). Sentiment assessment of text by analyzing linguistic features and contextual valence assignment. *Applied Artificial Intelligence, 22*(6), 558-601.

Shih, B., Koedinger, K. R., & Scheines, R. (2010). *Discovery of Student Strategies using Hidden Markov Model Clustering.* Paper presented at the the Proceedings of the 6th International Conference on Educational Data Mining.

Slavin, R. E., & Davis, N. (2006). Educational psychology: Theory and practice.

*Sphere engine*2015). Vol. 2015. Retrieved from sphere-engine.com

Stiefelhagen, R., & Zhu, J. (2002). *Head orientation and gaze direction in meetings.* Paper presented at the CHI'02 Extended Abstracts on Human Factors in Computing Systems.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive science, 12*(2), 257-285.

techinasia. (2014). Singapore plans to introduce programming lessons in public schools to boost the economy.

Teeters, A., El Kaliouby, R., & Picard, R. (2006). *Self-Cam: feedback from what would be your social partner.* Paper presented at the ACM SIGGRAPH 2006 Research posters.

Thompson, N., & McGill, T. J. (2012). Affective tutoring systems: enhancing e-learning with the emotional awareness of a human tutor. *International Journal of Information and Communication Technology Education, 8*(4), 75-89.

Thompson, N., & McGill, T. J. (2017). Genetics with Jean: the design, development and evaluation of an affective tutoring system. *Educational Technology Research and Development, 65*(2), 279-299.

Towns, S., FitzGerald, P., & Lester, J. (1998). *Visual emotive communication in lifelike pedagogical agents.* Paper presented at the Intelligent Tutoring Systems.

Vee, M., Meyer, B., & Mannock, K. L. (2006). *Understanding novice errors and error paths in object-oriented programming through log analysis.* Paper presented at the Proceedings of workshop on educational data mining at the 8th international conference on intelligent tutoring systems (ITS 2006).

Villon, O., & Lisetti, C. (2006). *A user-modeling approach to build user's psycho-physiological maps of emotions using bio-sensors.* Paper presented at the Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on.

Vizer, L. M., Zhou, L., & Sears, A. (2009). Automated stress detection using keystroke and linguistic features: An exploratory study. *International Journal of Human Computer Studies, 67*(10), 870-886. doi: http://dx.doi.org/10.1016/j.ijhcs.2009.07.005

Vural, E., Çetin, M., Erçil, A., Littlewort, G., Bartlett, M., & Movellan, J. (2008). Automated drowsiness detection for improved driving safety.

Vygotsky, L. (1978). Interaction between learning and development. *Readings on the development of children, 23*(3), 34-41.

Weiner, B. (1972). Attribution theory, achievement motivation, and the educational process. *Review of educational research, 42*(2), 203-215.

Weiten, W., Dunn, D., & Hammer, E. (2011). *Psychology applied to modern life: Adjustment in the 21st century* (10th ed.): Cengage Learning.

Weragama, D., & Reye, J. (2014). Analysing student programs in the PHP intelligent tutoring system. *International Journal of Artificial Intelligence in Education, 24*(2), 162-188.

WHSmith. (2017). Changes to the National Curriculum in England 2015.

Wood, H., & Wood, D. (1999). Help seeking, learning and contingent tutoring. *Computers & Education, 33*(2), 153-169.

Woolf, B., Burleson, W., Arroyo, I., Dragon, T., Cooper, D., & Picard, R. (2009). Affect-aware tutors: recognising and responding to student affect. *International Journal of Learning Technology, 4*(3-4), 129-164.

Woolf, B. P., Arroyo, I., Muldner, K., Burleson, W., Cooper, D. G., Dolan, R., & Christopherson, R. M. (2010). *The effect of motivational learning companions on low achieving students and students with disabilities.* Paper presented at the International Conference on Intelligent Tutoring Systems.

xLabs eye, gaze and head tracking via webcam. (2016). from https://xlabsgaze.com/

Xu, S., & Chee, Y. S. (2003). Transformation-based diagnosis of student programs for programming tutoring systems. *Software Engineering, IEEE Transactions on, 29*(4), 360-384.

Yamato, J., Ohya, J., & Ishii, K. (1992). *Recognizing human action in time-sequential images using hidden markov model.* Paper presented at the Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on.

Yeasin, M., Bullot, B., & Sharma, R. (2006). Recognition of facial expressions and measurement of levels of interest from video. *IEEE Transactions on Multimedia, 8*(3), 500-508. doi: http://dx.doi.org/10.1109/tmm.2006.870737

Yerkes, R. M., & Dodson, J. D. (1908). The relation of strength of stimulus to rapidity of habit-formation. *Journal of comparative neurology and psychology, 18*(5), 459-482.

Zajonc, R. B. (1980). Feeling and thinking: Preferences need no inferences. *American psychologist, 35*(2), 151.

Zeng, Z., Pantic, M., Roisman, G. I., & Huang, T. S. (2007a). *A survey of affect recognition methods: Audio, visual and spontaneous expressions.* Paper presented at the 9th International Conference on Multimodal Interfaces, ICMI'07.

Zeng, Z., Pantic, M., Roisman, G. I., & Huang, T. S. (2007b). *A survey of affect recognition methods: Audio, visual and spontaneous expressions*. Paper presented at the Proceedings of the 9th International Conference on Multimodal Interfaces, ICMI'07.

Zimmermann, P., Guttormsen, S., Danuser, B., & Gomez, P. (2003). Affective computing--a rationale for measuring mood with mouse and keyboard. *International journal of occupational safety and ergonomics*, *9*(4), 539-551.