Eye-Tracking as a Source of Information for Automatically Predicting User Learning with MetaTutor, an Intelligent Tutoring System to Support Self-Regulated Learning

by

Daria Bondareva

B.Sc., The M. Ye. Zhukovskyy National Aerospace University “KhAI”, 2009
M.Sc., The M. Ye. Zhukovskyy National Aerospace University “KhAI”, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Computer Science)

The University of British Columbia

(Vancouver)

March 2014
© Daria Bondareva, 2014
Abstract

Student modeling has been gaining interest among researchers recently. A lot of work has been done on exploring value of interface actions on predicting learning.

The focus of this thesis is on using eye-tracking data and action logs for building classifies to infer a student’s learning performance during interaction with MetaTutor, an Intelligent Tutoring System (ITS) that scaffolds self-regulated learning (SRL). Research has shown that eye tracking can be a valuable source for predicting learning for certain learning environments. In this thesis we extend these results by showing that modeling based on eye-tracking data is a valuable approach to predicting learning for another type of ITS, a hypermedia learning environment.

We use data from 50 students (collected by a research team at McGill University, which also designed MetaTutor) to compare the performance of actions and eye-tracking data (1) after a complete interaction, and (2) during interaction when different amounts of gaze and action data are available. We built several classifiers using common machine learning algorithms and techniques, with feature sets that are based on (1) eye-tracking data only, (2) actions data only and (3) eye-tracking and actions data combined. The results we found show that eye-tracking data brings important information in predicting student’s performance for an ITS supporting SRL in both overall and over time analysis. The features used for training classifiers suggest that usage of SRL tools available in MetaTutor can be a good predictor of learning.
Preface

This work is a part of a larger project conducted by the S.M.A.R.T. research group at McGill University.

The dataset was shared by the collaborators from McGill University. I did not participate in the study design or data collection (Chapter 3). MetaTutor, the test-bed used in this thesis, was also provided by the S.M.A.R.T. lab.

The eye-tracking data (Chapter 4) was processed using the EMDAT (Eye Movement Data Analysis Toolkit) developed by the Intelligent User Interfaces research group at the University of British Columbia. I updated the original data processing so it corresponds to the dynamic nature of MetaTutor and made appropriate changes to the EMDAT.

I used the python scripts package MTLogAnalyzer to process interaction logs (Chapter 5), which was originally developed by the collaborators from the S.M.A.R.T. group at McGill University. I extended this package with additional functionalities (e.g., calculating extended list of features, calculating features for online learning).

All machine learning experiments and analysis of the results (Chapters 7, 8) were performed by me.

A version of the research described in Chapter 7 has been published as Bondareva, D., Conati, C., Feyzi-Behnagh, R., Harley, J., Azevedo R., Bouchet, F. (2013). Inferring Learning from Gaze Data during Interaction with an Environment to Support Self-Regulated Learning. In Proceedings of AIED 2013, 16th International Conference on AI in Education, Springer. The co-authors from McGill University contributed to the paper by providing the raw eye-tracking and interaction data and helping to understand it at early stages of the project.
# Table of Contents

Abstract........................................................................................................................................... ii
Preface ................................................................................................................................................ iii
Table of Contents.......................................................................................................................... iv
List of Tables ....................................................................................................................................... viii
List of Figures ..................................................................................................................................... viii
List of Abbreviations....................................................................................................................... ix
Acknowledgements........................................................................................................................ x
Dedication........................................................................................................................................... xi
Chapter 1  Introduction..................................................................................................................... 1
  1.1  Thesis goals and approach ................................................................................................. 2
  1.2  Contributions of the work .............................................................................................. 3
  1.3  Outline................................................................................................................................... 4
Chapter 2  Related Work ................................................................................................................. 6
  2.1  User modeling ................................................................................................................... 6
  2.2  Eye-tracking in user modeling.......................................................................................... 7
  2.3  Student modeling ............................................................................................................. 8
  2.4  Assessing students’ cognitive states with eye-tracking ..................................................... 9
      2.4.1  Offline analysis of cognitive processes with eye-tracking data ................................. 9
      2.4.2  Real-time assessment of cognitive states using eye-tracking .................................. 10
  2.5  Predicting students’ affective states with eye-tracking ..................................................... 12
  2.6  Predicting students’ meta-cognitive states with eye-tracking ........................................... 13
Chapter 3  MetaTutor and Self-Regulated Learning ..................................................................... 14
  3.1  Self-regulated learning .................................................................................................... 14
  3.2  Overview of the environment ......................................................................................... 17
  3.3  Description of the study to collect behavioral data during interaction with
      MetaTutor.............................................................................................................................. 23
Chapter 4  Preparing and Processing the Eye-Tracking Data...................................................... 25
  4.1  Preparation of raw gaze data ........................................................................................... 25
      4.1.1  Pre-processing of gaze data ................................................................................... 26
      4.1.2  Gaze data validation ............................................................................................. 28
  4.2  Eye-tracking features.......................................................................................................... 31
Chapter 5  Preparing and Processing the Action Data............................................................... 35
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Parsing logs and features calculation</td>
<td>35</td>
</tr>
<tr>
<td>5.2 Actions features</td>
<td>37</td>
</tr>
<tr>
<td>5.2.1 General features of working with MetaTutor</td>
<td>38</td>
</tr>
<tr>
<td>5.2.2 Features related to learning goal management</td>
<td>39</td>
</tr>
<tr>
<td>5.2.3 Subgoal-related features for working with content</td>
<td>41</td>
</tr>
<tr>
<td>5.2.4 Features describing the note-taking</td>
<td>43</td>
</tr>
<tr>
<td>5.2.5 Features for SRL tools usage</td>
<td>44</td>
</tr>
<tr>
<td>6.1 Machine learning algorithms</td>
<td>46</td>
</tr>
<tr>
<td>6.2 Feature selection</td>
<td>47</td>
</tr>
<tr>
<td>6.3 Cross-validation</td>
<td>49</td>
</tr>
<tr>
<td>7.1 Dataset and class labels</td>
<td>51</td>
</tr>
<tr>
<td>7.2 Data preparation</td>
<td>53</td>
</tr>
<tr>
<td>7.3 Discussion of results</td>
<td>54</td>
</tr>
<tr>
<td>7.4 Analysis of eye-tracking features</td>
<td>57</td>
</tr>
<tr>
<td>8.1 Feature sets</td>
<td>60</td>
</tr>
<tr>
<td>8.2 Overall performance of the basic models</td>
<td>61</td>
</tr>
<tr>
<td>8.2.1 Interaction effect</td>
<td>65</td>
</tr>
<tr>
<td>8.3 Overall performance of ensemble models</td>
<td>68</td>
</tr>
<tr>
<td>8.3.1 Action-based ensemble model</td>
<td>73</td>
</tr>
<tr>
<td>8.3.2 Conclusions on ensemble modeling</td>
<td>73</td>
</tr>
<tr>
<td>8.4 Accuracy over time</td>
<td>74</td>
</tr>
<tr>
<td>8.4.1 Performance of Simple Logistic Regression on Gaze data over time</td>
<td>76</td>
</tr>
<tr>
<td>8.4.2 Performance of Multilayer Perceptron on Actions data over time</td>
<td>77</td>
</tr>
<tr>
<td>8.4.3 Performance of Simple Logistic Regression on the Full data over time</td>
<td>81</td>
</tr>
<tr>
<td>8.4.4 Performance of Multilayer Perceptron on Full data over time</td>
<td>84</td>
</tr>
<tr>
<td>8.4.5 Performance of Ensemble classifier over time</td>
<td>85</td>
</tr>
<tr>
<td>8.5 Discussion of results</td>
<td>89</td>
</tr>
<tr>
<td>9.1 Thesis goals satisfaction</td>
<td>91</td>
</tr>
<tr>
<td>9.1.1 Research question 1: Can eye-tracking data be used to assess learning performance of a student interacting with MetaTutor?</td>
<td>92</td>
</tr>
<tr>
<td>9.1.2 Research question 2: How well does eye-tracking perform in predicting student learning?</td>
<td>93</td>
</tr>
</tbody>
</table>
9.1.3 Research question 3: Which elements of the interface contribute most to assessing learning? ...........................................................................................................94

9.2 Limitations and future work ..............................................................................................................................................................................................95

Bibliography .................................................................................................................................................................................................................97
List of Tables

Table 4.1: Description of gaze-based features .................................................................33
Table 5.1: Working with content and durations .................................................................38
Table 5.2: Learning goals management ............................................................................40
Table 5.3: Subgoal-related features for working with content ..........................................41
Table 5.4: Taking notes actions .......................................................................................43
Table 5.5: List of tools available from the SRL palette in MetaTutor ..................................44
Table 7.1: Full dataset class descriptive statistics .............................................................52
Table 7.2: Accuracy and Kappa scores for different classifiers and feature sets ..................55
Table 7.3: Selected features for Simple Logistic Regression trained on Gaze ....................57
Table 7.4: Selected features for Simple Logistic Regression on Gaze .................................58
Table 8.1: Overall accuracies and Kappa scores for base models ......................................62
Table 8.2: Overall accuracies and Kappa scores for ensemble classifiers .........................69
Table 8.3: Prediction accuracy of ensemble models combined by feature set ....................73
Table 8.4: Accuracies and Kappa scores over time for selected models .............................75
Table 8.5: Selected features for Multilayer Perceptron trained on Actions .........................79
Table 8.6: Selected features for Simple Logistic Regression trained on Full .......................82
Table 8.7: Features Multilayer Perceptron trained on Full ...............................................85
### List of Figures

| Figure 3.1: Sample MetaTutor interface | ..........................................................17 |
| Figure 3.2: Normal layout with image icon | .....................................................................18 |
| Figure 3.3: Input layout | ...........................................................................19 |
| Figure 3.4: Setting subgoal dialog history | ....................................................................20 |
| Figure 3.5: Embedded notepad interface | .......................................................................21 |
| Figure 3.6: Full view layout in MetaTutor | .......................................................................22 |
| Figure 3.7: Main session timeline | ...........................................................................24 |
| Figure 4.1: Validation process | ............................................................................28 |
| Figure 4.2: Percentage of valid participants with different validity thresholds | ................................29 |
| Figure 4.3: Percentage of valid participants with different thresholds for auto-partition | ........30 |
| Figure 4.4: Gaze-based measures | ..........................................................................31 |
| Figure 4.5: Sample MetaTutor interface with AOIs highlighted | ...........................................33 |
| Figure 5.1: A sample of a MetaTutor log file | ...........................................................36 |
| Figure 6.1: Predicting mechanism in the ensemble model scheme | ...........................................47 |
| Figure 7.1: Overall accuracy of the 5 best performing algorithms over 3 gaze feature sets | ........56 |
| Figure 8.1: Overall accuracy of the 5 algorithms over 3 feature sets | ................................63 |
| Figure 8.2: Main effect of feature set | ........................................................................64 |
| Figure 8.3: Main effect of learning algorithm | ............................................................65 |
| Figure 8.4: Balance in predicting LL and HL of the three best performing classifiers | ..........66 |
| Figure 8.5: Overall performance of the best ensemble models | ...........................................70 |
| Figure 8.6: Average overall accuracies of best performing models | ...........................................71 |
| Figure 8.7: Comparison of performance for best performing base models and ensemble models | ............................................................72 |
| Figure 8.8: Average accuracy over time for selected classifiers | ............................................75 |
| Figure 8.9: Accuracy over time of Simple Logistic Regression on Gaze | ................................77 |
| Figure 8.10: Accuracy of Multilayer Perceptron on Actions over time | ................................78 |
| Figure 8.11: Accuracy over time of Simple Logistic Regression on Full | ...................................81 |
| Figure 8.12: Average performance of simple Logistic Regression over time on Gaze and Actions feature sets | ............................................................83 |
| Figure 8.13: Online accuracy of Multilayer Perceptron on Full feature set | ................................84 |
| Figure 8.14: Online learning accuracy of Ensemble model | ...................................................86 |
| Figure 8.15: Ensemble model with the base classifiers | .......................................................88 |
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOI</td>
<td>area of Interest</td>
</tr>
<tr>
<td>CAM</td>
<td>cognitive, affective, metacognitive</td>
</tr>
<tr>
<td>CSP</td>
<td>constraint satisfaction problem</td>
</tr>
<tr>
<td>EMDAT</td>
<td>eye movement data analysis toolkit</td>
</tr>
<tr>
<td>HL</td>
<td>high learners</td>
</tr>
<tr>
<td>ITS</td>
<td>intelligent tutoring system</td>
</tr>
<tr>
<td>LL</td>
<td>low learners</td>
</tr>
<tr>
<td>PA</td>
<td>pedagogical agent</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>PLG</td>
<td>proportional learning gain</td>
</tr>
<tr>
<td>SD</td>
<td>standard deviation</td>
</tr>
<tr>
<td>SRL</td>
<td>self-regulated learning</td>
</tr>
</tbody>
</table>
Acknowledgements

Foremost, I want to thank my supervisor, Prof. Cristina Conati, who patiently guided me along the way. Her encouragement, suggestions and feedback were always helpful and greatly appreciated.

I’m also grateful to my collaborators from the S.M.A.R.T. Lab, Prof. Roger Azevedo, Dr. François Bouchet, Reza Feyzi Behnagh and Jason Harley who were always ready to help me with every question I had in the beginning of the project. Without them this project would never happen.

I thank Prof. Alan Mackworth for thoroughly reading my thesis and providing comments.

I’m thankful to members of the Intelligent User Interfaces Group in UBC. Their support is priceless. I’m especially grateful to Samad Kardan for all the invaluable suggestions and discussions that helped me throughout this project.

I would like to thank my parents and my brother for their unconditional support at every step of this journey. I want to thank Georgii for sharing all the ups and downs with me. Finally, I’m grateful to my dear friends Misha, Narek and Iryna who added so much fun to my graduate student life.
To my Mom and Dad
for all their love and support
Chapter 1 Introduction

Intelligent Tutoring Systems (ITSs) are systems designed to simulate behavior of a human tutor [1]. These systems learn a student model over time based on observable student behaviors. The student model enables an ITS to provide personalized adaptation (e.g., feedback or hints) calibrated to a user’s level of mastery, learning behaviors, and specific needs, with the aim of improving the student’s learning experience.

Student modeling can be a difficult problem, because a large gap often exists between the students’ behaviors observable by an ITS and the students’ states and processes that must be modeled by the ITS. One approach that is being explored to address this problem investigates the value of data from a variety of sensors (e.g., action logs, eye-tracking, physiological data, facial expressions, body language, emotion reports) that can reduce the gap between student’s relevant states and the observations of the ITS.

This thesis contributes to the area of research by exploring the value of eye-tracking data (hereafter, also referred to as gaze data) in assessing student learning during and after interactions with MetaTutor, a multi-agent ITS that scaffolds self-regulated learning (SRL) while students study material on the human circulatory system. SRL [2] refers to a set of student skills for planning, monitoring, and evaluating their learning and behavior to constantly improve learning performance. This research is part of a larger endeavor to understand and model the relations among affect, cognition, and metacognition in learning with MetaTutor, by leveraging multi-channel data sources including think-aloud protocols, eye-tracking, human-agent dialogue, log-file, embedded quizzes, galvanic skin response, and face recognition.

We decided to begin focusing on gaze data because evidence already exists that it can provide useful information on several student modeling dimensions: cognitive [e.g. 3–5], metacognitive [6], and affective [7, 8]. Also, with the development of technology, modern
eye-trackers have become less intrusive than head-mounted versions that were used earlier and more affordable, making them more suitable for real-world applications.

We began investigating if and how gaze data can be used to predict learning in MetaTutor, because tracking whether or not a student is learning is important for the tutoring agent in deciding when to provide feedback or hints to improve the student’s learning experience. We also compared the value of eye-tracking data to interface actions data in assessing student learning with MetaTutor.

1.1 Thesis goals and approach

The value of gaze data was explored for predicting learning performance by building classifiers that used common machine learning techniques for: (1) gaze data, (2) interface actions data, and (3) a combination of the two data types. We began by looking at the performance of the models at the end of learning session (i.e., overall performance). Then, we looked at the performance of the classifiers during interactions with MetaTutor, as a function of the amount of interaction data available (i.e., performance over time).

The data for our research comes from a study provided by the S.M.A.R.T. research group at McGill University. The author of this thesis did not participate in the study design or data collection.

The research questions we address are as follows:

1. Can eye-tracking data be used to assess learning performance of a student interacting with MetaTutor
   a. At the end of the session?
   b. In real-time during interaction?
2. How well does eye-tracking perform in predicting student learning
3. Which elements of the interface contribute most to assessing learning?

1.2 Contributions of the work

The main contribution of this work are the findings that show that gaze data can indeed be a useful source of information for predicting student learning with MetaTutor. This conclusion is especially important because it does not exist in isolation.

In previous work [4], Kardan and Conati demonstrated that gaze data is a good predictor of learning for a different type of ITS (an interactive simulation to support learning by exploration). Specifically, they showed that models that are based solely on gaze can achieve high accuracy in distinguishing high and low learners. In [9], the authors extended their work by combining gaze data with interface actions data. Their findings suggested that eye-tracking data, if used in addition to interface actions data, significantly improves learning prediction accuracy in both overall (when full data about interaction is available) and over time (when only partial data is available) settings for their learning environment.

In this thesis, we approach a similar problem but with a different type of tutoring system, the MetaTutor hypermedia learning environment. The results reported here confirm the importance of gaze data as a predictor of learning across different types of learning environments that can be leveraged for providing real-time personalized support.

We found that eye-tracking data by itself is a good predictor of learning. Furthermore, it improves detecting low and high learners when used as an additional source of data in the action-based model. This finding is in-line with earlier reports [9]. In addition, we found that, in MetaTutor, some gaze-based classifiers perform significantly better than
action-based models and achieve the best balance when predicting high vs. low learners over time.

The comparison of gaze-based classifiers with actions-based classifiers reveals that eye-tracking data can be a beneficial alternative to an action-based model when predicting learning in an environment with a rather open-ended interaction, where students learn from browsing and reading relevant content and are free to use additional tools (i.e., taking notes or SRL tools) at any point in the learning session. This can be especially important because constructing action-based features can sometimes be a difficult task due to the complex nature of the interactions that require a deep understanding of the various interface actions for students to perform in an ITS. In contrast, eye-tracking features can be constructed with relatively little knowledge about the nature of the interactions within the interface, by defining a limited number of areas of interest and calculating the simple gaze-based measures that are relevant.

1.3 Outline

The rest of the thesis is organized as follows: Chapter 2 discusses related work on user modeling and student modeling and the use of eye-tracking for predicting different user states and behaviors (cognitive, meta-cognitive, and affective). In Chapter 3, basic concepts of self-regulated learning are introduced and MetaTutor, the test-bed environment used in this project, is described. In addition, the study that generated the data used in this thesis is described. In Chapter 4, the pre-processing of raw eye-tracking data and the feature set that was derived from the data is described. Chapter 5 describes the relevant MetaTutor interface actions and the relevant features that are derived from the actions. In Chapter 6, the machine learning techniques that were used in this project are explained. Chapter 7 describes the results for using only eye-tracking data to assess overall learning performance in MetaTutor. The goal of this chapter is to compare different sets of gaze-based features; for example, gaze-based measures that are interface independent vs. a set of gaze-based features that are specific to the
interface. In Chapter 8, the performance of classifiers that are trained on gaze features only, actions features only, and on a combination of these two feature sets are compared. The classification results are described for overall accuracy (at the end of interaction) that is achieved by training a set of machine learning algorithms and an ensemble of classifiers that combine sets of the classifiers. We also provide details on simulating predictive accuracy over time for a set of best performing classifiers. Finally, Chapter 9 presents the conclusions for this study and discusses limitations and future work.
Chapter 2  Related Work

2.1  User modeling

User modeling is one of the core components in designing intelligent interfaces. For meaningful personalisation and an adaptation of the interface to improve the user's experience, the system collects user data from various sources to build a user model. Models of the user's behavior and preferences have been designed for various applications, including recommender systems [e.g., 10–13], search tasks [e.g., 14–16], interactive visualizations [e.g., 17, 18], and educational applications [e.g., 9, 19–21].

User modeling gained a lot of attention in recommender systems. For example, Krumm et al. [12] showed the possibility of predicting locations where a user is likely to travel based on simple features describing his or her previous destinations collected from Twitter. Another application of user modeling is recommending meals in a restaurant based on information about likes/dislikes/allergies of customers [13].

Many researchers use information on explicit interface actions (e.g., mouse clicks and pressing buttons on a keyboard) to build user models [e.g., 22–24], as the most common and affordable source of data to track. Nevertheless, depending on what the designer is aiming to predict, using other sources of data might work better. For example, explicit interface actions might not be sufficiently descriptive for building a student model when explicit actions are minimized (e.g., when the core activity of the user-interface interaction does not involve explicit actions like when the interaction involves substantial reading). Some evidence also indicates that other types of data works well when predicting affective and cognitive states of the user; for example, eye-tracking [e.g., 4, 6], heart rate [e.g., 25, 26], pupil size, skin conductance [e.g., 27–29], electroencephalography [e.g., 28, 30], mouse pressure sensors [e.g., 27], and posture [e.g., 27].
2.2 Eye-tracking in user modeling

In this thesis, we explore user models that leverage eye-tracking data. Due to a rapid development of technology, eye-trackers have been become more affordable in recent decades and gaze data is often considered as an alternative or additional source for building user models.

In psychology, extensive research has examined eye-tracking for understanding cognition and perception. Liversedge and Findlay [31] showed that eye-tracking can indicate the cognitive processes that underlie visual search and reading. This motivated the use of eye-tracking data for user modeling in a variety of applications like information visualizations [32], driving simulators [33], search tasks [16], and problem-solving games [34].

Steichen et al. [32] explored the value of eye-tracking data for building models of users working on visualization tasks with two different types of graphs (bar graphs and radar graphs). The authors used a set of eye-tracking features to predict a user’s cognitive abilities, including perceptual speed, visual working memory, and verbal working memory. They ran a set of classification experiments and performed a detailed analysis of features that contribute the most to predicting the different characteristics. They found that attention to specific elements of the interface was crucial for better predictions of cognitive ability. The authors also reported that the approach works well for online cases where only partial data is available for evaluating personal characteristics.

Palinko et al. [33] used pupil size, recorded with a remote eye-tracker during a simulated driving task, to estimate cognitive load. Their results suggest that pupilometry can be a good source for estimating cognitive load during driving tasks.

Cole et al. [16] studied the use of eye-tracking data when performing a set of search tasks. They defined four tasks from the area of journalism and found that eye movement behavior is a plausible source for evaluating a user’s seeking strategy and task type.
The authors concluded that the ability to unobtrusively detect task type through eye-tracking could lead to adaptations tailored to the cognitive processes of the user.

Eivazi and Bednarik [34] analyzed gaze information during a problem-solving task in an interactive 8-tile puzzle game by training a Support Vector Machine to predict the user’s problem-solving cognitive states and performance. Users were divided into three levels of performance (low, medium, and high). The authors compared the values of gaze features for the groups and found that the high-performance group had fewer fixations (i.e., maintaining gaze at the same point of the screen) but longer fixation durations. Other features were difficult to compare due to a high variance in values. The results showed a low performance in predicting cognitive states but a good accuracy in predicting performance. Class-wise, their classifier performed better for predicting the low performers. The authors concluded that gaze features carry important information about a user’s problem-solving skills.

2.3 Student modeling

A subfield of user modeling research focuses on modeling students’ states to improve the learning experience with an ITS through content adaptation, personalized feedback and hints.

Similarly to user modeling, researchers in student modeling often use various sources (e.g., action logs [e.g., 9, 35], eye-tracking information [e.g., 4, 6], posture [e.g., 36, 37]) for assessing a student’s different states of cognition [e.g., 9], meta-cognition [e.g., 6, 8], and affect [e.g., 7, 8, 36, 37].

In the context of modeling students’ SRL processes, researchers have mainly relied on mining action logs. For instance, Kinnebrew and Biswas [38] used sequence mining of action logs to identify effective and ineffective behaviors in students who were interacting with Betty’s Brain, an ITS for scaffolding SRL via teachable agents. Bouchet et al. [39] conducted similar work with MetaTutor, the ITS used in this thesis. Sabourin
et al. [40] mined both actions and students’ self-reports of their affective states for early prediction of SRL processes during interactions with Crystal Island, a narrative-based and inquiry-oriented serious game for science.

2.4 Assessing students’ cognitive states with eye-tracking

Much work has been done with eye-tracking for predicting cognitive states of students. In this section, we focus on the value of eye-tracking in assessing cognitive states. First, we will discuss projects where eye-tracking is used for offline analysis to understand behaviors that contribute to cognitive processes. We will then discuss projects that attempt to use eye-tracking data in real-time to classify students by their performance and cognitive behavior.

2.4.1 Offline analysis of cognitive processes with eye-tracking data

Eye tracking has been used in a variety of research projects for offline analysis to explore relevant student cognitive processes [19, 20, 41, 42].

Anderson and Gluck and Gluck et al. [5, 19] present the earliest exploratory research into using eye-tracking data for student modeling. The work is innovative, in the sense that previous research was mainly focused on using actions data to evaluate cognitive processes while these authors looked at how eye-tracking data enriches the knowledge of the system about the student. They described several ways in which eye-tracking data can be used to model students of the EPAL Algebra Tutor, including the functionalities: predicting errors, detecting undesirable solution processes, and identifying students who ignore error messages. In the later work, Gluck and Anderson [5] discuss privacy issues of using eye tracking in student modeling.

Hegarty et al. [41] looked at the value of fixation-based information in a study of arithmetic word problems. Their study provides evidence that problem solvers who
focus on the words used to construct the problem are more likely to succeed than those who focus heavily on numbers and relational terms instead of other words.

Tsai et al. [20] conducted a study to examine students’ visual attention when predicting debris slide hazards in a multi-choice science problem. They analyzed fixation durations in a set of pre-defined areas of interest and found that all students tend to spend more time exploring relevant information. They also reported a difference in the scanning sequences of elements between different types of learners. Successful learners shift attention from irrelevant to relevant content, and unsuccessful learners shift their attention in the opposite direction.

Conati et al. [42] explored the factors affecting attention to adaptive hints during interaction with Prime Climb, an educational computer game designed to provide personalized support for learning factorization skills based on a student model. This was the first work that studied adaptive hints in an educational game. The authors found that attention to hints in Prime Climb positively affected students’ performance with the game (i.e., correctness of the player’s moves). They found that various factors, such as time and type of hint, attitude towards receiving help, and move correctness somewhat affected a student’s attention to hints provided in the game. The authors focused on three basic eye-tracking measures: total fixation time, fixations per word, and time to first fixation in each predefined area of interest. They found that attention to hints increased after a correct move. The authors noted that students may have been treating hints after a correct move as positive feedback. Another finding was that students with a positive attitude towards help tended to pay more attention to hints after a correct move.

2.4.2 Real-time assessment of cognitive states using eye-tracking

Some work has already used eye-tracking information for assessing cognitive states in a real-time setting.
Sibert et al. [21] explored gaze-tracking to assess reading performance in a system for automated reading remediation that provides support if the user’s gaze-patterns indicate difficulties in reading a word. This work was an early attempt to use eye tracking for real-time student assessment.

Amershi and Conati [43] combined gaze and actions data to model students’ reasoning during interactions with the Constraint Satisfaction Problem (CSP) applet, an exploratory learning environment. This educational tool allows a student to explore algorithm dynamics via interactive visualizations. The CSP applet consists of a set of variables, variable domains, and a set of constraints on variable values. Solving CSP requires finding a set of values that satisfies all constraints. The CSP applet illustrates the Arc Consistency (AC-3) algorithm for solving this problem. Algorithm flow is shown with interactive visualizations. A student can manage the flow of the algorithm by triggering execution one-step at a time, automatically running all steps until the problem is complete, and selecting a variable for splitting for further application of the algorithm.

The authors proposed a framework that combines both unsupervised and supervised classification to build a student model. Their approach relies mainly on automated techniques and thus does not require a lot of time and effort, as opposed to the knowledge-based approach where the model is derived from domain knowledge from experts. Amershi and Conati clustered students using k-means algorithms and feature vectors that combine actions and gaze information. Next, they analyzed the clusters to determine effective and ineffective interaction behaviors. These steps resulted in two clusters that represent effective and ineffective students. In the Online Recognition phase, the clusters were used to train a supervised classifier to detect successful and unsuccessful students online. The gaze info used in the study [43] consisted of simple, predefined gaze shifts between two salient areas of the interface.

This work was extended in [4], where a broader set of gaze features were used. Kardan and Conati showed that the enriched eye-tracking data by itself can be a valuable source for distinguishing good learners from low learners in the same environment, as was shown in their earlier work [43]. They continued this work by adding actions-related
information to the models [9], and found that the combining of action and eye-tracking data to the student model for the CSP Applet significantly improved the average online performance of classifiers that were trained to distinguish high learners from low learners.

2.5 Predicting students’ affective states with eye-tracking

At the affective level, Qu and Johnson [7] leveraged gaze data to assess student motivation in an ITS for teaching engineering skills. Their ITS used a Bayesian model to infer learners’ attention based on eye-tracking information and interface events. The system constructs a motivational model that can assess students’ motivation at any moment of the learning session. This model can be used to provide proactive help during learning to motivate the student’s learning.

Muldner et al. [8] looked at pupil dilation to detect relevant student affective and meta-cognitive states during interactions with EA-Coach, an ITS that supports analogical problem solving. They initially planned to analyze fine-grained emotions (e.g., “happy,” “sad,” and “bored”) using the students’ self-reports; however, the collected data had many ambiguities that prevented the classification of exact emotion. To increase the specificity, all instances of affect were divided into positive and negative, which resulted in a reasonable amount of data. The authors found that pupil dilation differed significantly for the positive and negative affective states. Larger pupil size was observed when students expressed positive affect. The authors concluded that the knowledge about the user’s affect could be used to provide a tailored interaction.

D’Mello et al. [3] designed Guru, a gaze-reactive ITS for biology topics that detects a student’s boredom and disengagement and reacts to promote positive affect and learning. Guru delivers content in the form of a dialog between a Pedagogical Agent and a student. The system monitors the student’s attention to the agent or relevant content and provides a spoken prompt (e.g., “Please pay attention”) if the student is not
continuously looking at the screen. In general, gaze-reactive prompts were effective in re-orienteeering the student's attention to the pedagogical agent. Nevertheless, students persisted to look off the screen. The results show that tracking a student's attention toward the Pedagogical Agent to guide the student's attention, improves student learning.

2.6 Predicting students’ meta-cognitive states with eye-tracking

At the meta-cognitive level, Conati and Merten [6] showed that using gaze data improved the ability of a student model to track students' self-explanation behaviors (i.e., generating explanations to one-self to improve one’s understanding), and consequent learning.

Muldner et al. [8], as discussed earlier, in addition to analyzing affect, also investigated the relationship between reasoning and pupillary response. They compared three types of reasoning (self-explanation, analogy, and other reasoning), and the results supported the idea that reasoning has an impact on pupil dilation. Pupil size was significantly larger for self-explanation than for other reasoning, though no significant difference in pupillary response was found between self-explanation and analogy. The authors concluded that the model may need additional information to distinguish these two types of reasoning since analogy requires the comparisons.
Chapter 3  MetaTutor and Self-Regulated Learning

MetaTutor, an adaptive hypermedia learning environment on the circulatory system, was used as a test-bed for our work. In addition to providing structured access to the instructional text and diagrams on the circulatory system, MetaTutor includes a variety of components designed to scaffold the learner’s use of SRL processes. MetaTutor was designed and developed by the research group of Dr. Roger Azevedo [44].

In this chapter, Self-regulated learning (SRL) is introduced and the model underlying MetaTutor is described to the level of detail sufficient for understanding this thesis. Then, the MetaTutor interface and the tools available to scaffold SRL are described. Finally, the study that was conducted by colleagues from McGill University for collecting the data used in this thesis is described.

3.1  Self-regulated learning

SRL is theory of human learning that focuses on students’ skills to plan, monitor, and analyze their learning processes to improve learning performance. Self-regulated learners use cognitive and metacognitive processes, such as setting goals, planning actions to achieve them, organizing their learning activities with a set of learning strategies, and monitoring and evaluating their performance during learning. One of the key points of self-regulation is that self-regulated learners are aware of the relations between the strategies they apply and the learning outcomes they achieve [45]. They constantly monitor the effectiveness of their learning methods and strategies and react to any feedback to improve learning. While learners show a deeper understanding and better performance in learning tasks when using self-regulation effectively, a large proportion of students do not know how to effectively self-regulate.
Thus, some researchers have focused on developing environments that support SRL (e.g., MetaTutor) by scaffolding self-regulatory processes related to planning, monitoring, and using learning strategies.

Researchers in this area developed a number of SRL models that attempt to explain how to effectively regulate learning activities [45–47] by: (i) proposing a time-ordered sequence of phases (such as planning, monitoring, control, and reflections) that students follow during learning and (ii) explaining how cognitive and motivational factors influence the learning process.

The SRL model that underlies MetaTutor is based on Winne and Hadwin’s model [2, 48], which was later extended by Azevedo and colleagues [44, 49]. Briefly, Winne and Hadwin divided the learning process into four basic phases:

1) defining tasks;
2) setting goals and planning;
3) enacting studying tactics;
4) adapting metacognition.

Winne and Hadwin separated task definition from setting goals. During the first phase of defining tasks, students scan the learning environment, the tasks given to them, and their knowledge about themselves to create their personalized perception of the task on which they will work [48]. Subsequently, students proceed to the second phase where they create current learning goals and plan strategies that will help them to achieve them, in the context of the task defined during the previous phase. In the third phase, students apply strategies and tactics that were set during the previous phase. At this point, the learner works on the task itself. In the last phase, students may choose to make changes to the way in which self-regulated learning is carried out if the outcomes of the previous stages are different from the expected standards. The student may also abandon this phase if a change to the tactics is not needed. Successful learners
reiterate through these phases; for example, they might choose to update their goals or experiment with new strategies for learning.

This model emphasizes the role of cognitive monitoring and control during all phases of learning. The final model incorporated in MetaTutor [44, 49] is based on several assumptions. First, successful learning involves a constant monitoring and regulating of cognitive, affective, and metacognitive (CAM) processes during learning. Second, usage of SRL is specific to the learning context and should be changed based on the current stage of learning. Third, learners should be able to monitor and control both internal factors (e.g., their prior knowledge on the topic) and external factors (e.g., utility of the current content) to succeed in learning. Fourth, a successful learner must make adaptive, up-to-date adjustments to these conditions based on judgments of their use of CAM processes. Finally, certain CAM processes (e.g., personal interest in the task and value of the task in achieving goals) are necessary for a learner's motivation.

The skill of using SRL is critical when studying with an open-ended learning environment like MetaTutor [49], where the student is exposed to hundreds of paragraphs of text and static diagrams. A successful, self-regulated learner should be able to regulate and use a set of key CAM processes while studying in the environment. These processes include selecting content that is relevant to the current subgoal, spending appropriate time on each page depending on the relevance of the content, deciding when to change a current subgoal, assessing the student's understanding of the content and updating it during the learning session, connecting prior knowledge with content, using a set of learning strategies (e.g., re-reading, summarizing, and making inferences), and making changes to the learning behavior based on feedback from the environment/teacher, timing, performance, and affective experiences, to improve learning.
3.2 Overview of the environment

The version of MetaTutor that was used in the present work [44] includes 38 pages of text and diagrams on the human circulatory system, organized with a Table of Contents that is displayed in the left pane of the environment (Figure 3.1). Students can easily navigate through the content by clicking on the relevant topic name. Text and diagrams are displayed separately in the two central panels of the interface.

![Sample MetaTutor interface](image)

**Figure 3.1: Sample MetaTutor interface**

By default, only a thumbnail image is available on the interface (see Figure 3.2 for an example). Thus, if students want to study the image in detail they need to click on the image thumbnail to get the full-size image (as in Figure 3.1).
Four pedagogical agents (PAs) are displayed in turn in the upper right-hand corner of the environment [50]. Each agent serves a different purpose: Gavin the Guide helps students navigate through the system, Pam the Planner assists students with setting appropriate subgoals for the learning session, Mary the Monitor helps students to monitor their progress towards finishing the current subgoal, and finally, Sam the Strategizer suggests ways for successfully using tools for writing notes and summaries. One PA is active at any given moment of the learning session. The interactions with PAs occur through the agents’ spoken prompts and feedback. Students react to the PA’s actions by typing an answer in a corresponding field (e.g., when asked to report prior knowledge) or by choosing the best matching option in a provided list of answers (e.g., when asked how confident the student is about the material covered).

The interface has several layouts that appear on the screen, based on the current phase of learning. The key elements of this multi-layout system are the content areas, including text and diagrams, Table of Contents, clock widget, goal and subgoals of the...
session, input area, agent face, and palette with additional SRL-related functionality supported by MetaTutor. The functionalities include taking notes (using a notepad that is embedded in the interface), writing summaries of the viewed content (typing in a special area), evaluating one’s current understanding of the information read by taking quizzes, and reporting the relevance of the current content to the active subgoal, etc. All of these can be initiated via the learning strategy palette, displayed in the right interface pane (Figure 3.1) or by explicit directions of a corresponding PA. When a student needs to type something in the interface (e.g., setting a new learning subgoal or reporting prior knowledge about the topic before exploring it) the layout shown in Figure 3.3 appears on the screen. At this point, only the input area is active and the student cannot access any content or other tools until finishing this step. The corresponding PA will guide the student through this process if necessary.

Figure 3.3: Input layout
MetaTutor has two levels of goals. The overall learning goal of the session is set by an instructor or a MetaTutor administrator and shared by all students during the session. It is located in the central upper pane of the MetaTutor interface (a large blue pane in Figure 3.1 under “Learning Goal and Subgoals”). The learning subgoals are established at the beginning of the learning session by students and can be updated at any moment in the learning session. In the MetaTutor interface, the learning subgoals are located under the overall subgoals. For example, in Figure 3.1, a student has two subgoals set: "Heart components" and "Blood components". Pam the Planner assists the student in choosing the two initial subgoals related to the overall learning goal for the session by parsing the student's input and telling the student if the chosen subgoals are too specific or too broad. The PA will also help the student set up a better subgoal, if needed. Figure 3.4 shows a sample dialogue of a student and Pam during this process. The same agent will appear in the top right corner every time a student decides to add a new subgoal. The shading of the subgoal bars in the corresponding panel show the student's current progress towards completing that subgoal as the interaction proceeds (see shading of "Heart components" subgoal in Figure 3.1).

![Figure 3.4: Setting subgoal dialog history](image)

By default, all students begin learning with the first initial subgoal being active, though they can change the order of subgoals by prioritizing any other subgoal at any moment.
All students are encouraged to take notes when learning with MetaTutor, in one of two ways: by using an embedded notepad interface that is available from the right pane of the screen, or by using a digital pen that is connected to the computer. A sample of the note-taking layout is shown in Figure 3.5.

![Figure 3.5: Embedded notepad interface](image)

The rest of the tools provided within MetaTutor are to trace and foster the student’s usage of SRL processes, and can be initiated from the learning strategies palette; they do not have a separate layout.

A timer is available during the session, in the right corner, to ensure that students are aware of the time constraints. The understanding of the material is monitored by taking
page quizzes (available from the learning strategies palette, or initiated by a PA) and subgoal quizzes (available when completing a subgoal), or by writing summaries of the covered material (available from the learning strategies palette, or initiated by a PA). Students can evaluate the relevance of the active content to the currently active subgoal on their own by using a corresponding tool from the learning strategies palette or when asked by a PA.

MetaTutor also has a Full View layout (Figure 3.6), where only content (text and diagram) are available with no access to SRL tools.

![Figure 3.6: Full view layout in MetaTutor](image)

The lungs are important because they are where gases are exchanged between the circulatory system and the ambient air. Blood circulating through the lungs picks up oxygen, carries it to the heart, where it is pumped throughout the body. This is also where carbon dioxide moves from the blood, back out into the air, to be pushed out when the lungs deflate as a person exhales. Blood returns to the left atrium of the heart via the pulmonary veins.

Air enters the body through the nostrils or mouth, down the pharynx, past the vocal cords, and through the trachea. The trachea splits into two branches, the left and right bronchial tubes, which split into smaller and smaller bronchioles, down to the alveolus (plural for alveoli). This structure is akin to a tree trunk splitting into smaller and smaller branches. Alveoli are grape-like air sac structures wrapped in capillaries. When the alveoli inflate with inhaled air, oxygen is transferred from the air to the red blood cells and carbon dioxide dissociates from red blood cells and exits the circulation as exhaled wastes.

There are between 300 and 400 million alveoli in each lung and if they were spread out, their surface would occupy approximately 1,000 square feet, nearly 50 times the total surface area of the skin.
3.3 Description of the study to collect behavioral data during interaction with MetaTutor

A study was conducted in 2012 by colleagues from McGill University with the goal of collecting multi-channel data to examine the role of cognitive, metacognitive, and affective processes during learning with MetaTutor [51]. The sources of collected data included think-aloud protocols, video and audio recordings, and log files containing complete information on the interaction with the learning environment. In addition, learners also wore an Affectiva Q sensor skin conductance bracelet [52] to measure fluctuations in their arousal. Eye-tracking information was collected using the Tobii T60 eye-tracker [53].

All participants used MetaTutor on desktop computers with a Core 2 Duo 2.80 GHz processor, 2 GB of RAM and Windows XP, using a 17" monitor with a 1024 x 768 resolution and the MetaTutor application running in full-screen mode. The Tobii eye-tracker was embedded in the monitor.

The study included two conditions: an adaptive condition, in which the MetaTutor’s PAs provided prompts and feedback adapted to each student’s performance; and a non-adaptive condition, in which prompts and feedback from agents were generic and were generated without information about student assessment. Participants in the non-adaptive condition had access to all SRL tools in MetaTutor. Apart from the PAs' feedback, the interface and functionality were identical in both conditions.

The study consisted of two sessions. In the first session, participants (university students who were randomly assigned to one of the two study conditions) completed a pre-test on the circulatory system and a demographics questionnaire. In the second session, participants were working with the MetaTutor environment. The time-line for the sessions and the average durations for the participants are shown in Figure 3.7. The sessions began with the calibration of the apparatus, including the Tobii T60 eye-tracker and followed by a welcome video tutorial that explained the main features of the interface. Students watched additional tutorials at random times in the session, which
provided hints about some of the functionalities of the interface. Different students could watch different numbers of tutorials based on the set of tools they were using during the session. Subsequently, each participant was asked to set two subgoals for the session; i.e., specifying two areas of the circulatory system that the student was aiming to learn during the session. On average, students spent 15 minutes setting the subgoals. The participants then studied with MetaTutor for exactly one hour, excluding time spent on an optional break and on completing the questionnaires. The participants always began with the first subgoal, but were able to prioritize the other subgoal, or set a new one. The MetaTutor session was followed by a post-test. During the MetaTutor session, students were stopped at several points to complete a set of questionnaires on emotions. All participants were offered an optional 5-minute break in the middle of the session.

![Figure 3.7: Main session timeline](image)

In the experiments presented in this thesis, eye-tracking data and actions logs that were recorded during the interaction with MetaTutor were analyzed. We build classifiers to distinguish students by their performance, based on the results of the pre-test and post-test collected during the study. The analysis of the responses to the emotion questionnaire is covered in a separate research project on affective levels and is beyond the scope of the present work.
Chapter 4 Preparing and Processing the Eye-Tracking Data

In this chapter, we begin by describing the preparation of raw gaze data. First, we removed data that was not related to the learning session to minimize the presence of irrelevant data in the experiments. Then, we describe the gaze data validation, where we discard low quality data from some students’ gaze data. In the last section, we describe two groups of features (interface-independent and interface-specific) that form the final feature set used in our experiments.

4.1 Preparation of raw gaze data

The Tobii T60 eye-tracker was used to collect eye-gaze data. This model is integrated into a 17” monitor and thus allows subjects to move naturally during a study. The Tobii T60 collects data at a rate of 60 Hz (60 gaze data points per second for each eye) with precision and accuracy for horizontal coordinates of 0.18-0.36°/0.4-0.5°, and vertical coordinates of 0.18-0.30°/0.4-0.6° [54].

Interactions with an eye-tracker of this kind is not much different from interactions with a standard desktop computer. Because of this, however, the collected data might have a high level of noise (invalid samples) due to the loss of calibration during rapid movements, not looking at the screen, or blinking.

In this section, we describe the key elements of data preparation for reducing the amount of invalid samples in the dataset. This is an important step because we need to guarantee that the data is of high quality for calculating the features.

The approach we adopted in this study for preparing and processing eye-tracking data is generic and can be applied to a variety of interfaces, because feature construction does not depend on information about the interface with which the students are
interacting. The approach was proposed in [4] and has been successfully used for another type of learning environment.

The Tobii eye tracker records gaze position, pupil size, and other measures with relevant time-stamps at a constant rate of 60 samples per second. The raw eye-tracking data is exported from Tobii Studio (supporting software for processing raw gaze data recorded by Tobii eye-trackers [55]) in tab-separated files. These files provide easy-to-read information of all measures tracked by a Tobii eye-tracker. The researchers can export partial data (e.g., certain lists of measures), if necessary.

In addition to Tobii Studio, we used pre-processing and validation tools available in the EMDAT (Eye Movement Data Analysis Toolkit) developed by the IUI group at the University of British Columbia [56]. EMDAT requires a set of files exported from Tobii Studio as inputs.

The preparation of raw gaze data includes two major separate steps: pre-processing and validation. During the pre-processing, we discard data that is not relevant to the study or use artificial techniques to improve the quality. At the validation step, we discard invalid data (data with low quality).

4.1.1 Pre-processing of gaze data

First, during pre-processing, we removed gaze samples corresponding to activities that did not contribute to learning about the circulatory system and that were added for study design purposes only. The list of these activities includes: taking emotion questionnaires (that popped up on the screen every 15 minutes in reading sessions, blocking the MetaTutor interface), watching video tutorials on using MetaTutor, and taking an optional break in the middle of the study. During these activities, no interaction with the MetaTutor interface occurred and the students were not completing any tasks related to studying about the circulatory system.

We defined three different types of interaction between the user and MetaTutor:
1. **Main**: corresponds to normal interaction with the MetaTutor environment, including setting subgoals and browsing the content;

2. **Questionnaire**: filling in all questionnaires that students were asked to fill in during interactions;

3. **Video**: watching video tutorials that introduce MetaTutor and give details on using certain functionalities of the tutoring system.

For the purposes of this analysis, we were only interested in activities that represent learning. We focused on the Main interaction only as it aggregates all activities related to studying the content on the circulatory system in MetaTutor. Other types of interaction may have been useful for different types of analyses; e.g., predicting affective states during a session.

The next step of pre-processing was to reduce the level of noise in the gaze data that was recorded during the learning session. Noise can be caused by two types of sources: students not looking at the screen (whether or not due to session-related task) and eye-tracker fails to track gaze while student is looking at the screen (e.g., due to loss of calibration, blinking, blockage between camera and eye).

We addressed the noise from the first group by removing sequences of data corresponding to session parts where the student typically looks away; for example, during the learning session when students are encouraged to take notes and communicate with the agents by typing information in the corresponding area of the interface. During these activities, students tend to switch their gaze between keyboard, their notes, and the screen. When the eye-tracker loses track of the student's eye, invalid data is reported. Thus, even though the eye tracker was working properly, the rate of valid samples was rather low, which affects the overall quality.

Moreover, some invalid samples that are not captured by the eye-tracker, while a student is looking at the screen, can be restorable. If the user is looking at the same point before and after a short sequence of invalid samples, it can be assumed that the user was looking at the same point during this “loss”.
4.1.2 Gaze data validation

For the purposes of this research, high quality data is needed to guarantee a reliable representation of a real-world situation. The quality of data is defined by the level of valid samples in the gaze data for each participant (e.g., validity level).

We validated the data from pre-processing using the technique described previously [4]. The validation process is performed in three major steps (Figure 4.1).

![Diagram of the validation process](image)

**Figure 4.1: Validation process**
In Step 1, we assess the validity level of each participant and remove all participants with a rate of valid samples that falls below the threshold we set (75%). The threshold was chosen after looking at the relation between its different possible values and the percentage of valid subjects remaining for analysis after using the threshold. From Figure 4.2, the percentage drops drastically with values of the validity threshold that are above 75%. The lower thresholds filter out fewer participants but are less reliable.

In Step 2, we removed continuous invalid samples to improve the validity level of the available gaze data (henceforth, this process is referred to as auto-partition). The auto-partition searches for all invalid gaps larger than 300 ms and discards them until no more can be found. This process is fully automated in EMDAT. The remaining sequences of gaze samples created by auto-partition are called segments. The auto-partition improves the overall validity level of participants since it creates separate segments by removing invalid samples only. In this case, the final calculations would not be affected by continuous invalid data. Nevertheless, some of the segments may
still be of low quality due to the sparsity of invalid samples, and they should be removed. For this, we used the same threshold as in step 1 for discarding participants (75%) due to limitations of implementation in EMDAT.

Since auto-partitions may result in discarding too much data, making the remaining data meaningless, we remove from our dataset any participants that have a large amount of discarded data (Step 3 in Figure 4.1).

Another threshold (80%) was used to control this process. Similarly to the previously described threshold, we looked at the relation between the remaining data rate threshold value and the number of participants that would come above it (Figure 4.3).

![Validity threshold for auto-partition](image)

**Figure 4.3**: Percentage of valid participants with different thresholds for auto-partition

Two of the chosen thresholds guaranteed a fair quality of data for the analysis and would not remove too much data. The higher thresholds resulted in a drastic increase in the discarded data; however, the lower thresholds did not seem to be reliable.
4.2 Eye-tracking features

The Tobii eye-tracker captures gaze information in terms of *fixations* (i.e., maintaining gaze at one point on the screen, shown as circles in Figure 4.4) and *saccades* (i.e., a quick movement of gaze from one fixation point to another, shown as a line between two fixations in Figure 4.4). Gaze patterns are further defined by measures that represent gaze direction, for example:

- *Absolute path angle* - the angle between a saccade and the horizontal (angle X in Figure 4.4)
- *Relative path angle* - the angle between two consecutive saccades (angle Y in Figure 4.4).

Following the approach suggested by [57], and followed in [4], we computed a large variety of features based on raw gaze data. These are divided into two types. The first type was generated by applying summary statistics such as mean and standard deviation (SD) to the above measures, taken independently of the specific interface layout. This process generated 10 features representing general gaze trends that do not take into account the information on the interface (i.e., MetaTutor), or Areas of Interest (AOI) (see column “No AOI features” of Table 4.1). The second type consists of features that incorporate interface-specific information, in terms of salient AOIs of the
MetaTutor’s interface. We defined seven of these AOIs (labeled with rectangles in Figure 4.5): Text Content, Image Content, Goal, Subgoals, Learning Strategies Palette, Agent, and Table of Contents.

- **Text Content**: this AOI covers the area with textual material from 1 of 38 pages available in MetaTutor.

- **Image Content**: this AOI covers the area that contains illustration associated with the current text content. Text Content and Image Content cover the central area of the interface.

- **Table of Contents**: this AOI covers the most left area of the interface, and provides structured listing of sections that make navigation through the content easy.

- **Overall Learning Goal**: this AOI covers the text that provides information on the purpose of the session. This part is static and remains the same for all participants.

- **Subgoals**: this AOI covers an area with subgoals set by the student for the learning session. Overall Learning Goal and Subgoals cover the top central area of the screen.

- **Agent**: this AOI covers the top right corner of the MetaTutor interface and is a visual representation of one of the PAs.

- **Learning Strategies Palette**: this AOI covers an area under the Agent and provides access to a set of SRL tools and note-taking.
Figure 4.5: Sample MetaTutor interface with AOIs highlighted

Table 4.1: Description of gaze-based features

<table>
<thead>
<tr>
<th>No-AOI Features</th>
<th>AOI-based Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate and Number of Fixations</td>
<td>Fixation rate in AOI</td>
</tr>
<tr>
<td>Mean and SD of Fixation Duration</td>
<td>Proportion of fixation time and fixation number in AOI</td>
</tr>
<tr>
<td>Mean and SD of Saccade Length</td>
<td>Duration of longest fixation</td>
</tr>
<tr>
<td>Mean and SD of Relative Path Angles</td>
<td>Proportion of transitions from every other AOI to the current one (7 different features)</td>
</tr>
<tr>
<td>Mean and SD of Abs Path Angles</td>
<td></td>
</tr>
</tbody>
</table>

For each AOI, we calculated the following features: rate of fixations, proportion of fixation time and of fixation number, and duration of longest fixation. In addition to these features, proportional measures were used to assess the relative magnitude of attention.
devoted to each AOI over the course of a complete interaction (proportion of transitions from every other AOI to the current one and proportion of transitions within the AOI). In total, 77 AOI-based features were determined (summarized in column “AOI-based Features” of Table 4.1).

Most of the AOIs were available during the whole session; however, text content and image content are dynamic AOIs that become visible only during a browsing session, which begins after the student establishes two obligatory subgoals to be covered during the session. This sets certain restrictions on the feature calculations. First, we had to exclude some of the commonly used interface-specific features, like time to first fixation in the interface, since each student needed a different time to make this AOI available on the layout for the first time. Second, we had to revise the calculation of proportions by taking into account only the time when the AOI was available on the screen.

Due to the flexibility in choosing SRL tools to support learning and the open-ended nature of interactions with MetaTutor, some users never tried some of the tools (e.g., embedded notepad for taking notes or reading content in a full-view mode). Because of this, no attention patterns were tracked for some students and these features were excluded from our set to keep the set independent of the student’s explicit choices within the interface. Instead, the information about using these SRL tools was captured in the model by a set of actions related to the features, as discussed in the next chapter.
Chapter 5 Preparing and Processing the Action Data

In this chapter, we describe features based on the student’s explicit actions that were traced during the MetaTutor learning session. First, we describe the log files and the way we retrieve information about the actions from the files. This is followed by a discussion of the actions feature set.

5.1 Parsing logs and features calculation

We used the python scripts package MTLogAnalyzer to process logs. It was developed by the same research group who collected the data used in this thesis (the S.M.A.R.T. Lab at McGill University). MTLogAnalyzer parses log files recorded for each student during interactions with MetaTutor and calculates a set of measures that describe the student’s explicit interface actions. A log file (see sample in Figure 5.1) contains a collection of tab-separated lines with information on all events initiated by the student (e.g., student opens a new page or student adds a new subgoal) or originating in the system (e.g., PAs’ spoken prompts or change of layouts). Each line contains the event ID, absolute and relative timestamps, internal code of the action type needed for parsing, and other details about the event. For example, the first line in Figure 5.1 is a recording of the student opening a new page. This line corresponds to event ID = 142. The last column of this recording contains the page ID (7) and page title (“Parts of Blood Overview”). The event with ID=145 shows an example of a PA’s spoken prompt. The last column contains the exact log of the prompt that is heard by the student.
Figure 5.1: A sample of a MetaTutor log file

The features calculated with MTLogAnalyzer represent various activities that students pursue within the ITS; for example, number of set subgoals, duration of the learning session, number of notes written, and number of times students are evaluated for their understanding of the material. The initial set of calculated measures was designed by Bouchet et al. and described in [58]. Mostly, it contained frequencies of deploying SRL tools during a learning session.

Knowing just the frequencies on their own might not be sufficiently descriptive. For example, some students take notes often but spend little time, while others access the embedded notepad less often but spend more time whenever they work on the notes. To represent these patterns of behavior, the durations should be taken into account as well. Similar logic can be applied to other SRL tools. To address this, we extended the original group of features by computing how much time the tools were used. We also calculated several measures that describe how students work with content in the environment. We added corresponding functionality to MTLogAnalyzer. All frequencies and durations were normalized over session duration, to reduce the impact of variance in session lengths across students.

The final feature set contains 47 features: 26 were from the original MTLogAnalyzer, to which we added another 21. The feature set covers all key processes that occur during learning with MetaTutor, including various reading patterns (e.g., reading relevant or
irrelevant content regarding subgoals, or studying visuals), managing learning goals (e.g., through setting, changing, and completing subgoals), monitoring personal understanding (e.g., through quizzes and summaries) and overall relevance of material (e.g., using special tools in MetaTutor to assess the relevance of the current page to the goals), taking notes, and making inferences from relevant material.

5.2 Actions features

As described in Section 3.3, the MetaTutor session is divided into several parts with various activities. They serve different purposes in the study session and activate several layouts of MetaTutor. These activities include watching video tutorials, setting subgoals, studying content, and completing post-tests. In the feature set, we do not take into account the time spent watching video tutorials or taking tests since these activities are not relevant to learning new material in MetaTutor. The rest of the session can be divided into two parts: 1) planning, when students set two obligatory subgoals for learning and report prior knowledge on the topic, and 2) browsing, when students work with content. Students do not have access to the content of MetaTutor before starting the browsing session. For calculating features, we will consider actions from these two parts only, as other activities are beyond the purpose of this research.

The actions features we calculate are divided by relevance to certain activities in MetaTutor and ascribed to the following groups:

- General features of working with MetaTutor
- Features related to learning goal management
- Subgoal-related features for working with content
- Features describing the note-taking
- Features for SRL tools usage

These subsets of features are described in the following subsections.
5.2.1 General features of working with MetaTutor

This group covers all of the simple measures of working with MetaTutor and the topic content. These features are the most general statistics, in that they do not include any information for using SRL during learning sessions. The list of features with their descriptions are shown in Table 5.1. The features that were available in the initial feature set are marked “McGill” (here and in other tables) in the last column. The newly added features are marked “UBC”.

Table 5.1: Working with content and durations

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>DurationFullInteraction</td>
<td>Duration of the learning session, with time spent on video tutorials and questionnaires excluded.</td>
<td>UBC</td>
</tr>
<tr>
<td>TimeSpentWithContentOverall</td>
<td>Time spent browsing/reading the content (text and diagrams).</td>
<td>UBC</td>
</tr>
<tr>
<td>TotalTimeWithContent</td>
<td>Time spent browsing the content with time spent taking and checking notes or deploying SRL processes excluded.</td>
<td>UBC</td>
</tr>
<tr>
<td>NumberImagesOpen</td>
<td>Number of images opened by student.</td>
<td>UBC</td>
</tr>
<tr>
<td>NumberPagesOpen</td>
<td>Number of pages opened by student.</td>
<td>UBC</td>
</tr>
<tr>
<td>AVGPageTime</td>
<td>Average time spent on page.</td>
<td>UBC</td>
</tr>
<tr>
<td>TotalTimeFullView</td>
<td>Time spent studying in Full View layout.</td>
<td>UBC</td>
</tr>
<tr>
<td>NumberFullView</td>
<td>Number of times FullView layout was accessed.</td>
<td>UBC</td>
</tr>
</tbody>
</table>

a Features were normalized over duration of the session.

b Frequencies calculated per 10-minute periods, normalized over duration of the session.

We calculated the duration of planning and browsing in the DurationFullInteraction feature, which was used as the session length to normalize the other features.
The rest of the features listed in Table 5.1 represent working with content during the browsing session, to provide more information on how students work with the content in MetaTutor.

*TimeSpentWithContentOverall* is based on a measure from the McGill set. The original feature measured the time between timestamps for the first and the last action in the browsing session. We updated it by excluding time spent on quizzes and video tutorials. *TotalTimeWithContent* is similar to *TimeSpentWithContentOverall* but excludes time intervals when students were using SRL tools or when the content was not available in the interface (e.g., when students evaluated their understanding of the material they had just read). This measure is the best estimation of reading duration when no additional sources of data are available.

The *NumberFullView* and *TotalTimeFullView* features bring to the feature set knowledge about working with full view layout, as described in Section 3.2, which was not represented in the eye-tracking feature set due to lack of data for some participants.

Some of the other high level statistics for working with content include: number of pages opened and average time spent on a page (*NumberPagesOpen* and *AVGPageTime*), and the number of times the student opens a large image (*NumberImagesOpen*).

### 5.2.2 Features related to learning goal management

In the beginning of the session, students were asked to set two subgoals. The browsing session began by working on the first initial subgoal, though students could freely change the active subgoal or add a new one. When they believed that they had completed their subgoals, they could validate it by taking a short quiz and then move to the next subgoal or work independently. In this subsection, we describe the features that trace user manipulations for the subgoals. The full list of relevant features is provided in Table 5.2.
Table 5.2: Learning goals management

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>TotalTimeSettingSG(^a)</td>
<td>Total time spent setting subgoals during the session (including 2 initial subgoals set at the beginning of the session and all voluntary subgoals set later).</td>
<td>UBC</td>
</tr>
<tr>
<td>AverageTimeSettingSG(^a)</td>
<td>Average time spent setting a subgoal (including 2 initial subgoals set at the beginning of the session and all voluntary subgoals set later).</td>
<td>UBC</td>
</tr>
<tr>
<td>numSG(^b)</td>
<td>Number of subgoals set during the session (including 2 initial subgoals set at the beginning of the session and all voluntary subgoals set later).</td>
<td>McGill</td>
</tr>
<tr>
<td>numSGValidated(^b)</td>
<td>Number of subgoals validated by student (reported as complete and with a short quiz).</td>
<td>McGill</td>
</tr>
<tr>
<td>numSGAttempted(^b)</td>
<td>Number of subgoals the student tries to work on.</td>
<td>McGill</td>
</tr>
<tr>
<td>numSGChanged(^b)</td>
<td>Number of times user changed the current subgoal.</td>
<td>McGill</td>
</tr>
<tr>
<td>numPLAN(^b)</td>
<td>Number of times student managed subgoals (including setting, changing order, or completing).</td>
<td>McGill</td>
</tr>
<tr>
<td>FlagWorkedWithoutSubgoal</td>
<td>True, if student worked without an active subgoal; False, otherwise.</td>
<td>McGill</td>
</tr>
</tbody>
</table>

\(^a\) Features were normalized over duration of the session.

\(^b\) Frequencies calculated per 10-minute periods, normalized over duration of the session.

Most of these features were taken from the initial feature set; including: numSG, numSGValidated, numNumSGChanged, numSGAttempted, numPLAN and FlagWorkedWithoutSubgoal.

We extended this group by adding features associated with the average time for setting up a new subgoal (AverageTimeSettingSG) and the full time spent setting subgoals during the whole session, which included two initial subgoals and all voluntary subgoals set later (TotalTimeSettingSG). This was to have a sense of how long it took students to set a subgoal, which could indicate the time spent by the student for this process.
5.2.3 **Subgoal-related features for working with content**

Since the students were pursuing certain learning goals when studying the circulatory system with MetaTutor, the reading pages behavior should be assessed in terms of the learning goals. In this subsection, we discuss measures related to studying content with the subgoals set. Primarily, we were interested in how much time students spent on relevant or irrelevant content while they worked with MetaTutor. By working with content, we mean working in the normal layout (Figure 3.2) without any SRL tools activated, or the full layout (Figure 3.6). We combined the actions from these two layouts since we estimated the time that students spend reading in both layouts without SRL tools, except for page relevance. The calculated features are listed in Table 5.3.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>RatioPagesRelevantToInitialSubgoal1ReadWhenSubgoal1Active</td>
<td>Ratio of pages a student opened that were relevant to the first subgoal when it was active.</td>
<td>McGill</td>
</tr>
<tr>
<td>RatioPagesRelevantToInitialSubgoal2ReadWhenSubgoal2Active</td>
<td>Ratio of pages a student opened that were relevant to the second subgoal when it was active.</td>
<td>McGill</td>
</tr>
<tr>
<td>RatioPagesRelevantToInitialSubgoal1ReadAtAnytime</td>
<td>Ratio of pages a student opened that were relevant to the first subgoal during the whole session.</td>
<td>McGill</td>
</tr>
<tr>
<td>RatioPagesRelevantToInitialSubgoal2ReadAtAnytime</td>
<td>Ratio of pages a student opened that were relevant to the second subgoal during the session.</td>
<td>McGill</td>
</tr>
<tr>
<td>TimePageRelSGأخ</td>
<td>Time spent on pages that were relevant to the currently active subgoal.</td>
<td>UBC</td>
</tr>
<tr>
<td>TimePageIrrelSGأخ</td>
<td>Time spent on pages that were irrelevant to the currently active subgoal.</td>
<td>UBC</td>
</tr>
<tr>
<td>ProportionTimeRelSGOverReading</td>
<td>Proportion of time spent on relevant pages over total time spent with content.</td>
<td>UBC</td>
</tr>
<tr>
<td>ProportionTimeIrrelSGOverReading</td>
<td>Proportion of time spent on irrelevant pages over total time spent with content.</td>
<td>UBC</td>
</tr>
<tr>
<td>TotalTimeWithRelTextContentأخ</td>
<td>Total time spent on relevant pages with no images open.</td>
<td>UBC</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Origin</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>TotalTimeWithRelFullContent&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Total time spent on relevant pages with an image open.</td>
<td>UBC</td>
</tr>
<tr>
<td>TotalTimeWithIrrelTextContent&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Total time spent on irrelevant pages with no image open.</td>
<td>UBC</td>
</tr>
<tr>
<td>TotalTimeWithIrrelFullContent&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Total time spent on irrelevant pages with image open.</td>
<td>UBC</td>
</tr>
</tbody>
</table>

<sup>a</sup> Features were normalized over duration of the session.

The initial set from the McGill collaborators introduced features that measured the ratio of pages relevant to both initial subgoals over all pages read during a learning session, and over the time when the corresponding subgoals were active:

\[
\text{RatioPagesRelevantToInitialSubgoal1ReadWhenSubgoal1Active},
\]
\[
\text{RatioPagesRelevantToInitialSubgoal2ReadWhenSubgoal2Active},
\]
\[
\text{RatioPagesRelevantToInitialSubgoal1ReadAtAnytime},
\]
\[
\text{RatioPagesRelevantToInitialSubgoal2ReadAtAnytime}.
\]

We extended this subset by adding a set of features that measures how much time students spent with relevant or irrelevant content in respect to the currently active subgoals (TimePageRelSG or TimePageIrrelSG). We also added proportions (ProportionTimeRelSGOverReading and ProportionTimeIrrelSGOverReading) of time spent on relevant and irrelevant content over the full time spent with content (TotalTimeWithContent). These features were designed to bring into the feature set information about how much time students spend on reading relevant content as it might indicate whether or not students are following the subgoals and succeeding with finding relevant material in MetaTutor.

To get a better understanding of how students work with content, we created four additional measures that account for the relevance of the open pages and the students'
attention to the corresponding visuals. By “attention,” we mean evidence that the student explicitly opened a full-size image by clicking on the thumbnail. These features represent the duration of time spent on pages that are:

(1) relevant to the current subgoal with no image ($\text{TotalTimeWithRelTextContent}$),

(2) relevant to the current subgoal with full-size image or with full-view layout active ($\text{TotalTimeWithRelFullContent}$),

(3) irrelevant to the current subgoal with no image ($\text{TotalTimeWithIrrelTextContent}$),

(4) irrelevant to the current subgoal with full-size image or with full-view layout active ($\text{TotalTimeWithIrrelFullContent}$).

### 5.2.4 Features describing the note-taking

The group of features that described actions related to taking notes is reported in Table 5.4

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoteTakingDuration$^a$</td>
<td>Time spent on taking notes using embedded notepad.</td>
<td>McGill</td>
</tr>
<tr>
<td>NoteCheckingDuration$^a$</td>
<td>Time spent on checking notes in embedded notepad.</td>
<td>McGill</td>
</tr>
<tr>
<td>NoteTakingNum$^b$</td>
<td>Number of times student added or changed notes in embedded notepad.</td>
<td>McGill</td>
</tr>
<tr>
<td>NoteCheckingNum$^b$</td>
<td>Number of times student accessed notes in embedded notepad to review the content without making any changes to notes.</td>
<td>McGill</td>
</tr>
<tr>
<td>TimePaperNotes$^a$</td>
<td>Number of times a student added paper notes.</td>
<td>UBC</td>
</tr>
</tbody>
</table>

$^a$ Features were normalized over duration of the session.

$^b$ Frequencies calculated per 10-minute periods were normalized over the duration of the session.
The original feature set included features for two separate activities when working with notes in the embedded notepad (Figure 3.5), which were reused in our feature set: taking notes (content of the note is changed) and checking notes (no changes in content). For both of these activities, the feature set contained the frequency and total duration.

In addition, we were able to trace the time spent by the student taking notes, or intending to take notes, using the digital notepad. Since we did not have access to the notes that were taken using the digital notepad, we assumed that the student began taking notes when the digital pen was moved, and stopped taking notes when the digital pen was put back (TimePaperNotes).

5.2.5 Features for SRL tools usage

Table 5.5 shows the list of features calculated for the MetaTutor session that were related to using the SRL tools (excluding those that were related to subgoals or to taking notes, as discussed earlier in this chapter). Most of these measures were defined by the McGill collaborators, who mostly counted frequencies for use of each SRL tool.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeINF(^a)</td>
<td>Time spent making inferences.</td>
<td>UBC</td>
</tr>
<tr>
<td>TimePKA(^a)</td>
<td>Time spent reporting prior knowledge.</td>
<td>UBC</td>
</tr>
<tr>
<td>TimeSummary(^a)</td>
<td>Time spent writing summaries.</td>
<td>UBC</td>
</tr>
<tr>
<td>numSUMM(^b)</td>
<td>Number of times student added summaries.</td>
<td>McGill</td>
</tr>
<tr>
<td>numMPTG(^b)</td>
<td>Number of times the student managed their progression toward the current subgoal by assessing their current understanding (therefore leading to a subgoal quiz).</td>
<td>McGill</td>
</tr>
<tr>
<td>numPKA(^b)</td>
<td>Number of times the student was prompted to activate, or activated on their own initiative, prior knowledge about their current subgoal.</td>
<td>McGill</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Origin</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>numJOL&lt;sup&gt;️️️️&lt;/sup&gt;</td>
<td>Number of times the student was prompted to judge, or judged on their own initiative, how well they were learning from the page they were currently viewing.</td>
<td>McGill</td>
</tr>
<tr>
<td>numFOK&lt;sup&gt;️️️&lt;/sup&gt;</td>
<td>Number of times the student was prompted to express, or expressed on their own initiative, their feelings about their knowledge regarding the page they were viewing.</td>
<td>McGill</td>
</tr>
<tr>
<td>numCE&lt;sup&gt;️️&lt;/sup&gt;</td>
<td>Number of times the student was prompted to evaluate, or evaluated on their own initiative, the content of the page they were viewing regarding the subgoal they were working on.</td>
<td>McGill</td>
</tr>
<tr>
<td>numINF&lt;sup&gt;️️&lt;/sup&gt;</td>
<td>Number of times the student took the initiative to make an inference about the content of the page they were viewing.</td>
<td>McGill</td>
</tr>
<tr>
<td>AVGNumSRLperPage toSG1Active</td>
<td>Average number of SRL processes per page when the first subgoal was active.</td>
<td>McGill</td>
</tr>
<tr>
<td>AVGNumSRLperPage toSG2Active</td>
<td>Average number of SRL processes per page when the second subgoal was active.</td>
<td>McGill</td>
</tr>
<tr>
<td>AVGNumSRLpePage toSG1AnyTime</td>
<td>Average number of SRL processes per page.</td>
<td>McGill</td>
</tr>
<tr>
<td>AVGNumSRLpePage toSG2AnyTime</td>
<td>Average number of SRL processes per page.</td>
<td>McGill</td>
</tr>
</tbody>
</table>

<sup>a</sup> Features were normalized over the duration of the session.

<sup>b</sup> Frequencies calculated per 10-minute periods were normalized over the duration of the session.

In addition to the features that were designed by the McGill collaborators, we calculated durations for some SRL processes whenever possible. This included: time spent making inferences (*TimeINF*), reporting prior knowledge (*TimePKA*), and writing summaries (*TimeSummary*). Other processes did not involve any continuous actions. For example, when evaluating their understanding of the material (*numJOL*), the students were asked a multi-choice question that led to a quiz.
Chapter 6  Relevant Machine Learning Techniques

We approached the problem of assessing student learning with MetaTutor as a classification problem. In this chapter, the machine learning techniques used in this study are explained. First, the machine learning algorithms that we used in the experiments and ensemble classifiers are described. Then, the feature selection methods are discussed, and finally, the approaches for validating the results are described.

6.1 Machine learning algorithms

We trained a set of algorithms, available in the WEKA machine learning toolkit [59] using the available dataset. In this thesis, we report only the top five that achieved the highest performance with the dataset. They are: Simple Logistic Regression [60], Multinomial Logistic Regression [61], Naïve Bayes [62], Random Forest [63], and Multilayer Perceptron [64].

Besides training a set of individual learning algorithms, we trained several ensemble classifiers [65]. The central idea for ensemble modeling is to combine the outcome of several classifiers (referred to as base classifiers) to create a classifier that outperforms them.

The mechanism for ensemble modeling in this work is shown in Figure 6.1. Base classifiers take unlabeled data as input and classify it independently. The predicted labels are processed by a combination mechanism that produces the final prediction. Various methods can be used to combine the set of classifiers (e.g., majority voting, performance weighting, entropy weighting, and based on Bayes rule) [65]. We used the majority voting method since empirical evidence in similar research [9] suggests that it boosts the performance of learning algorithms. Majority voting is a simple technique that
assigns a label that receives the greatest number of votes (i.e., predicted by the greatest number of classifiers).

![Figure 6.1: Predicting mechanism in the ensemble model scheme](image)

### 6.2 Feature selection

In this work, we have access to a very limited dataset of about 50 participants after the validation process. The feature set described in the previous two chapters includes more than 100 various measures of attention patterns and interface activities that define the feature space. Nevertheless, this setting can lead to over-fitting when training certain classifiers [66], because it is easier to fit small samples in high dimensional spaces (the “curse of dimensionality” problem [67]). In addition, some features might be bad predictors for learning. To address these problems we tried reducing the space dimension using several techniques.

The first step was to ensure that the created feature set does not have highly correlated features since they do not bring any further information to the model and are redundant [66].
A correlation between features can be detected by using Principal Component Analysis (PCA), which converts originally correlated variables into a set of components corresponding to linearly uncorrelated variables [68]. The great advantage of this approach is a significant reduction in the feature space with little loss of data. The downside of PCA is that the technique creates a new set of features that are not intuitive and may be difficult to interpret.

The other common approach to reducing the features of space dimensionality is to perform feature selection, which is a process of selecting a subset of features to be used in the model training. Two popular techniques have been widely used for feature selection: filters (algorithm-independent feature selection) and wrappers (algorithm-specific feature selection) [66, 69]. These techniques give rise to subsets or rankings of the original features, which is advantageous if one wants to know which features out of all available features, might contribute to the trained models.

Filters are used to rank the features, based on certain measures without specific knowledge about the learning algorithm. Common measures include: information entropy, Pearson's correlation coefficient, and inter-class distance [70]. Filters are faster and computationally more efficient than wrappers since they usually provide feature ranking rather than a subset of features. In contrast, wrappers evaluate feature sets using machine learning algorithms that serve as predictors. Since feature selection should be repeated for each learning algorithm, this approach is more computational intensive than the use of filters.

We tried both wrappers and filters on our feature set and found that wrappers worked better than filters for the algorithms listed in the previous section.

The efficiency of a wrapper depends mostly on the search method used to find the best subset. In an ideal situation the best approach would be to exhaustively go over all possible combinations of features; however, this is only possible for datasets that involve a small number of features, because of the large computations. The most popular approach is the heuristic best first search, with forward selection or backward
selection. In forward selection, features are added to an initially empty set until the best subset is found. In backward selection, the search begins from an initially complete set and features are eliminated until the best subset is achieved. We tried both ways for selecting features but did not find any noticeable difference in their performance. In our work, we used forward selection because it is faster than the backward selection.

6.3 Cross-validation

Obtaining an unbiased estimate of a machine learning algorithm performance is a well known problem. Generally, to get the best estimate of performance, one needs to test the model with unseen data that has not been used in the training set. Nevertheless, this is not feasible for cases with a very limited dataset and the whole data is needed to train the model. The simplest approach to this problem is to use re-sampling techniques, such as k-fold cross-validation [71].

In general, the procedure for k-fold cross-validation is as follows: the whole dataset is randomly divided into k subsets of equal size (i.e., folds); in the training stage, all folds but one are used to train an algorithm. The remaining fold is used for testing. The process is repeated until all folds have been used for testing. Unfortunately, this method of cross-validation suffers from bias and high variance.

The results of cross-validation are often pessimistically biased to a lower estimate. The partitioning into folds may result in differences in the distribution across the folds. To ensure uniform distribution, we used the stratification of folds [71], by randomly sampling an equal number of datapoints from each class to each fold.

From a practical perspective, the high variance of k-fold cross-validation means that the results of two different k-fold cross-validations for the same algorithm with the same dataset can produce different results [72]. The problem with variance can be fixed by performing repeated k-fold cross-validations, where the re-sampling into k folds is
repeated several times. The average accuracy over all runs is reported as the estimated performance of the learning algorithm.

In the literature, the most common configurations for cross-validation are: 10-fold cross-validation with 10 runs, 2-fold cross-validation, and leave-one-out cross-validation. Research has shown that the 10-run 10-fold cross-validation has the lowest bias and lowest variance [71, 73]. Thus, we used this method for validating our results.
Chapter 7  Machine Learning Experiments using Eye-Tracking Data

One of the main research questions in this thesis addresses the value of eye-tracking data for inferring learning. The initial focus was on two different types of features, as described in Chapter 4: interface independent (i.e., can be calculated for any kind of interface) and interface specific (i.e., some knowledge of the interface is needed to define them). These features formed two feature sets: no-AOI and AOI-based. A third (Gaze) feature set was formed by combining the other two.

We trained and compared a set of classifiers using these three sets of gaze-based features. The classifiers evaluated student's learning at the end of the interaction, when the complete data from the learning session was available (i.e., overall accuracy). Furthermore, we looked at the gaze patterns that contribute the most to learning. A variation of this research was accepted as a full paper at AIED2013 (Conference on Artificial Intelligence in Education 2013 in Memphis, TN) [74].

7.1  Dataset and class labels

In our experiments, we used data from 68 students who participated in the study, as described in Chapter 3. Two participants were excluded as they had scored the maximum in the pre-test and thus would not be able to show any traceable learning gain.

We applied pre-processing and data validation, as described in Chapter 4, to the remaining data from 66 students, with 16 participants being filtered out. The final size of the dataset used for this work was 50 participants.

We defined two classes of learning performance based on the median split of proportional learning gain. Proportional learning gain (PLG) is a measure of the
proportion of a student’s test score gain over the highest possible improvement from pre-test to post-test. PLG is calculated with Equation 7.1 [75].

\[
PLG = \begin{cases} 
\frac{post-pre}{max-pre}, & post - pre > 0 \\
\frac{post-pre}{pre}, & otherwise 
\end{cases}
\] (7.1)

We divide the sample into High-Learners (HL) and Low-Learners (LL), based on the median split over PLG. This resulted in 25 LL and 25 HL. The descriptive statistics for this split are shown in Table 7.1. An independent samples t-test revealed a significant difference between these two classes (t(48) = 11.09, p < 0.0001, Cohen’s d = 3.14).

<table>
<thead>
<tr>
<th></th>
<th>Number of students</th>
<th>Mean</th>
<th>Median</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL</td>
<td>25</td>
<td>13.11</td>
<td>9.09</td>
<td>17.11</td>
</tr>
<tr>
<td>HL</td>
<td>25</td>
<td>66.15</td>
<td>60</td>
<td>16.70</td>
</tr>
<tr>
<td>Total</td>
<td>50</td>
<td>39.63</td>
<td>41.48</td>
<td>31.58</td>
</tr>
</tbody>
</table>

Our dataset includes the full set of participants from the original MetaTutor study, ignoring the two different conditions (non-adaptive and adaptive, described in Chapter 3). To justify this decision, we analyzed the learning difference between participants who were exposed to these two versions of MetaTutor. For the purpose of this thesis, we compared the mean PLG for adaptive (mean PLG = 42.92, SD = 24.98) and non-adaptive (mean PLG = 36.34, SD = 37.28) conditions to ensure that learning performance was not biased towards any of them. The independent samples t-test of over 50 valid participants showed no significant difference between the means of these two groups (t(48) = 0.73, p = 0.47, Cohen’s d = 0.21). Thus, for the purpose of this
analysis, the data for the adaptive and non-adaptive conditions were collapsed without reducing the reliability of the experiments.

### 7.2 Data preparation

A large number of features can lead to over-fitting when only relatively small datasets are available for training. To avoid this problem, we first removed all highly correlated features from each of the three eye-tracking sets (using Pearson’s $|r| = 0.9$ as the threshold [76]). We did this because no new information is gained by adding them to the training set [66].

Second, we reduced the number of features in the two larger feature sets (*AOI*-based and *Gaze*) by performing wrapper feature selection [66], as described in Section 6.2. To further reduce the likelihood of over-fitting, the feature selection process was cross-validated. The final sets of features were obtained by keeping only the top 10 features that were selected most often by the feature selection process.

We chose the threshold of the top 10 features as follows. To guarantee that automated feature selection would result in a feature set that contained predictive features and would not cause over-fitting, we ran a preliminary analysis to estimate the number of features to be used in the final classification. We set up different thresholds for the number of selected features (from 5 to 20 features) and evaluated the performance of the corresponding classifiers. From the results, we chose a threshold of 10 features since the size of the feature set gave the best performance, on average, for all learning algorithms and feature sets. Moreover, this size of feature set is in-line with general recommendations for using feature sets, which suggest that they should be 5-10 times smaller than the size of the training set [77].
7.3 Discussion of results

We used the WEKA data mining toolkit to train a variety of classifiers in the feature selection for our three feature sets: Gaze, AOI-based, and no-AOI. The results are summarized in Table 7.2. All of the results reported here are based on 10-fold cross-validation, with 10 runs per fold, and pertaining to the 5 best performing classifiers among the ones we tested (Simple Logistic Regression, Multinomial Logistic Regression, Naïve Bayes, Random Forest, and Multilayer Perceptron). For each feature set (no-AOI, AOI-based, and Gaze), we report: overall accuracy (percentage of data points correctly classified), accuracy in each class (LL and HL), and kappa scores (another commonly used measure of accuracy that accounts for agreement due to chance; thus, often making it more robust than standard measures) [78]. The evaluation of kappa was performed using the following guidelines [79]: kappa < 0.2 is considered to be poor, 0.21-0.4 is fair; 0.41-0.6 is moderate; and >0.61 is good.

To ascertain the impact that different feature sets have on classification performance, we performed two, two-way ANOVA with feature set (3 levels) and classifiers (5 levels) as factors, for both overall accuracy and kappa scores. The two analyses generated analogous results, thus, here we discuss only the results for overall accuracy, because they are easier to interpret in terms of practical classification performance. Mauchly’s Test of Sphericity indicated that the assumption of sphericity was violated for the main effect of feature set ($\chi^2(2)=7.71$, $p=0.021$) and for the interaction of feature set and classifier effect ($\chi^2(35)=65.68$, $p=0.004$). We report results after the corresponding Greenhouse-Geisser corrections have been applied.
<table>
<thead>
<tr>
<th>Algorithm name</th>
<th>Overall accuracy (%)</th>
<th>Class accuracy (%)</th>
<th>Kappa</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>SD</td>
<td>Mean LL</td>
</tr>
<tr>
<td><strong>no-AOI Feature Set</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple Logistic Regression</td>
<td>50.4</td>
<td>3.77</td>
<td>53.2</td>
</tr>
<tr>
<td>Multinomial Logistic Regression</td>
<td>54.4</td>
<td>4.27</td>
<td>52</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>54.2</td>
<td>2.60</td>
<td>42.4</td>
</tr>
<tr>
<td>Random Forest</td>
<td>54</td>
<td>3.79</td>
<td>55.2</td>
</tr>
<tr>
<td>Multilayer Perceptron</td>
<td>53</td>
<td>3.38</td>
<td>51.2</td>
</tr>
<tr>
<td><strong>AOI-based Feature Set</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple Logistic Regression</td>
<td>62.2</td>
<td>3.03</td>
<td>50.8</td>
</tr>
<tr>
<td>Multinomial Logistic Regression</td>
<td>70.4</td>
<td>4.45</td>
<td>64.4</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>69.8</td>
<td>3.52</td>
<td>69.2</td>
</tr>
<tr>
<td>Random Forest</td>
<td>68</td>
<td>5.73</td>
<td>76</td>
</tr>
<tr>
<td>Multilayer Perceptron</td>
<td>61.6</td>
<td>5.04</td>
<td>57.6</td>
</tr>
<tr>
<td><strong>Gaze Feature Set</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple Logistic Regression</td>
<td>81.2</td>
<td>2.23</td>
<td>79.2</td>
</tr>
<tr>
<td>Multinomial Logistic Regression</td>
<td>68.6</td>
<td>2.20</td>
<td>69.2</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>77</td>
<td>2.86</td>
<td>63.6</td>
</tr>
<tr>
<td>Random Forest</td>
<td>66.6</td>
<td>4.98</td>
<td>76.8</td>
</tr>
<tr>
<td>Multilayer Perceptron</td>
<td>69.6</td>
<td>3.67</td>
<td>68</td>
</tr>
</tbody>
</table>

Figure 7.1 shows the mean overall accuracy for each combination of classifier and feature set. Significant main effects are seen for both classifier (F(4, 36) = 8.67, p<0.001, ηp² = 0.49) and feature set, (F(1.23, 11.12) = 236.16, p<0.001, ηp² = 0.96), further qualified by a significant interaction between factors, F(3.50, 31.48) = 13.93,
p<0.001, ηp²=0.61), showing that classifier type influences the relative accuracy that can be achieved with each feature set. We performed planned contrast analysis (with corresponding Bonferroni adjustments) to gain a better understanding of the relative value of AOI-dependent and AOI-independent features. On average, this analysis shows that the performance of the classifiers that were trained on the Gaze feature set is significantly better than those trained on AOI-based features (t(31.48) = 5.05, p<0.001, Cohen’s d = 1.80). In turn, the latter classifiers perform better than those trained on no-AOI (t(31.48) = 10.76, p<0.001, Cohen’s d = 3.83). In particular, the highest overall accuracy was achieved by Simple Logistic Regression on the Gaze dataset (81.2%, kappa = 0.62), which also shows a good balance in class accuracy (79.2% on LL and 83.2% on HL) (Table 7.2).

![Figure 7.1: Overall accuracy of the 5 best performing algorithms over 3 gaze feature sets](image)

We see this result as strong evidence for the value of eye-tracking data as a source of rich information about student modeling, because it shows that gaze information can be a good predictor of student learning, even before taking into account other student...
interaction behaviors (e.g., interface actions). As discussed in the Introduction, this result seems to generalize across at least some learning environments that are different in nature, because similar accuracies were found in [4].

Simple Logistic Regression of the Gaze dataset performs significantly better ($t(31.48)=3.93$, $p<0.01$, Cohen’s $d =1.40$) than the best performing classifier for AOI-based features, namely Multinomial Logistic Regression (70.4% accuracy, kappa = 0.41). This classifier is also quite unbalanced in terms of class accuracy (64.4% for LL, and 76.4% for HL), indicating that AOI-independent features have considerable added value when combined with AOI-dependent ones, though on their own, they do not perform that well.

### 7.4 Analysis of eye-tracking features

For the Simple Logistic Regression classifier, which showed the best overall accuracy for the Gaze feature set, the 10 features selected for training models included one AOI-independent feature (mean of relative path angles) and 9 AOI-dependent features. The latter included rates for fixations on Text and Image Content and 7 features representing transitions between AOIs (full list of transitions is shown in Table 7.3):

| Table 7.3: Selected features for Simple Logistic Regression trained on Gaze |
|-------------------------------------|-----------------------------------------------|
| **General**                         | meanrelpathangles                             |
| **Attention to specific AOIs**      | TextContent_fixationrate,                     |
|                                     | ImageContent_fixationrate,                    |
| **Transitions between AOIs**        | Subgoals_proptransfrom_TableOfContents        |
|                                     | LearningStrategiesPalette_proptransfrom_Subgoals |
|                                     | TableOfContents_proptransfrom_TableOfContents |
|                                     | TextContent_proptransfrom_Agent,              |
|                                     | TextContent_proptransfrom_ImageContent,       |
|                                     | TextContent_proptransfrom_TableOfContents,    |
|                                     | ImageContent_proptransfrom_Subgoals           |
In this classifier, the following AOIs appeared most often in the 7 selected transitions: Text (3 features), Table of Contents (3 features), Subgoals (3 features), and Image Content (2 features). On top of this, information on Attention to Agent and Learning Strategies Palette AOIs was supported by one feature for each of these two AOIs (proportion of transitions from Agent to Text Content and proportion of transitions from Learning Strategies Palette to Subgoals).

The fact that five of the ten selected features are related to AOIs that represent SRL tools (Subgoals, Agent, Learning Strategies Palette) suggests that attention to these elements is indeed important for assessing learning with MetaTutor.

The mean and standard deviations for all features for two classes are shown in Table 7.4.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>LL</th>
<th></th>
<th></th>
<th></th>
<th>HL</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>SD</td>
<td>Mean</td>
<td>SD</td>
<td>Mean</td>
<td>SD</td>
<td></td>
</tr>
<tr>
<td>meanrelpathangles</td>
<td>2.085875</td>
<td>0.095586</td>
<td>2.122968</td>
<td>0.076163</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subgoals_proptransfrom_TableOfContents</td>
<td>0.119372</td>
<td>0.096334</td>
<td>0.057998</td>
<td>0.040304</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LearningStrategiesPalette_proptransfrom_Subgoals</td>
<td>0.009902</td>
<td>0.014914</td>
<td>0.009155</td>
<td>0.013676</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TableOfContents_proptransfrom_TableOfContents</td>
<td>0.631967</td>
<td>0.151684</td>
<td>0.716435</td>
<td>0.067195</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextContent_fixationrate</td>
<td>0.002843</td>
<td>0.000414</td>
<td>0.002765</td>
<td>0.000324</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextContent_proptransfrom_Agent</td>
<td>0.000946</td>
<td>0.001038</td>
<td>0.001269</td>
<td>0.001393</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextContent_proptransfrom_ImageContent</td>
<td>0.017273</td>
<td>0.008814</td>
<td>0.020364</td>
<td>0.011704</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextContent_proptransfrom_TableOfContents</td>
<td>0.027767</td>
<td>0.034548</td>
<td>0.015635</td>
<td>0.009317</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ImageContent_fixationrate</td>
<td>0.003089</td>
<td>0.000471</td>
<td>0.003346</td>
<td>0.000489</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ImageContent_proptransfrom_Subgoals</td>
<td>0.005611</td>
<td>0.00714</td>
<td>0.00762</td>
<td>0.008211</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A closer look at the selected features reveals that HL students showed a higher rate of transitions from Agent to Text Content and from Subgoals to Image Content. On average, HL also showed a higher transitions rate within Table of Contents. LL showed a higher rate of transitions from Subgoals to Learning Strategies Palette and from Table of Contents to Text Content. LL also had a lower rate of meanrelpathangles, compared to HL. The rate of fixations on Text Content was slightly higher for LL; however, the rate of fixations on Image Content was higher for HL, which implies that HL are more attentive to the visuals in MetaTutor.

The strongest difference between the two classes was observed in the transitions between Table of Contents and Subgoals. We found that students in the LL class showed a larger proportion of transitions from Table of Contents to Subgoals, compared to the transitions for HL (0.1193 and 0.0580, for LL and HL, respectively). Thus, low learners may re-evaluate the relevance of the content to subgoals more often than high learners (e.g., when selecting the next page to study, they look at the subgoals to see if the next page still matches their current subgoals). In turn, this may signify that low learners are less confident in their plans for a learning session or do not sufficiently plan their learning during the planning phase.

To get a better understanding of the underlying processes, we looked at the other measures that describe attention to subgoals, which had been discarded by feature-selection from the current model. We found some strong differences between the two classes. The longest fixation in subgoals for HL (mean = 1056.56 ms, SD = 422.72) was higher than for LL (mean = 816.20, SD = 268.84). Another strong difference was seen in the proportional number of fixations that were traced in Subgoals: HL tended to focus more on Subgoals (mean = 0.0217, SD=0.0081), compared to LL (mean = 0.0149, SD=0.0092). Furthermore, HL students tended to have more transitions within the Subgoals AOI (mean = 0.5813, SD = 0.1050), compared to LL (mean = 0.5058, SD = 0.1371). These results suggest that LL and HL have different approaches to working on subgoals. While LL pay less attention to subgoals themselves, HL tend to focus on subgoals independently from other tools or available content.
Chapter 8  Machine Learning Experiments with Full Data

The goal of the experiments we describe in this chapter is to compare the value of eye-tracking data and action logs in predicting learning at the end of interactions, when full interaction data is available (i.e., overall performance) and during the learning session with MetaTutor (i.e., performance over time). We used data from the study described in Chapter 3.

Following the same approach we used in Chapter 7, we trained five classification algorithms (Simple Logistic Regression, Multinomial Logistic Regression, Naive Bayes, Random Forest, and Multilayer Perceptron) on three different feature sets that incorporated eye-tracking data and action data separately and in combination (Actions, Gaze, and Full). We also trained several ensemble models, combining a selection of base models with the majority voting scheme.

Finally, we simulated the online learning performance for the winning classifiers by dividing the interaction into progressively longer segments of relevant interaction data and feeding them incrementally to the trained model.

In Section 8.1, we describe the feature sets that were used. Section 8.2 compares overall performance of the 15 base models. In section 8.3, we discuss the performance of the ensemble models. In section 8.4, we evaluate over time the performance of the selected best models. In the last section, we discuss our experimental results.

8.1  Feature sets

As discussed in Chapter 7, the feature set combining AOI-based and no-AOI features, generally achieved a better performance than when they are used separately. Thus, we henceforth consider the combination of interface-independent and interface-specific
gaze features (AOI-based and no-AOI) and refer to this set as the Gaze feature set. Here, we use the same Gaze models that were described in the previous chapter. The Actions feature set is used as was described in Chapter 5. And finally, we define the Full feature set as the union of the features from Gaze and Actions.

Like the experiments with the eye-tracking data, described in the previous chapter to address the problem of over-fitting, we applied several steps to reduce the number of features used for training classifiers by removing highly correlated features from each of the three sets (using Pearson’s |r| = 0.9 as a threshold [76]) and then running a wrapper feature selection with 10-fold cross-validation.

### 8.2 Overall performance of the basic models

In this section, we analyze the prediction performance of different models at the end of the interaction when complete data is available. We trained a set of classifiers on the data from the complete interaction with MetaTutor using the WEKA machine learning toolkit [59]. We refer to these 15 models (5 algorithms on 3 feature sets) as the base models. The overall accuracy, variance, kappa score, and class accuracy for two classes (LL and HL) are reported in Table 8.1.
Table 8.1: Overall accuracies and Kappa scores for base models

<table>
<thead>
<tr>
<th>Algorithm name</th>
<th>Overall accuracy, %</th>
<th>Class accuracy, %</th>
<th>Kappa</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>SD</td>
<td>Mean LL</td>
</tr>
<tr>
<td><strong>Actions Feature Set</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple Logistic Regression</td>
<td>63.8</td>
<td>5.17</td>
<td>64.0</td>
</tr>
<tr>
<td>Multinomial Logistic Regression</td>
<td>60.8</td>
<td>3.60</td>
<td>59.2</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>64.2</td>
<td>3.52</td>
<td>60.8</td>
</tr>
<tr>
<td>Random Forest</td>
<td>72.0</td>
<td>2.97</td>
<td>74.0</td>
</tr>
<tr>
<td>Multilayer Perceptron</td>
<td>73.8</td>
<td>2.60</td>
<td>71.2</td>
</tr>
<tr>
<td><strong>Gaze Feature Set</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple Logistic Regression</td>
<td>81.2</td>
<td>2.23</td>
<td>79.2</td>
</tr>
<tr>
<td>Multinomial Logistic Regression</td>
<td>68.6</td>
<td>2.20</td>
<td>69.2</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>77.0</td>
<td>2.86</td>
<td>63.6</td>
</tr>
<tr>
<td>Random Forest</td>
<td>66.6</td>
<td>4.98</td>
<td>76.8</td>
</tr>
<tr>
<td>Multilayer Perceptron</td>
<td>69.6</td>
<td>3.66</td>
<td>68.0</td>
</tr>
<tr>
<td><strong>Full Feature Set</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple Logistic Regression</td>
<td>81.4</td>
<td>4.39</td>
<td>80.8</td>
</tr>
<tr>
<td>Multinomial Logistic Regression</td>
<td>74.6</td>
<td>2.54</td>
<td>77.6</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>75.0</td>
<td>2.41</td>
<td>66.0</td>
</tr>
<tr>
<td>Random Forest</td>
<td>76.0</td>
<td>4.98</td>
<td>82.0</td>
</tr>
<tr>
<td>Multilayer Perceptron</td>
<td>80.8</td>
<td>2.40</td>
<td>82.4</td>
</tr>
</tbody>
</table>

The general trends in accuracies across classifiers are shown in Figure 8.1. Models built on Gaze and Actions fluctuate widely, while the Full models seem to be more stable. To validate the significance of these results, we performed a repeated measures two-way (3 by 5) ANOVA with feature set and learning algorithm as independent factors and class accuracies as dependent variables. Mauchly's Test of Sphericity indicated that the assumption of sphericity was violated for the main effect of feature set ($\chi^2(2)=8.18$, $p=0.017$) and for the interaction of feature set and classifier effect.
(χ²(35)=74.88, p<0.001). We report results with corresponding Greenhouse-Geisser corrections applied.

![Figure 8.1: Overall accuracy of the 5 algorithms over 3 feature sets](image)

Significant main effects on occurring on both learning algorithm (F(4, 36) = 20.01, p < 0.001, ηp²=0.69) and feature set (F(2, 10.97) = 115.48, p < 0.001, ηp²=0.93). We also found significant interaction effect (F(3, 26) = 21.34, p < 0.001, ηp²=0.70) between factors. This interaction is reflected in the trends shown in Figure 8.1. In the next two subsections, we analyze these effects in more detail. Main effects of feature set and learning algorithm

To illustrate the main effect found for feature set, Figure 8.2 shows prediction accuracies for each set averaged over the five selected learning algorithms.
Further statistical analyses (planned contrasts with corresponding Bonferroni adjustments) revealed that the *Full* (average accuracy 77.56%) feature set significantly outperforms the *Gaze* (average accuracy 72.6%) feature set ($t(26) = 4.10, p < 0.001$, Cohen’s $d = 1.61$), which in turn, outperforms the *Actions* (average accuracy 66.92%) feature set ($t(26) = 8.80, p < 0.001$, Cohen’s $d = 3.45$).

These results indicate that if both gaze and actions data is available, using their combination for training classifiers is a more reliable approach. Nevertheless, the data in Table 8.1 and Figure 8.1 suggests that classifiers on *Gaze* can achieve similar accuracies if a good algorithm is selected. This is discussed further in the next session.

The results for the main effect for feature set support the findings of Kardan and Conati [9], where the models, based on the combined feature set, outperformed *Actions* and *Gaze*. The main difference between their results and our results is that in their experiments the best performing models, based on *Actions*, performed as good or better than the *Gaze*-based model, while in our experiments, the *Gaze* data-based models performed significantly better than *Actions* in assessing learning with MetaTutor. Another difference is that the best *Gaze*-based classifier (Simple Logistic Regression) achieved a comparable prediction accuracy (~81%) to the models based on the *Full*
feature set, which was not observed by Kardan and Conati. A possible explanation may be the different types of interaction in the two learning environments used in the studies. In MetaTutor, a substantial part of the activities that students learn is reading. In the CSP applet [9]; however, several actions allow students to trace the arc-consistency algorithm. Thus, while students still have to look at the outcomes of the tracing, the main dynamics of the interaction is defined by how they use the tracing actions.

Regarding the main effect of the algorithm, Figure 8.3 shows that, on average, Simple Logistic Regression (75.47%) and Multilayer Perceptron (74.73%) show the best performance of all three feature sets (t(26)= 2.51, p<0.01, Cohen’s d = 0.99).

![Figure 8.3: Main effect of learning algorithm](image)

### 8.2.1 Interaction effect

Three of the fifteen base models achieved overall accuracy close to 81%. These included Simple Logistic Regression trained using the Gaze feature set and the Full feature set, as well as Multilayer Perceptron trained using the Full feature set. All of these showed a good balance in predicting LL and HL (Figure 8.4). Balance is one of
the key factors in assessing model performance and should be taken into account by designers of an ITS when developing personalized interactions targeted to student-specific needs. With well-balanced models and accurate overall performance, the personalized interaction can be targeted to both groups equally.

Since Simple Logistic Regression on the Full feature set achieves slightly better balance than the other two classifiers between LL (80.8%) and HL (82%) overall, we used it for comparisons with overall performance of the other classifiers, shown later in this chapter and henceforth refer to it as the Best Base Classifier.

![Figure 8.4: Balance in predicting LL and HL of the three best performing classifiers](image)

Depending on the type of personalized feedback that a system provides, one could also focus on a classifier with a better performance for a target group of learners. For example, predicting LL might be the focus of an intelligent tutor because LL that are detected early can get feedback or hints from the system to improve their experience with the ITS. In this case, it might be acceptable to use a classifier with a better level of
predicting LL individuals as the target group, even though this would increase the chance of misclassifying HL as LL, to provide an unnecessary tutorial intervention. If we decide to target LL with our adaptation, we could consider Multilayer Perceptron trained on the Full Feature set as the best base classifier since it is slightly better in predicting LL (82.4%) than HL (79.2%) (Figure 8.4). In the case of targeting HL (e.g., providing more challenging and interesting tasks) Simple Logistic Regression trained on the Gaze features would work better since it achieves 83.2% in predicting HL and 79.2% in predicting LL. Thus, depending on the target group, these classifiers might be more preferable over the more balanced, Best Base Classifier.

Considering the remaining classifiers from the initial set of 15 classifiers we trained, Random Forest on the Full feature set is as good in predicting LL (82%) as the best performing classifiers (t(9) = 0.41, p=0.69, Cohen’s d = 0.23), though it is significantly worse in predicting HL (70%)(t(9) = 3.88, p =0.004 , Cohen’s d = 1.94). Similarly, Naive Bayes on the Gaze feature set shows great performance for HL (90.4%), which is the highest performance for this class over all models (t(9) = 4.32, p = 0.002, Cohen’s d = 1.91) but it fails to predict LL (63.6%). Classifiers of this kind could be used for providing feedback targeted to one group, as discussed above. For example the former classifier would work well to help LL, while the latter classifier would be more suitable for targeting HL. In any case, the significant difference in performance between the two classes means that many false positive predictions would occur for the other class with the lower performance. Feedback based on these unbalanced assessments of performance can lead to negative outcomes for misclassified students (e.g., HL might feel bored when provided with adaptations designed for LL). This could worsen the experience with the learning environment. When designing personalized adaptation, the outcomes from misclassifying students need to be carefully considered.

Even though the Gaze models, on average, do not outperform models that are based on Full data, Simple Logistic Regression on the Gaze feature set performs as well as the best performing Full data-based models (Simple Logistic Regression and Multilayer Perceptron) overall. From these findings, we conclude that the Gaze data is a valuable
source for assessing students’ overall learning. Moreover, the Gaze data outperforms Actions for MetaTutor. Nevertheless, the classifiers based on the Full feature set are more stable in their overall performance, regardless of the algorithm used. Using Full for a real-world application might be safer than using Gaze, if the designer of the ITS has no prior knowledge about which algorithm performs better. This partially answers the questions about the value of eye-tracking data when interacting with MetaTutor.

### 8.3 Overall performance of ensemble models

Several successful attempts have been made to use ensemble classifiers in student modeling [9, 80, 81]. Ensemble models combine several base classifiers to create a new one with stronger predictions [65]. In this thesis, we trained a set of ensemble models that used different combinations of base classifiers, as described in the previous section.

The approaches to combining classifiers include:

i. **Combining feature sets.** We trained five separate ensemble models (one for each of the five learning algorithms from the previous section), where the base models differed in the dataset used for training (Gaze, Actions, or Full). This approach has been successfully used [9] and showed significant improvement in performance.

ii. **Combining learning algorithms** over all feature sets. The analysis of main effects revealed a significant effect of the feature set. To make use of the superiority of one feature set over others (Full over the other two, and Gaze over Actions), we trained three separate ensemble models (one for each of the three feature sets), where the base models differed in the learning algorithm used for training.

iii. **Combining the best performing models** from the 15 base classifiers. We combined three and five of the best base classifiers that achieved the highest accuracy overall (Table 8.1) and trained two ensemble classifiers.

Table 8.2 summarizes the best performing ensemble classifiers from each of the three approaches described above. For each approach, we report all models that showed
comparable or better than average overall performance than the best base classifier. Table 8.2 includes: (1-2) two ensembles that correspond to Simple Logistic Regression and Multilayer Perceptron with grouping over feature sets (Actions, Gaze, and Full); (3) an ensemble that combined the three best performing algorithms on the Full feature set (Simple Logistic Regression, Multilayer Perceptron, and Random Forest); and (4-5) two ensembles based on the top three and the top five best performing classifiers over the fifteen base models. We added line (6) with the Best Base Classifier (Simple Logistic Regression trained with Full feature set) to Table 8.2 for comparison.

### Table 8.2: Overall accuracies and Kappa scores for ensemble classifiers

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Overall accuracy, %</th>
<th>Class accuracy, %</th>
<th>Kappa</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Model</td>
<td>Mean</td>
<td>SD</td>
<td>LL</td>
</tr>
<tr>
<td></td>
<td><strong>Combining feature sets</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Simple Logistic Regression</td>
<td>80.4</td>
<td>3.32</td>
<td>80.0</td>
</tr>
<tr>
<td>2</td>
<td>Multilayer Perceptron</td>
<td>80.4</td>
<td>1.50</td>
<td>77.2</td>
</tr>
<tr>
<td></td>
<td><strong>Combining learning algorithms</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Ensemble, Full</td>
<td>84.2</td>
<td>2.79</td>
<td>84.40</td>
</tr>
<tr>
<td></td>
<td><strong>Combining the best performing models</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>BestEnsemble3</td>
<td>86.4</td>
<td>2.8</td>
<td>86.40</td>
</tr>
<tr>
<td>5</td>
<td>BestEnsemble5</td>
<td>82.8</td>
<td>2.99</td>
<td>78.80</td>
</tr>
<tr>
<td></td>
<td><strong>Baseline</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Best Base Classifier</td>
<td>81.4</td>
<td>4.39</td>
<td>80.8</td>
</tr>
</tbody>
</table>

Figure 8.5 shows overall accuracies for the best performing ensembles. Combining models over feature set did not improve the performance of the base classifiers. The other two approaches; however, showed an improvement over the best base classifiers. The best performance was achieved by BestEnsemble5 that combines three best performing classifiers from the base classifiers (Simple Logistic Regression on Full and Gaze; and Multilayer Perceptron on Full feature set).
A repeated measures one-way ANOVA with a model type as a factor (6 levels) showed a significant difference between models ($F(5,45) = 8.24$, $p<0.001$, $\eta^2_p = 0.478$). As a follow-up, we ran a set of pair-wise comparisons with corresponding Bonferroni adjustments to compare the best base classifier with the other five models.

In the rest of this section, we discuss each of the approaches to combining models separately. We begin by looking at combining base models by feature set (classifiers (1) and (2) in Table 8.2). This approach to combining classifiers has been successfully used [9]: the weaker classifier was “voted out” by the other two.

Figure 8.6 compares performance of ensembles based on Simple Logistic Regression and Multilayer Perceptron with the best base classifier. They both achieved an accuracy of 80.40% overall. The Simple Logistic Regression accuracy is also very balanced (80.0% in LL prediction and 80.8% in HL prediction). We did not find any improvement with this approach over the base classifier, using our data ($t(9)=1.16$, $p = 0.27$, Cohen’s $d = 0.23$).
In general, an ensemble model performs the best if each of its base classifiers fails for a particular group of data, so that most of the remaining classifiers assess without errors. Unfortunately, in our case, the classifiers built by combining learning algorithms over the three feature sets tended to agree in predictions (both correct and incorrect). For example, in the case of the Simple Logistic Regression ensemble, the Full and Gaze base classifiers agreed in predictions far more often than the Action base classifier agreed with any of these. Since the Action base classifier showed the lowest performance among the three base models, it does lead to much improvement when the other two classifiers disagree.

The ensemble, based on the three best performing base classifiers trained on the Full feature set (classifier (3) in Table 8.2) showed an improved performance (84.2%) over the best base classifier (81.4%), though it was not statistically significant (t(9) = 2.49, p = 0.20).

The performance of the ensembles that combined the top three (BestEnsemble3) and top five (BestEnsemble5) (classifiers (4) and (5) in Table 8.2) best performing
algorithms of the fifteen base models is shown in Figure 8.7. *BestEnsemble3* is a combination of Simple Logistic Regression on *Full* and *Gaze* sets and Multilayer Perceptron on *Full*. *BestEnsemble5* included the same three classifiers that were used in *BestEnsemble3* and Naïve Bayes on *Gaze* and *Full*.

![Figure 8.7: Comparison of performance for best performing base models and ensemble models](image)

Combination of the three best performing classifiers is significantly better than the best performing base classifier, with Bonferroni adjustments applied ($t(9) = 3.41$, $p = 0.042$, Cohen’s $d = 1.28$). The accuracies for predicting LL and HL are 86.4%. The good balance in *BestEnsemble3* can be explained by the fact that all base models used to construct this ensemble classifier are very balanced. *BestEnsemble5* shows a poor balance, compared to *BestEnsemble3*, and does not significantly outperform overall the best base classifier ($t(9)=1.05$, $p = 0.32$, Cohen’s $d = 0.35$). This can be explained by the addition of two very unbalanced classifiers (Naive Bayes on *Full* and *Gaze* models) that tended to label more students as HL (overall HL class accuracies of 84% and 90.4%, respectively).
8.3.1 Action-based ensemble model

We trained ensemble models, based on the best performing classifiers trained on Actions features only, despite their low performance, compared to those trained on Gaze and Full feature sets to see if combining several algorithms would boost the performance of our base models, when only data on interface actions is available. We did this because the user's actions are easiest to use for collecting a source of information about the user. This is crucial for understanding its value in terms of predicting learning. Table 8.3 reports the prediction accuracy for an ensemble classifier built by combining the three best performing classifiers on the Actions feature set (Multilayer Perceptron, Naïve Bayes, and Random Forest).

Although the ensemble model (average accuracy of 69.40%) shows an increased performance over the average accuracy of base classifiers trained on Actions (66.92%), it does not outperform Multilayer Perceptron (73.8%), which is the best performing classifier on the corresponding feature set.

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Overall accuracy, %</th>
<th>Class accuracy, %</th>
<th>Kappa</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>SD</td>
<td>LL</td>
</tr>
<tr>
<td>Actions</td>
<td>69.40</td>
<td>3.35</td>
<td>64.40</td>
</tr>
</tbody>
</table>

8.3.2 Conclusions on ensemble modeling

We trained a set of ensemble classifiers using several different approaches for combining base algorithms. First, we followed an earlier scheme [9] and created five different ensemble models, each combining three base classifiers with the same learning algorithm trained over the three different feature sets, but this approach did not work well using our dataset. We tried taking advantage of the Full feature set by grouping into an ensemble classifier, three best performing base classifiers trained on Full data, but this approach also did not significantly improve the performance, compared to the best base classifier (Simple Logistic Regression trained on Full feature...
set). The most promising ensemble was a combination of the three best performing models from the set of base classifiers, as it achieved an overall accuracy of 86.40%. This result was significantly better than the performance achieved by the best base classifier. The best ensemble was also very balanced in evaluating LL and HL.

We also explored the value of the ensemble approach for Actions-based models to see if the approach would improve the performance of classifiers, when Gaze data is not available. Nevertheless, we did not find any significant improvement over the base classifiers, based on Actions.

8.4 Accuracy over time

In this section, we investigated the performance of the proposed classifiers over time; i.e., as a function of the amount of observed interaction data. We wanted to see if the trained classifiers could be used “on the fly” to provide personalized feedback for the learner. To address this issue, we simulated an online learning session by dividing the data into chunks of equal time intervals (2 min) and feeding them incrementally to the trained classifiers.

We simulated online learning for four base classifiers and the winning ensemble model (BestEnsemble3), referred to as Ensemble in this section. The subset of base classifiers includes 3 best performing classifiers over all 15 base models (Simple Logistic Regression trained on the Gaze feature set and the Full feature set, and Multilayer Perceptron based on the Full feature set) and the best performing classifier over the Actions feature set. Even though Multilayer Perceptron on Actions showed low performance, in comparison to other winning classifiers, we also simulated online learning for this classifier to see how fast it reaches a reasonable performance (higher than chance, 50%). With this simulation, we wanted to check for a potential benefit from using actions in a real-time session for MetaTutor to evaluate students’ learning. The
results for average accuracy over time for the learning algorithms are summarized in Table 8.4 and visualised in Figure 8.8.

### Table 8.4: Accuracies and Kappa scores over time for selected models

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Feature set</th>
<th>Average</th>
<th>SD</th>
<th>LL</th>
<th>HL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Logistic Regression</td>
<td>Gaze</td>
<td>67.77</td>
<td>12.03</td>
<td>73.15</td>
<td>62.38</td>
</tr>
<tr>
<td></td>
<td>Full</td>
<td>63.82</td>
<td>10.16</td>
<td>59.48</td>
<td>68.15</td>
</tr>
<tr>
<td>Multilayer Perceptron</td>
<td>Actions</td>
<td>55.88</td>
<td>7.78</td>
<td>66.84</td>
<td>44.91</td>
</tr>
<tr>
<td></td>
<td>Full</td>
<td>63.92</td>
<td>10.83</td>
<td>41.92</td>
<td>85.93</td>
</tr>
<tr>
<td>Ensemble</td>
<td></td>
<td>67.75</td>
<td>12.61</td>
<td>62.13</td>
<td>73.38</td>
</tr>
</tbody>
</table>

Figure 8.8: Average accuracy over time for selected classifiers

Repeated measures for a one way ANOVA with model type (five models listed in Table 8.4) as the independent variable and performance over time as the dependent variable were performed. This test revealed significant differences in the performance \( \text{F}(1.62, 160.30) = 135.01, p < 0.0001, \eta_p^2 = 0.58 \). The pair-wise comparisons with Bonferroni
adjustments revealed that Simple Logistic Regression on Gaze data performed significantly better than Simple Logistic Regression and Multilayer Perceptron trained using Full feature set ($t(99) = 10.35$, $p < 0.001$, Cohen's $d = 0.35$) and was not significantly different from the Ensemble model ($t(99) = 0.05$, $p = 0.96$, Cohen’s $d = 0.0012$). Nevertheless, when evaluating performance of a classifier over time one should also look at how balanced it is during the session and how early in the session the classifier can distinguish the target groups of students.

In the following subsections, we discuss the performance of the selected models, features used to build classifiers, and the classifiers' behavior.

### 8.4.1 Performance of Simple Logistic Regression on Gaze data over time

Figure 8.9 shows the online accuracy for Simple Logistic Regression, built with only Gaze features. The accuracy of classification starts to constantly grow above the baseline after seeing 24% of the data, and after seeing 42% of the data, it stabilizes at over 70%. The model slowly reaches 80% after seeing 85% of the data. The classifier appears to be quite balanced during the session in distinguishing LL and HL. The difference in class prediction accuracy, on average, is not more than 10%. In the end, this difference drops to less than 4%.
The average overall accuracy during the session was 67.77%. In terms of class predictions, average performance over time was 73.15% and 62.38%, for LL and HL, respectively (if we ignore the first 24% of the data, where the classifier is very unbalanced; average accuracies were 69.26%, 76.86%, and 73.06% for LL, HL, and All, respectively). As discussed in Section 7.4, the selected features for this classifier mainly include the features (9 out of 10) that describe attention separate AOIs (e.g., Text Content and Image Content). This explains the lack of improvement in predictions during the first 20% of the MetaTutor session.

### 8.4.2 Performance of Multilayer Perceptron on Actions data over time

Figure 8.10 shows the online accuracy for Multilayer Perceptron trained, using Actions features only. The classification accuracy barely changes until 54% of the data is seen. For the remaining part of the session, this prediction accuracy starts to grow but is very unbalanced until the last few minutes (3% of the session duration, on average) of the MetaTutor session.
An abrupt change occurs in the leading class in the middle of the interaction, which can be explained by the features used for training this classifier. The fact that students had to work with different tools and layouts resulted in a lack of data at the beginning of the session for calculating some features during parts of the session. For example, students did not have access to the content of MetaTutor until they set two initial subgoals and reported prior knowledge on the topics they planned to study, so that the corresponding features (e.g., TimeSpentWithContentOverall) were not calculated. We report the full list of features that were selected by feature selection for this classifier in Table 8.5. Groups of actions listed in the table were defined in Section 5.2. Our model relied heavily on the features describing students’ usage of SRL tools and embedded notepad: eight of ten features describe creating inferences, managing subgoals, evaluating personal ability to accomplish the subgoals, and working with notes. Only three features that were related to subgoals were calculated from the beginning of the session (NumSG, TotalTimeSettingSG, and numPLAN). Five other features were not calculated until students spent a reasonable amount of time studying the topic. These included
numJOL, a feature that shows how students evaluate their learning (e.g., “I don’t understand this material,” and “I feel strongly I understand this material”). On average, the first occurrence of this happened at 54 minutes from the session start (48% of the average session time). Similarly, the number of times students expressed their feeling about learning the content from the current page (e.g., “I feel, I can understand it,” and “I feel I don’t understand this”) (numFOK) and the number of times that students spent making inferences from the content (TimeINF) were tracked, with the first occurrence happening, on average, after 47% and 53% of the average session time, respectively.

Table 8.5: Selected features for Multilayer Perceptron trained on Actions

<table>
<thead>
<tr>
<th>Group of actions</th>
<th>Selected features</th>
</tr>
</thead>
<tbody>
<tr>
<td>General features for working with MetaTutor</td>
<td>TimeSpentWithContentOverall, TimeFullView</td>
</tr>
<tr>
<td>Learning goals management</td>
<td>NumSG, TotalTimeSettingSG, numPLAN</td>
</tr>
<tr>
<td>Working with notes</td>
<td>NoteCheckingDuration, NoteCheckingNum</td>
</tr>
<tr>
<td>Other SRL tools usage</td>
<td>TimeINF, , numJOL, numFOK</td>
</tr>
</tbody>
</table>

Other activities that were not tracked until late in the session included events related to checking notes (NoteCheckingNum and NoteCheckingDuration). These features were calculated late in the session since, on average, students began taking notes only after 60% of the session had passed.

Actual studying of the content on the circulatory system happened only after watching video tutorials and setting subgoals, which was about 25 minutes of interaction (23%). For instance, the feature TimeSpentWithContentOverall showed the proportion of time a student spends with content available in the currently active layout. Another feature that was selected for this model described if and for how long students used the Full View layout with fullscreen content but without access to any SRL tools (TimeFullView). This feature was calculated for the first time after 38% of the session.
Features related to subgoals were calculated from the beginning of the MetaTutor session. These included the number of subgoals set during the session ($\text{NumSG}$), time spent setting subgoals ($\text{TotalTimeSettingSG}$), and number of times students managed their subgoals by setting, changing, or completing them ($\text{numPLAN}$). Initially, all students were asked to set exactly two subgoals; thus, the values of the two features ($\text{NumSG}$ and $\text{numPLAN}$) that showed quantities were the same for all students at the beginning. Students could set additional subgoals or change their current subgoals later after the start of the browsing session. Thus, the values of these features were changed later.

Only one feature of ten ($\text{TotalTimeSettingSG}$) had a meaningful value at the beginning of the session. The absence of sufficient data during the first part of the learning session was interpreted by the model as being a lack of corresponding actions (e.g., no information on reading content during the planning phase of the session). In this case, the classifier assesses the students' learning based on the observed data. A closer look at the features reveals that LL, on average, made fewer inferences and set fewer subgoals than HL. At the beginning of the learning session, the model would tend to label a student as LL because of lack of data to make a proper prediction. The assessment would change rapidly as the system tracks more actions related to features used.

Some changes could be made to lead to a better performance over time for the Action-based models. Since students do not need to use certain SRL tools until they actually start reading the content, sparse data is the result. To deal with this issue, we could divide the learning session into parts, like (1) setting up subgoals, (2) working on the first initial subgoal, and (3) working on the second initial subgoal, and train separate classifiers for each part. This approach could improve performance of the classifier since it would not suffer from a lack of values for certain features at the beginning of the session and would provide assessment for the current activities only.
8.4.3 Performance of Simple Logistic Regression on the Full data over time

Figure 8.11 shows the classification accuracy over time for Simple Logistic Regression trained on Full feature set, which is one of the best performing and balanced models for predicting overall performance based on the complete interaction in the MetaTutor session.

![FULL SLR, 10R10F](image)

**Figure 8.11: Accuracy over time of Simple Logistic Regression on Full**

For this model, four Gaze features and six Actions features were selected (see Table 8.6 for the list of selected features). All selected Gaze features are transitions between and inside several AOIs. Three of these features (transitions to Text Content from Image Content, and to Subgoals from Table of Contents and within Table of Contents) were also selected for the gaze-based classifier. In addition, transitions inside Image Content were selected. The Actions features included four features on usage of the SRL tools in MetaTutor (number of times students evaluated their learning (numJOL), time spent on reporting prior knowledge (TimePKA), average number of SRL processes traced while student worked on the first and the second subgoals (AVGNumSRLpePagetoSG0AnyTime and AVGNumSRLpePagetoSG1AnyTime), number of times student checked notes (NoteCheckingNum), and rate of relevant pages
opened while working on the first initial subgoal \((\text{RatioPagesRelevantToInitialSubgoal1ReadWhenSubgoal1Active})\). These features were not calculated until after the start of the browsing session.

Table 8.6: Selected features for Simple Logistic Regression trained on \textit{Full}

<table>
<thead>
<tr>
<th>Feature set</th>
<th>Group of actions</th>
<th>Selected features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaze</td>
<td>Transitions</td>
<td>Subgoals_proptransfrom_TableOfContents, TableOfContents_proptransfrom_TableOfContents, TextContent_proptransfrom_ImageContent, ImageContent_proptransfrom_ImageContent</td>
</tr>
<tr>
<td>Actions</td>
<td>Working with notes</td>
<td>NoteCheckingNum</td>
</tr>
<tr>
<td></td>
<td>Other SRL tools usage</td>
<td>numJOL, TimePKA, AVGNumSRLpePageToSG0AnyTime, AVGNumSRLpePageToSG1AnyTime</td>
</tr>
<tr>
<td></td>
<td>Working with content</td>
<td>RatioPagesRelevantToInitialSubgoal1ReadWhenSubgoal1Active</td>
</tr>
</tbody>
</table>

The average trend, over time, for Simple Logistic Regression trained on the \textit{Full} feature set is similar to the one trained on the \textit{Gaze} data only (Figure 8.12). Like the \textit{Gaze} classifier, performance improved after 20% of the data becomes available. Such behavior can be explained by the fact that all selected features, except \textit{TimePKA}, are not calculated until the start of the browsing session. \textit{TimePKA} is first calculated after the student sets two learning subgoals. The classifier performance constantly improves with some minor fluctuations as more data becomes available. The classifier, based on the \textit{Gaze} data constantly outperforms the classifier based on the \textit{Full} data.
Figure 8.12: Average performance of simple Logistic Regression over time on Gaze and Actions feature sets

Compared to the Gaze classifier, which has a good balance in predicting low and high learners, the classifier trained on the Full data shows little balance until the last 20% of the interaction. This is similar to Multilayer Perceptron trained on Actions, which is also very unbalanced until the last minutes of the MetaTutor session. In Simple Logistic Regression on Full actions, the balance of the classifier improves after seeing 50% of the MetaTutor session. Like the previous model (Multilayer Perceptron on Actions feature set), the Actions features used in this classifier are calculated late in the session, after seeing at least 50% of the data. The features: numJOL, AVGNumSRLpePagetoSG0AnyTime, and AVGNumSRLpePagetoSG1AnyTime were calculated later. Good balance is achieved in distinguishing LL and HL after 80% of the data is seen. Overall, this classifier is better balanced than the classifier based on Actions.

These results suggest that using the Actions features and certain gaze features in the form in which they are calculated in this thesis, brings imbalance to the predictions of our classifiers since some behavioral data is not available throughout the whole session.
8.4.4 Performance of Multilayer Perceptron on Full data over time

We simulated the performance for Multilayer Perceptron on Full feature set (Figure 8.13), over time, since it is one of the best performing classifiers in the initial set of models. Like the other classifiers, the prediction accuracy begins to constantly grow above the baseline after seeing 27% of the data. The 70% accuracy is achieved after seeing 40% of the data. Nevertheless, this classifier is very unbalanced until the last 20% of the interaction.

![Figure 8.13: Online accuracy of Multilayer Perceptron on Full feature set](image)

This classifier used a set of features that represents various sets of processes. It was trained with five Gaze features and five Action features (Table 8.7). The Gaze features included both transitions between AOIs and information on attention to specific AOIs. Nevertheless, only one feature (Subgoals_proportionnum_dynamic) was calculated from the beginning of the session. The Action features cover reading and note-taking activities as well as usage of various SRL processes. Similarly to the previously
discussed classifiers, all five Action features are calculated at later phases of the learning session.

In contrary, the data on student’s attention to the interface elements is collected earlier in the session, and hence the Gaze classifier achieves a good balance faster. These findings suggest that gaze data alone can work better as a source for predicting learning with an open-ended environment where student’s actions are unstructured and sparse.

<table>
<thead>
<tr>
<th>Feature set</th>
<th>Group of features</th>
<th>Selected features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaze</td>
<td>Transitions</td>
<td>OverallLearningGoal_proptransfrom_TableOfContents</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Subgoals_proptransfrom_TableOfContents</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ImageContent_proptransfrom_ImageContent</td>
</tr>
<tr>
<td></td>
<td>Attention to AOIs</td>
<td>Subgoals_proportionnum_dynamic,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TableOfContents_fixationrate</td>
</tr>
<tr>
<td>Actions</td>
<td>Working with content</td>
<td>RatioPagesRelevantToInitialSubgoal1ReadWhenSubgoal1Active</td>
</tr>
<tr>
<td></td>
<td>Working with notes</td>
<td>NoteCheckingDuration</td>
</tr>
<tr>
<td></td>
<td>SRL tools usage</td>
<td>TimePKA, numCE, AVGNumSRLpePagetoSG0AnyTime</td>
</tr>
</tbody>
</table>

### 8.4.5 Performance of Ensemble classifier over time

Figure 8.14 shows performance over time for the best ensemble described earlier in the chapter. This ensemble combines the following classifiers: Simple Logistic Regression on Full and Gaze feature sets and Multilayer Perceptron trained on Full feature set. This classifier achieved an 86.40% overall accuracy (with full data available for student evaluation), which is a better performance than the best performing base classifiers. Nevertheless, the average performance over time for this ensemble was 67.75%, which is not significantly different from Simple Logistic Regression trained on Gaze data (t(99)=0.51, p = 0.96). Like the classifiers discussed earlier, the performance of the
For a deeper evaluation of the ensemble’s performance over time, we looked at the online trends for predicting each class and for predicting both of them independently. Figure 8.15 shows three separate graphs with performance trends over time that predict both classes (Figure 8.15 - a), low learners (Figure 8.15 - b), and high learners (Figure 8.15 - c). Each graph shows four trends: one for the ensemble classifier and the three base classifiers combined for this ensemble. The trends for both classes have accuracies (Figure 8.15 - a) that are similar to the base classifiers: the performance begins to constantly grow after seeing 20% of the data. The ensemble's trends for LL and HL performance over time (Figure 8.15 - b and Figure 8.15 - c, respectively) are very similar to those of Simple Logistic Regression trained on Full features. Nevertheless, the ensemble begins to outperform it in predicting HL after seeing 25% of
the data. This may be explained by the fact that another base classifier, Multilayer Perceptron on *Full* feature set is continually good at detecting high learners, and thus, might be contributing much to the performance of the ensemble for this class. Predicting LL is different: the trends of the Ensemble and Simple Logistic Regression are very similar until 73% of data is observed, when the ensemble shows a slightly better performance over this class.

The Ensemble does not work well in the online setting due to the high imbalance in distinguishing the assessing of learning for two of the three base classifiers (see Figure 8.15 - b and Figure 8.15 - c for trends in predicting LL and HL). Surprisingly, the third model (Simple Logistic Regression of Gaze feature set) with a better balance (shown in Figure 8.9) does not improve the results. The Ensemble achieves a high and stable accuracy for predicting HL but fails to predict LL in half of the session (the classifier achieves 60% accuracy after seeing 50% of the data).

Moreover, the model has the same drawbacks as in the Simple Logistic Regression on Full feature set and Multilayer Perceptron on Full feature set and is very unbalanced. Nevertheless, it catches up in balance slightly faster than the other models: after seeing 82% of the data, the overall accuracy stabilizes over 80% and the difference between the two classes does not exceed 10%. The average accuracy over time is 67.75%, with a 62.13% accuracy in predicting Low Learners and a 73.38% accuracy in predicting High Learners.
Figure 8.15: Ensemble model with the base classifiers
8.5 Discussion of results

In this chapter we discussed the performance of a set of classifiers trained with three feature sets (Actions, Gaze, and Full) overall and over time.

The models performed better than the baseline (chance, 50%), and the results showed that Gaze-based classifiers on average performed better than Actions-based classifiers, when the full interaction data is available. We conclude that the low performance of the Actions-based classifiers is caused by the fact that the main learning in MetaTutor that occurs through reading that cannot be tracked by parsing the explicit actions on their own. Classifiers trained on the Full feature set, which is a combination of Actions and Gaze data, achieve a higher average performance, than those trained on the Gaze data. Nevertheless, the best performing classifier on the Gaze data only showed an accuracy that was comparable to the best performing classifier on the Full feature set. Thus, the Gaze only data can be as predictive as if it was combined with Actions (~81% overall).

Following the approach of [9], we trained several ensemble classifiers and used several methods for grouping the base classifiers. The only approach that showed a significant improvement over the base model classifiers was grouping the three best performing models (Simple Logistic Regression on Full and Gaze feature sets and Multilayer Perceptron on Full feature set). This approach achieved an overall accuracy of 86.40%, compared to 81.20% that was achieved by the best performing base classifier.

We simulated online learning for the best performing classifiers to explore their value in predicting learning when only partial data from interaction is available. The performance of all classifiers begins growing constantly above the baseline after seeing 20%-30% of the data, and the performance stabilizes above 70% after seeing 37%-45% of the data. An accuracy of 80% is achieved by the last 15% of the interaction time. This can be explained by the learning session, where students do not start reading the content until after they set two subgoals to work on and report prior knowledge on the subgoals (on average, this took 25 minutes (or 23%) of the average session time). We found that only
Simple Logistic Regression on the *Gaze* dataset showed a stable and balanced performance during the session. Other classifiers that were trained on *Actions or Full* showed very poor balance in distinguishing LL and HL, until the last minutes of interaction. The low performance of classifiers over time and the lack of balance can be explained by the nature of interactions with MetaTutor. First, the main activity for MetaTutor students is reading, which cannot be tracked by explicit interface actions. Second, MetaTutor is used by a variety of students, who may not need to use all of the tools available in MetaTutor; they would be free to choose the tools they want and have time to do this. Consequently, some of the features that describe usage of SRL tools or reading behaviors are not calculated until a specific point in the session is reached. This affects the performance of classifiers that were trained on the *Actions* features. For example, the Multilayer Perceptron on *Actions* features fails to assess learning of students until they are 50% through their session. Our results show that *Gaze* data alone can do better than if used in addition to *Actions* when inferring learning with MetaTutor.

In addition, we looked at the features that were selected for the winning classifiers. We found that gaze-based models rely greatly on the transitions between AOIs, and the Action-based models use many features that describe the usage of SRL tools and working with content. For the best classifiers, based on the full feature set, selected features were distributed equally between *Gaze* and *Actions*. The selected *Gaze* features included mainly *AOI-based* features. The most popular AOIs were content areas, subgoals, and Table of Contents. For the Simple Logistic Regression trained on the *Full* features, three of four features were also present in the feature set for Simple Logistic Regression trained on *Gaze*. The *Actions* features included information about the number of times and time spent on checking notes and reporting prior knowledge; the average number of SRL processes per each of the two initial subgoals and the ratio of pages relevant to initial subgoals. This supports the conclusion that information on the usage of SRL tools can contribute to estimating the learning performance of students.
Chapter 9  Conclusions and Future Work

In this thesis, an approach is presented for training classifiers to assess student learning performance at the end (overall accuracy) and during (over time) a learning session with MetaTutor, a hypermedia learning environment with a set of tools for monitoring SRL processes. We used data from 66 university students, collected during a study that was conducted in 2012 by a group of collaborators from McGill University. We did not participate at any stage of the study design, preparation, or running of experiments, but had full access to the collected eye-tracking and interaction data.

We described the full process of training models from the pre-processing of raw data to inferring labels for low learners and high learners. Our approach is based on machine learning techniques. The methodology of preparing eye-tracking data and training classifiers is not specific to the environment and can be applied to any other ITS. We formed three separate feature sets based on the available eye-tracking data (Gaze), action logs (Actions), and their combination (Full). We trained a set of classifiers using commonly used machine learning algorithms (Simple Logistic Regression, Multinomial Logistic Regression, Naive Bayes, Random Forest, and Multilayer Perceptron) and ensemble algorithms.

9.1  Thesis goals satisfaction

The research questions, stated in the Introduction, focus on the value of eye-tracking data. We aimed to see if eye-tracking is a useful source to predict learning at the end and during learning sessions. We compared Gaze classifiers with classifiers that are based on features from explicit interactions with MetaTutor since they can easily serve as a source of data.
9.1.1 Research question 1: Can eye-tracking data be used to assess learning performance of a student interacting with MetaTutor?

This is the main question of our research. Previous findings have already confirmed that eye-tracking can contribute to predicting learning as an independent source of data [4] and in combination with actions [9] for other types of ITSs.

We designed two sub-sets of eye-tracking features that represent interface-independent (no-AOI) and interface-specific (AOI-based) measures. The former set of features includes high level measures based on simple statistics of fixations and saccades (e.g., mean and standard deviation of fixations duration). The latter set is based on statistics of fixations within each of seven pre-defined AOIs and the transitions between them. Our machine learning experiments showed that classifiers trained with AOI-based features (best overall accuracy over five models = 70.4%) achieve significantly better results than those based only on no-AOI features (best overall accuracy over five models = 54.4%). Nevertheless, the best results were achieved when using a combination of these two feature sets: the best performing gaze-based classifier (Simple Logistic Regression) achieved 81.2% accuracy overall. It was also very balanced in terms of distinguishing High Learners and Low Learners (83.2% and 79.2%, respectively).

These findings confirm that eye-tracking data is a promising source in assessing student learning performance overall, when the full interaction data from learning sessions with MetaTutor is available.

Inspired by these results, we simulated learning over time to evaluate whether or not gaze-based classifiers are capable of detecting LL and HL during interactions with MetaTutor to provide students with personalized feedback or hints.

Online learning for the Simple Logistic Regression on Gaze data was simulated by feeding chunks of data incrementally to the trained model, corresponding to two minutes of interaction with MetaTutor to the classifier. The classifier showed constant improvement over the baseline (chance, 50%) after seeing only 23% of the data. An
accuracy of 70% was reached after seeing 42% of the data. The classifier was also very balanced during the session.

**9.1.2 Research question 2: How well does eye-tracking perform in predicting student learning?**

To confirm the value of classifiers based on eye-tracking, we compared their performance with classifiers trained with features calculated from explicit actions within the interface. On average, gaze-based classifiers (average accuracy = 72.6%) showed higher performance overall than did actions-based classifiers (average accuracy: 66.92%).

The best performing Actions classifier (Multilayer Perceptron, 73.8%) showed a significantly lower performance, compared to the best performing gaze-based model (Simple Logistic Regression, 81.2%). Like the Simple Logistic Regression on Gaze data, we simulated online learning for Multilayer Perceptron trained on Actions to see how quickly it would improve above the baseline, in assessing learning over time. The trend over time was very unbalanced during learning sessions. In contrast, the Simple Logistic Regression based on Gaze quickly achieved a significant improvement over the baseline.

We trained a set of classifiers based on the full feature set that combined Gaze and Actions features. The average performance of this set of classifiers (77.56%) was significantly better than the average performance of the Gaze classifiers (72.6%). The classifiers on the Full feature set showed less variance in overall performance across different algorithms. Nevertheless, the two best classifiers in the Full and Gaze feature set (both with Simple Logistic Regression as the learning algorithm) achieved comparable overall accuracies (81.2% and 81.4%, for Gaze and Full, respectively). We conclude that, with no prior knowledge about which learning algorithm to use, using the Full feature set for training classifiers is safer as it shows higher accuracy on average.

The simulated online learning for Simple Logistic Regression trained on the Full feature set showed that the average performance over time (63.82%) was lower than the
performance of Simple Logistic Regression on Gaze (67.77%). Although the average trends were similar, the classifier trained on the Full data was very unbalanced until the last 20% of the MetaTutor session.

In addition to combining eye-tracking and actions features to a joint feature set, we combined different base classifiers with a simple majority voting technique (ensemble model). Following the approach of [9], we created five ensemble models that combined three base models, differing by the learning scheme, but we did not find any significant improvement in performance over classifiers trained on the Full set of features. The combination of the best three base models (Simple Logistic Regression trained on Gaze and Full feature sets and Multilayer Perceptron trained on Full feature set) achieved an overall accuracy of 86.4% and showed good balance when assessing learning at the end of the MetaTutor session but lack of balance over time.

We showed that the Gaze classifier can perform as well as Full data-based models overall. Moreover, the Gaze classifier showed the best balance and the fast improvement in performance over time when simulating online learning. These results suggest that Gaze data alone can do better than if used in addition to Actions when assessing student learning with an open-ended learning environment.

9.1.3 Research question 3: Which elements of the interface contribute most to assessing learning?

We analyzed features selected for the best performing classifiers. The selected features for Simple Logistic Regression on Gaze data included seven transitions between AOIs, two rates of fixations in Text and Image content, and one interface-independent feature. Five features were related to AOIs that represent SRL tools, suggesting that attention to SRL tools is indeed important for assessing learning.

This is supported by the results of feature selection for Multilayer Perceptron on Actions and Gaze and Simple Logistic Regression on the Full feature set. For each of these, a set of interaction-based features on reporting prior knowledge, taking notes, judging personal understanding, and working with subgoals was selected. We also found that
using *Actions* features in our classifiers resulted in a lack of balance when simulating online learning sessions.

The feature selection for *Full* classifiers resulted in roughly equal numbers of *Gaze* and *Actions* features selected.

This can be explained by the nature of interactions with MetaTutor. Some features (e.g., those related to taking and checking notes, and reading relevant content) that were used for building models are not available immediately at the start of the session, because students do not need to take corresponding actions until later in the session. Until students start using these tools, the classifier lacks data to produce meaningful predictions and shows poor performance.

### 9.2 Limitations and future work

The approach described in this thesis could be improved in several ways. First, we found that *Actions* features do not work well in assessing learning during the MetaTutor session. The overall average performance of classifiers based on *Actions* was lower than classifiers trained on *Gaze* or *Full* features. In the case of online simulations, all of the classifiers that include features based on interaction with MetaTutor (trained on *Actions* and *Full* feature sets) showed a very unbalanced performance over time, until the last few minutes of the learning session. As suggested before, this is likely because students rarely perform certain actions (e.g., actions related to note-taking) until they spend sufficient time working with MetaTutor. To address this problem, we could divide the interaction into several stages corresponding to the different activities in MetaTutor (e.g., four stages: (1) setting-up subgoals, (2) working on the first initial subgoal, (3) working on the second initial subgoal, and (4) working on additional subgoals) and training separate classifiers for each stage. This strategy could improve the predictions, because the new classifiers would be based on actions that students usually perform at
each stage. For example, in the first stage, only features relevant to setting-up subgoals would be used to train the classifiers.

Second, we could parse all inputs typed by students (e.g., notes, summaries, and reports on prior knowledge) and design new features based on these.

Third, we could calculate more gaze-features that describe more reading behaviors (e.g., fixations per word).

In future work, we would want to adopt these improvements with additional machine learning techniques (e.g., ensemble classifier with a more advanced combination method) to improve the accuracy of assessing learners. Once the predictions of low and high learners are reliable, we are planning to use the updated student model to improve the underlying AI mechanism in pedagogical agents. Since no correlations have been found between agent prompts and learning performance (adaptive and non-adaptive conditions), agents have reinforced the use of certain SRLs but have not affected the efficiency of learning. As a possible explanation, the recommendations may be provided at the wrong time or students may not understand how to benefit from them. An updated model could be more efficient in assessing students’ actual cognitive and meta-cognitive states and provide personalized prompts and feedback to scaffold self-regulated learning and help students master new material and skills efficiently.
Bibliography


52. Affectiva, Q.: Sensor.


