ADAPTIVE SERIOUS EDUCATIONAL GAMES USING MACHINE LEARNING

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN THE FACULTY OF SCIENCE AND ENGINEERING

2018

By
Harits Ar Rosyid
School of Computer Science
Contents

Abstract 12

Declaration 13

Copyright 14

Acknowledgements 15

1 Introduction 16
   1.1 Serious Educational Games Overview .......................... 16
   1.2 Necessity of Personalising Serious Educational Games ...... 18
   1.3 Existing Strategies to Develop an Adaptive SEG ............ 20
   1.4 Our Approach to Developing an Adaptive SEG ............... 22
   1.5 Aims and Objectives ........................................... 23
   1.6 Thesis Structure ............................................... 24

2 Background 25
   2.1 Games for Learning Purposes .................................. 25
   2.2 Recent Techniques for Developing Serious Games .......... 25
       2.2.1 The Two Main Components in Serious Games .......... 28
   2.3 Assessment Techniques for an Adaptive SEG ................. 30
       2.3.1 What is Learning and Enjoyment? ...................... 30
       2.3.2 Assessing Learning in Serious Games ................. 32
       2.3.3 Assessing Enjoyment in Serious Games ............... 33
   2.4 Adaptation Techniques ......................................... 34
       2.4.1 Player-centric Adaptation in SEG .................... 35
       2.4.2 Spaced Repetition Learning ......................... 37
   2.5 Rule-based Systems ............................................. 38
# List of Tables

2.1 Confusion Matrix for a Binary Class Problem. .......................... 64

3.1 The conditions to categorise game content based on difficulty level. . 82
3.2 Z-test on Proportion of Reported FUN. ................................. 91
3.3 Z-test on Proportion of Learning Improvement. ......................... 92
3.4 Reported experiences: Learning Performance vs. Enjoyment. ........ 93

4.1 The Two-dimensional Game Playing Routines of the Participants. . . 107
4.2 Survey Participants and Samples by Sessions. ........................... 108
4.3 Features with Importance Rank for Predicting Enjoyment. ............. 117
4.4 Learning Classes. ............................................................ 121
4.5 Features with Importance Rank for Predicting Learning Performance. 125

5.1 Simulated Adaptation Experiments ....................................... 153
5.2 Missions Totals of GMO Performances (each experiment runs in 100 iterations) ......................................................... 163
List of Figures

2.1 Rule-based system architecture (reproduced from various sources) . . . 39
3.1 SEG development framework (reproduced from [1]) . . . . . . . . . . 71
3.2 Exemplary game content of different difficulty levels (left-right): Easy,
    Medium, Hard (reproduced from [1]). . . . . . . . . . . . . . . . . . 83
3.3 Mapping result regarding total games in a cluster (reproduced from [1]). 85
3.4 Mapping result regarding the sum of standard deviation of game content
    properties in a cluster (reproduced from [1]). . . . . . . . . . . . . . 86
3.5 SEG game session’s procedures (reproduced from [1]). . . . . . . . . . 87
3.6 Chem Dungeon’s game stage layout (reproduced from [1]). . . . . . . 88
3.7 User survey’s procedures (reproduced from [1]). . . . . . . . . . . . . 90
4.1 Relational Database for Serious Educational Game Survey . . . . . . 106
4.2 Enjoyment Classification Measures . . . . . . . . . . . . . . . . . . 115
4.3 Enjoyment Classification using Reduced sets of Features . . . . . . . 120
4.4 Distribution of Samples for LP Training set . . . . . . . . . . . . . . . 123
4.5 Optimal Performance of Random Forest Predicting LP classes . . . 124
4.6 Learning Performance Classification using Reduced sets of Features . 127
5.1 Adaptive SEG Diagram . . . . . . . . . . . . . . . . . . . . . . . . . . 132
5.2 The Game Structure of SEG . . . . . . . . . . . . . . . . . . . . . . 134
5.3 Experience-driven Game Content (EdC) Baseline Search . . . . . . . 148
5.4 Clustering Analysis in EdC . . . . . . . . . . . . . . . . . . . . . . . 150
5.5 Hierarchy of Simulation Experiments . . . . . . . . . . . . . . . . . . 152
5.6 Cumulative Rewards per mission without GMO (Random Assignment)
    (i.e. a Mission = six game stages) . . . . . . . . . . . . . . . . . . . 156
5.7 Cumulative Rewards per mission using GMO without Machine Learning,
    Random Sequence of Mission (i.e. a Mission = 6 game stages) . . 158
5.8 Cumulative Rewards per mission using GMO without Machine Learn-
ing, Raw Sequence of Mission (i.e. a Mission = 6 game stages) . . . . 158
5.9 Mission Simulation - RL GMO (PGF and SbS-RL), exploration rate: 0.1159
5.10 Mission Simulation - RL GMO (PGF and SbS-RL), exploration rate: 0.3159
5.11 Mission Simulation - RL GMO (PGF and MbM-RL), exploration rate:
   0.1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160
5.12 Mission Simulation - RL GMO (PGF and MbM-RL), exploration rate:
   0.3 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 161
5.13 Mission Simulation - RL GMO (MPF and SbS-RL), exploration rate:
   0.1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 162
5.14 Mission Simulation - RL GMO (MPF and SbS-RL), exploration rate:
   0.3 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 162
5.15 Mission Simulation - RL GMO (MPF and MbM-RL), exploration rate:
   0.1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164
5.16 Mission Simulation - RL GMO (MPF and MbM-RL), exploration rate:
   0.3 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164
5.17 Comparison of 'TotalMission' values . . . . . . . . . . . . . . . . . . . . . . 165
C.1 Procedures inside GMO . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 199
C.2 Priority Group Fitting strategy . . . . . . . . . . . . . . . . . . . . . . . . 200
C.3 Most Populated group Fitting strategy . . . . . . . . . . . . . . . . . . . . . . 201
C.4 Stage-by-Stage Sequencing Agent . . . . . . . . . . . . . . . . . . . . . . . . 202
C.5 Mission-by-Mission Sequencing Agent . . . . . . . . . . . . . . . . . . . . . . 203
Abbreviations

AEE  Adaptive Experience Engine
AFC  Alternative Forced Choice
AGNES  Agglomerative Clustering Analysis
ANN  Artificial Neural Network
AUC  The measure of the area under Receiver Operating Characteristics (ROC) curve
BIRCH  Balanced Iterative Reducing and Clustering using Hierarchies
CbKST  Competence-based Knowledge Space Theory
CF  Clustering Feature
CFTREE  Clustering Feature Tree
CV  Cross validation method
DBN  Dynamic Bayesian Network
DIANA  Divisive Clustering Analysis
DP  Bellman’s Dynamic Programming
EdC  Experience-driven Content
EID  Education material numeric index
EP  Experience Prediction
FN  False Negative
FP  False Positive
FPR  False Positive Rate
GC  Game Content
GMO  Game Mission Optimiser
ID3  Decision Tree algorithm
LbPCG  Learning-based Procedural Content Generation

LP  Learning Performance

LP0  Category of learning performance: *not learning* due to no prior knowledge

LP1  Category of learning performance: *recognition* of a new knowledge

LP2  Category of learning performance: *recalled* a prior knowledge

MbM  Mission-by-mission reinforcement learning target

MCQ  Multiple Choice Question

MLP  Multi Layer Perceptron

MPF  Most Populated group Fitting strategy

NFQ  Neuro-fitted Q-Learning module in Python

NN  Neural Network

NPC  Non Player Character

OOB  The out of bag test samples

PCG  Procedural Content Generation

PFNN  Phase-functioned Neural Network

PFS  Perceptron Feature Selection

PGF  Priority Group Fitting strategy

PostGE  Examination after a game session

PreGE  Examination before a game session

PT  Periodic Table

RF  Random Forest algorithm

RL  Reinforcement Learning method

ROC  Receiver Operating Characteristics
**RPG**  Role-playing Game
**RTS**  Real-time Strategy game
**SbS**  Stage-by-stage reinforcement learning target

**S.C.R.U.B.**  A serious game for learning about bacteria removal

**SEG**  Serious Education Game

**SFFS**  Sequential Floating Forward Selection
**SFS**  Sequential Forward Selection

**SG**  Serious Game

**SLP**  Single Layer Perceptron

**SMOTE**  Synthetic Minority Oversampling Technique

**SRL**  Spaced Repetition Learning

**STS**  Stratified Sampling method

**TN**  True Negative

**TP**  True Positive

**TPR**  True Positive Rate
Abstract

ADAPTIVE SERIOUS EDUCATIONAL GAMES USING MACHINE LEARNING
Harits Ar Rosyid
A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy, 2018

The ultimate goals of adaptive serious educational games (adaptive SEG) are to promote effective learning and maximising enjoyment for players. Firstly, we develop the SEG by combining knowledge space (learning materials) and game content space to be used to convey learning materials. We propose a novel approach that serves toward minimising experts’ involvement in mapping learning materials to game content space. We categorise both content spaces using known procedures and apply BIRCH clustering algorithm to categorise the similarity of the game content. Then, we map both content spaces based on the statistical properties and/or by the knowledge learning handout. Secondly, we construct a predictive model by learning data sets constructed through a survey on public testers who labelled their in-game data with their reported experiences. A Random Forest algorithm non-intrusively predicts experiences via the game data. Lastly, it is not feasible to manually select or adapt the content from both spaces because of the immense amount of options available. Therefore, we apply reinforcement learning technique to generate a series of learning goals that promote an efficient learning for the player. Subsequently, a combination of conditional branching and agglomerative hierarchical clustering select the most appropriate game content for each selected education material. For a proof-of-concept, we apply the proposed approach to producing the SEG, named Chem Dungeon, as a case study to demonstrate the effectiveness of our proposed methods.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on presentation of Theses.
Acknowledgements

All praises and thanks to Allah the Greatest who make this precious journey worthwhile. An uncountable gratitude to my supervisor Dr Ke Chen whose helps, supports, constructive criticism and patience were extraordinary in guiding my PhD journey. A very deep thanks also go to my parents, wife and children, your supports and prayers are limitless. Many appreciation to my big family, participants and friends for their supports. Also, great appreciations to examiners: Professor Minhua Eunice Ma and Dr Xiao-Jun Zeng whose thorough evaluations and guidance have led me to the doctoral award I am pursuing.

This PhD research was fully funded by the Directorate General of Resources for Science, Technology and Higher Education, Indonesia (DIKTI).
Chapter 1

Introduction

In this thesis, we introduce the novel development method Adaptive Serious Educational Game applying Machine Learning techniques. Section 1.1 provides an overview of educational games, Section 1.2 describes why an adaptation is important in serious educational games, Section 1.3 affirms existing strategies, Section 1.4 briefly explains our approach, Section 1.5 examines our research aims and objectives, Section 1.6 describes the structure of the thesis and Section 1.7 provides some definitions of terms used in this thesis.

1.1 Serious Educational Games Overview

Serious educational games (SEG) are games that focus on the learning experience rather than entertainment [2]. However, from our experience, the fun factor is involved in SEG game sessions. In particular, besides the knowledge presented in the game, there are gaming elements embedded to make the player’s learning enjoyable. For instance: game challenges, missions, game mechanics, etc. So, we argue that it is not solely the player's learning involved in an educational game session, but also other affective experiences, particularly enjoyment.

Knowledge representation in a serious game (SG) is not necessarily crude but is often encased or enhanced in the game environment. A very simple example is as follows: The Pac-Man game is known as an entertainment game where a player navigates a game character (avatar) to eat dots and avoid enemies. The education version of Pac-Man, known as Number Munchers (popular in the 1980s and 1990s) [3], replaces dots with equations. The player’s mission in that game is to collect equations that produce a particular answer. In such a way, players (students) can learn maths equations with
an element of fun involved (e.g., challenges created by avoiding an enemy). Number Munchers is therefore an example of an educational game with an element of challenge and learning that makes it motivating and amusing [4]. For instance, by playing SGs, players can repeat a game level (with inherent learning tasks) he failed without the fear of losing marks. The following paragraphs discuss the advantages of serious games from various perspectives.

A serious game offers unlimited replay-ability at a minimum risk of a real damage, in terms of a calculated result, or exam failure. One SG for training levee patrollers in Netherlands [5] is an example; it simulates cases that may happen to a levee (artificial dike). The game offers a tutorial scenario where players gain a basic knowledge about calamity symptoms (e.g., a leak in a part of the dyke) and appropriate responses. Failures are welcomed because there is no real damage caused by this SG; however, it simulates damage and generates appropriate hints or feedback that influences players to learn from it. Levee patrollers can train themselves using this game repetitively until they become familiar with the symptoms and solutions.

A serious game is also intended to minimise expenses while maintaining the actual scenario for the serious purpose. This is seen from the example of a serious game that has been developed for simulating medical treatment in a combat situation [6]. It familiarises trainees with the common medical-related cases that often emerge in combat and provides the necessary and appropriate care. Such simulated environments reduce the expense of a real-time or a lab-based simulation where specific medical equipment and human patients must be available. Again, the trainees do not need to avoid failure because no real damage or losses occur in the SG scenario. In addition, a repetitive scenario is welcomed as it reinforces the trainee’s acknowledgement of the situations and treatments.

Game elements in an educational game entail some of the critical factors to engage and motivate students when learning embedded knowledge. User studies using the Crystal Island serious game, a game to introduce microbiology to eighth-grade students, proved such expectations [7]. The statistics of the studies emphasised that the use of a game-based environment has significantly improved students’ learning progress (i.e. derived from the difference between pre- and post-tests) [8]. It also revealed that the three-dimensional game environment made the students feel more engaged and motivated to accomplish the mission. Especially for some students who cared about their in-game performances, as this typical game-based learning encouraged them to solve the inherent mystery as fast as possible [8].
So, as an alternative and/or a supplementary learning method, a SEG or game-based learning offers many advantages over classic learning strategies such as reading books or tutorials. However, developing the SEG appears to be a complex task. Because one must consider various factors such as ensuring the game is appropriately representing the knowledge and is suitable for the target players. Therefore, SG development frameworks were proposed to ensure game products follow the criteria for a game that is built for learning, such as [9, 10, 11, 12]. Guidelines within these frameworks are theoretically correct given that an SEG contains complex content, such as learning materials, learning methods, computer graphics, game mechanics, etc. Following one of these frameworks should thus produce an appropriate SEG. However, such frameworks provide high-level procedures that are sometimes lacking technical-level implementations. As a result, developers are left alone determining the actions that follow the framework, such as via collaboration with experts from the relevant domain [13]. Nevertheless, most of them expose various human resources that at some point have made the development runtime expensive and time-consuming. So far, only one method claimed it minimises such high requirements yet produces a quality game-based learning, i.e. [12]. In addition, infeasibility becomes the issue when the SG’s content spaces are immense. Therefore, the first challenge for our research is creating a framework that allows the development of SEG more automatically. We do not intend to replace the expert’s knowledge with a computer program, but we would like to maximise the use of available information of the content to be the basis for the SEG. To some extent, we want to minimise the involvement of experts in the development process and make this applicable for a broader range of developers.

1.2 Necessity of Personalising Serious Educational Games

Using a modern search engine, one can easily research educational games (for example, using the keyword phrase ‘education games’ produces approximately 216,000,000 results in 0.6 seconds). This number shows the vast selection of educational games and topics available. By a number of random selections for the resulting educational games, we found that most of them have non-adaptive contents. In this case, to progress the game, a player must complete the game at a level he/she feels confident to do, repeating the same level if he/she fails. To accomplish that, a player should force him/herself to persevere although boredom or anxiety is growing. Unfortunately, not all players can motivate themselves appropriately in such a case.
1.2. Necessity of Personalising Serious Educational Games

The rapid development of game engines that offer many features and easiness of use allows the development of SGs with more capabilities for complementing education. Some researchers propose methods for developing sophisticated SGs that present dynamic contents to maintain the positive experiences of the player. For example, the SG Travel in Europe presents a specific sequence of learning materials for a particular player [14]. Likewise, the 80 Days Around the World game generates dynamic narratives based on a player's in-game actions [15]. Elektra serious game generates dynamic interventions driven by the estimated player’s motivation [16]. These examples of personalised SGs (commonly known as adaptive serious games) bring optimised experiences regarding players’ characteristics. Specifically, different players have various abilities and proficiency, which should be taken into account to drive the adaptation of the SG [2, 17, 18]. Such differences are identified via a method called the 'assessment of target experiences' [18, 19].

There are various experiences of the player that drive the personalization. The player's learning was reported to receive the most attentions from many scholars in the field of adaptive SGs, such as Travel in Europe and Sea Game by Bellotti [14], Math games by Ismailovic [19], Crystal Island narrative-based SG [7, 20] and much more. Affective experiences were also reported to drive the adaptation of SGs, such as, Emotion [21], Motivation [15, 16, 22, 23], Enjoyment [14], and Engagement [24]. An affective experience can serve the same importance as the learning outcome. However, in this research we are focusing on Learning and Enjoyment because those are the key experiences in playing an SG [2, 17, 18].

An assessment’s result informs a player’s needs in order to optimise the future outcome [18]. The following sentences describe the relationship between an assessment result with the adaptation. Assume in a game stage the SEG contains high-level difficulty content, but it impedes the player to accomplish the main mission (i.e. a simplification of an assessment method via the game stage accomplishment). In this case, we may translate the situation that the player needs an easier challenge in relation to the content. In other words, the player needs the content to have a lower level of difficulty to pave an easier way for the player to complete the mission. Yet, such a simplified assessment is lacking reliability in that it does not produce an accurate result. An assessment based on one, two or only a small number of inputs (e.g., final result, total failures or duration) is sometimes ambiguous as a category of the experience target. For instance, a victory in a serious game session may entail a different meaning if the
player has learned the knowledge with dedication, through a lucky guess or just playing around. To solve such a problem, we can force the commonly known assessment methods into the SEG, such as: questionnaire or pre- and post-game examinations. On one hand, a reliable and effective assessment method for a user’s learning is through an examination. The difference between pre- and post-treatment examinations quantitatively measures the learning gain [25]. This assessment method has been effectively applied since 1965 [26]. On the other hand, a questionnaire is the popular method to self-report an affective experience, such as enjoyment [27]. However, these assessment methods are intrusive if they are applied in a serious game. This is because the players have to encounter a significant change of conditions from the exciting fun of gaming to a seriously thoughtful exam or questionnaire. Thus, the effect of such an enforcement is a disengagement of the player from the game. Meanwhile, abundant in-game actions named play-log exist and there is a corresponding game environment that can be the potential source for an assessment [18]. Therefore, we are challenged to identify these inputs and their threshold values that represent an experience of the player. Once these input variables are generalised, it should confidently assess the player’s experience automatically without interfering gaming immersion.

1.3 Existing Strategies to Develop an Adaptive SEG

Adaptation is principally a dynamic process that resulting changes of one or more interacting entities to maintain the desired condition. Adaptive games require feedback signals to maintain the expected state. For instance, a player naturally adapts him/herself to the game challenges or difficulties. In this case, the feedback comes from the player’s gaming achievements (real) or experiences (abstract). Through the repetitive play of a game level, his/her gaming achievements increase up to a certain point. So, the adaptation in this regards is the player’s gaming ability. In a dissimilar sense, the adaptation in our research is targetting the game content personalisation that is driven by the player’s gaming experiences such as learning, enjoyment, motivation, etc. Basically, the desired outcome in this adaptive system is a positive experience, the feedback signal is represented by variables of the game session and the game content generator sits between the two entities.

Theoretically, Ismailovic et al. define an adaptation as the intervention in a game session by changing the serious game’s content based on the player’s need [19]. An intervention has many forms and source. For instance, a teacher who accompanies
the player could provide a guidance [19]. This guidance is dynamic according to the player’s needs and conditions. A more advanced intervention happens in the game engine where the content generation follows the player’s gaming preference [28, 29]. Our research is inline with such an adaptation description. The challenge to accomplish this is to develop a game engine capable of generating game content automatically and dynamically. By content adaptation, it is expected that the player achievements can improve or becoming optimal in the following game session.

With respect to that definition, there are three types of adaptation methods: manual, self-reported and automatic [30]. The first method is the most known technique in any genre of game where the player manually configures the content suitable for him/her. The second is somewhat new in the SG research field, as it forces the player to report his experience in a given questionnaire. We argue that both adaptation methods potentially interrupt the players’ enthusiasm for gaming. This could be the result of a disengagement of the player from the SEG during a prolonged application. Therefore, an automatic method is preferable whereby the content personalisation is automatically performed without interrupting the player. One of the highlighted methods is explained in the following paragraphs.

Ismailovic mentioned in his article that prior to the application of an adaptive serious game to target players, an exploratory study was carried out via interviews with relevant experts and the target players [19]. They, then collaborated together to develop a serious game. Using the produced serious game, a further user study was held to model the relationships between the players, game content and the treatment given. In the study, a tutor observed a player and the game content during a game session. In this case, the game content consisted of both controllable and static game elements. Based on the tutor’s knowledge and observation, he/she manipulated the game elements to meet the player’s needs. However, it was lacking descriptions regarding the extent to which the game elements should be built and organised to fit the adaptivity. Moreover, the model of the player’s experiences and adaptation via a direct observation and treatment from a tutor during a game session was difficult and complex to administer because strict qualifications for the tutors, beta players and observation targets had to be fulfilled. Furthermore, it raised a feasibility issue as to when abundant observation scenarios should be held to perfect the adaptation model.

Such issues that emerged regarding the stages involved in developing an adaptation of a game-based learning are the core motivations that drive this research. We challenge ourselves to develop a personalisation strategy for the SEG content that is
non-intrusive, more efficient and feasible. It is worth noting that the ultimate goal of
the research is learning and fun, which can be useful for general purposes. Such as
various techniques involving knowledge improvement or reinforcement, enhancing a
sporting ability, or improving musical knowledge. Yet, how do we provide the best
content for the target users to allow them to learn and have fun simultaneously?

1.4   Our Approach to Developing an Adaptive SEG

In general, we divide our research into three modular stages: development, non-intrusive
assessment and adaptation.

The development stage enables the developer an efficient way to build the SEG
by appropriately organising the adaptable content. In previous approaches, knowledge
and game content spaces were correlated by the design that experts had defined [19,
31]. In contrast, there is a possibility that two separate content spaces can be combined
to create an SEG. The two separate content spaces are publicly available in today’s
technology and this could be a potential alternative to produce a serious game more
easily. Instead of intensive work by an expert, we expose the information retrieved
from the available resources and inherent properties of both content spaces to annotate
and categorise them. A hierarchical clustering apply here, depending on the feasibility
of handling the content space. Given both content spaces are appropriately categorised;
the next step becomes much simpler for the developer to establish the algorithm that
map both content spaces seamlessly. As a result, this stage will produce the content
module of the SEG that one can flexibly generate playable game stages for a player.
Hence, it can potentially make the development process more efficient with minimal
involvement from experts.

An assessment is crucial for the adaptive serious game [16, 18, 22, 32, 33]. Our
goal is developing a non-intrusive assessment method that maintains the immersion
in the game. It is based on the principle that learning is observable via the change of
behaviour [34]. In the serious game’s case, the player’s actions are the source of this
noticeable change of behaviour, reflecting their learning. Meanwhile, the enjoyment
is a general affective driven by various responses. In daily life, enjoyment is also ex-
pressed from facial, auditory or visual appearance. In the field of game study, these are
observable when a player feels excited when they accomplish a game task, or a mis-
mission is challenging for him/her [35, 36]. Thus, in order to generalise the relationships
between the in-game data consisting of game content and play-log features with an
experience category of the player, we perform a couple of procedures. In one, we run a survey for data collection. In the second, we apply supervised machine learning to build the predictive model of a target’s experience that is developed using the collected dataset. When applied, the assessment module can predict the player’s experience confidently and non-intrusively via the in-game data.

Once the assessment has produced some meaningful information, the next stage of the adaptive SG is the content regeneration or performing of a dynamic intervention (known as adaptation) [19]. The challenge at this stage is to create the appropriate adaptation method given the two content spaces and several experiences of the player. Instead of a manual and interrupting reconfiguration of the content, we develop an adaptation agent that generates an optimised series of SEG content through time. The agent learns the mapping between previously generated content with the corresponding player’s experiences. Hence, the adaptation method is no longer forcing the players to a manual configuration or an expensive technique. Rather, our method is self-reinforcing its adaptation information time by time.

1.5 Aims and Objectives

A number of methods and approaches have been proposed to develop a SEG (e.g. [14, 19, 21, 37, 38]). However, very few research has been done wherein two separate content spaces are explicitly combined inside an adaptive SEG. Moreover, no methods have been proposed in a situation where the experts are highly limited.

Therefore, this research proposes a three-module adaptive SEG development. Some modules’ developments involve a rule-based approach, some others are proprietary algorithm—when it is feasible for the developer—and/or the relevant machine learning techniques. The key points to our approach are: (a) the mix between two separate content spaces, (b) a non-intrusive assessment, (c) machine learning applications, (d) a content personalisation inspired by reinforcement learning.

The hypothesis of this research is: *a player’s learning is more efficient when the SG is adapted based on his/her learning and enjoyment*. There are some challenges to verify the hypothesis. We summarise them in the following research questions: (1) How to map different types of content spaces? (2) How to generalise experience-specific in-game behaviour? (3) How to generate the suitable content from both content spaces based on the predicted experiences of the player?
The main contributions of the work presented in this paper are summarized as follows: a) we propose a novel framework for effective and efficient SEG development; b) we test the produced SEG with human players via user survey and statistical analysis; c) a dataset containing the public-annotated game data; d) a generic method for constructing the non-intrusive assessment; g) an adaptation method that works for both content spaces hierarchically; and h) the modular design allows us to design, test and assemble them iteratively and flexibly.

We conduct this research with support from a test-bed game that potentially draws in a player’s attention and focus. The game is called Chem Dungeon, which is a rogue-like in-a-maze game that helps a player to memorise 100 chemical compounds [1]. This serious educational game is a new product contributed under our development framework. The next significance of the resulting research is that the non-intrusive assessment is able to predict the player’s learning and enjoyment via game data with overall performance metrics above 80%. Most importantly, the adaptation method allows the virtual player to completely recall 100 chemical compounds at least twice as fast when compared to the same virtual player who plays the SEG using a brute-force mission generation. Furthermore, the adaptation method applies a spaced repetitive learning strategy ensuring that the player’s knowledge is improved during the process.

1.6 Thesis Structure

This thesis is organised as follows. In Chapter 2 we introduce related background and basic Machine Learning techniques that will be critical to the rest of this thesis. In Chapter 3 we will describe our proposed framework for the serious educational game as well as the demonstration to develop a new serious educational game under our framework. The produced SEG will be used throughout the research. In Chapter 4 we will elaborate our method to build the non-intrusive assessment which predicts the player’s learning and positive affective experience via game data. In Chapter 5 we introduce our adaptation technique, addressing the justification for this approach and the enabling techniques that are used. Finally, in Chapter 6, we conclude by summarizing the contributions of this thesis and further work that could be done in the future.
Chapter 2

Background

In this chapter we explain some related information underlying the project we are in. We begin with an overview of some existing development frameworks followed by the available assessment methods and adaptation strategy. Then, we describe the machine learning techniques applicable to solve the problems in adaptive SEG.

2.1 Games for Learning Purposes

Serious Educational Game (SEG) refers to an alternative learning methodology that applies game technology to primarily promoting players’ learning along with gaining positive cognitive and affective experience during such a learning process [17]. Elements of challenge and learning within such a game construct activities for motivation and amusement [4]. SEG is also named in different terminologies such as game-based learning or edu-game. In this thesis, we treat all these terminologies interchangeably and refers the SEG development to the procedure that builds up a game for a learning purpose.

To give a better perspective of developing an adaptive SEG, we elaborate some literatures in three parts: i) development techniques, ii) assessment methods and iii) adaptation strategy.

2.2 Recent Techniques for Developing Serious Games

This section’s paragraphs elaborate some of the existing techniques of a serious game development. In general, most of the methods are complex and they are applying a
collaborative strategy. Particularly, the design of the game play, game mechanics, scenario and other serious game’s elements sounds proprietary to the knowledge. On one hand, the game elements represent the knowledge accordingly. On the other hand, the immense resources and time consumption are inevitable. Consequently, the following subsection explains a contradicting perspective of development strategy where content spaces are independent and highly available in today’s technology. Therefore, this standpoint underpins our proposed method that can solve the existing issues of the current development techniques.

Sara and Jarvis [11] in their paper used a four-dimensional framework that originally allows education practitioners to choose and apply games effectively in their learning practice [39]. This framework excluded the game development process, instead, it was used only for analysing and measuring some (commercial off-the-shelf) games with respect to the learning goals in practice. In [11], they apply the framework into the development procedures. The original framework highlights the importance of four main aspects of a serious game. First, the SG’s context that defines the game’s utilisation. Some examples of contextual factors including: application of the SG in a classroom setting or in an outdoor/natural environment. One can involve technical supports and the correlation of the SG with the target location/environment. Hence, it affects which genre or type would suit the application (e.g., first-person shooter, arcade, role play, simulation etc.). Second, the specification of the target users, a.k.a the players, is often defined as a complex community that its characteristics are critical for the effectiveness of the SG in practice. They further argued that collecting the target user specifications determines the success of a SG [11]. The third is the representation of the SG as the creative part of the SG developers. The appropriate representation should make the players persevered and engaged with the SG. Such a representation requires a deep understanding of the characteristics between the knowledge and the SG design; thus, made them the prominent aspect inside the SG [11]. In the fourth, the SG developers must pay a good amount of attention to the theory and approaches used in the SG’s application. In earlier work, they found that learning processes are supported by associative (i.e. instruction-based and often task-oriented), cognitive (constructivist) and situation-based perspectives [40]. These perspectives often drive the learning progression at various points. A game-based learning or simulation sometimes far from a learning experience, but it depends on how the application of a learning strategy inside the game. For instance, e-learning applications may adopt the experiential or problem-based learning strategy [41, 42] interchangeably in the
2.2. RECENT TECHNIQUES FOR DEVELOPING SERIOUS GAMES

process [43]. For SG developers who may be lacking of knowledge of learning theory can collaborate with tutors and learner groups to focus on the SG design in best practice [13].

They revealed that the successful educational application was built upon the intensive focus towards the target users. Thus, the game-play, game mechanics or functionality must be driven by this information [11]. Such strategy requires a strong collaboration between the target user group and the development team. Therefore, Sara and Jarvis created a SG development process consists of seven steps [39]. The first step is collecting user requirements by interviewing individuals from the target players. In the second step, categorisation of the cases into correlated graphs based upon the interviews results. Third step: experts validate the relationships of the categorised cases. Fourth step: designing the SG scenarios and refined to particular requirements. Step five: developing the functional specification of the SG. Step six: beta-testing the SG with a group of selected players. Step seven: iterative testing with a broader group of players. These steps clearly show that the approach is thorough and intensive that requires high cost and high time consumption. Hence, this framework is not applicable for every teacher or developers to develop serious games.

Some alternative frameworks proposed in the past are summarized in the followings. Hirumi [9] believed that edu-games should integrate game design, game content and education theory to ensure the development of a successful game-based learning. If education experts solely developing the SG, the output may not be exciting and lacking engaging SG game design. In contrast, if game developers let alone dominating the design process without an appropriate education instruction, the resulting educational games may fail to apply an important pedagogy as the prerequisite for an effective learning. An extensive elaboration is a necessity for designing serious games that integrate learning strategy with a game development process to optimise games-based learning [44, 45, 46].

The education technology is lacking in research on how to design game environments that foster learning at the same time it is entertaining. Therefore, Kiili proposed an Experiential Gaming Model which expected to help game designers understand the learning process in games by embedding pedagogical aspects into game design procedure [12]. It underlined that Flow Theory [36] was the prominent aspect promoting the optimal learning of the players. Kiili disputed that providing the large resources, such that commercial games require them, were not affordable for the educational games’
CHAPTER 2. BACKGROUND

development. Therefore, she applied the formative design procedure that reduces expense and time consumption in her model. She proposed three main goals in the model, which include: 1) Description of knowledge transfer via game, 2) Design of Flow inside the learning-based game and 3) Abstraction of the learning-based game design. Her approach allowed a fast prototyping because it organises a pre-development process ensuring the developers refine the game content and features when necessary. Additionally, it allowed the target users to get involved in the process. Unfortunately, the design technique applied in the high-level description and its guidelines for the educational game development process was rather complex [10]. She argued that the model was applicable for developers with a limited funding to create a good educational game.

More frameworks and models of serious games development process are available in [10, 47, 48]. These approaches emphasised that the design of a serious game is, generally, driven by aspects inherent in the learning materials and education strategy. Moreover, the proposed development frameworks require rigorous procedures that involve interviews with target users (including teachers and students) and various experts (e.g., game development, education, psychology and so on), lengthy development stages and testing units. Hence, those development frameworks have to rely on a close relationship between learning materials and game design (proprietary educational game). Such development frameworks inevitably incur the high cost as their development procedures are laborious and time-consuming; thus, limit the growth of educational games.

2.2.1 The Two Main Components in Serious Games

Based on our observation on produced serious games and a number of relevant literature, there are generally two main components in an SG: knowledge and game content spaces [19, 31]. The knowledge space is formed to encode learning materials concerning the subject knowledge to be learned by players, while the game content space contains playable game elements that convey the knowledge chunks implicitly. This is generally required by any serious games as argued in [49, 50] where a serious game is defined as a computer program that combines serious (for knowledge learning) and game (for entertainment) purposes. Thus, mapping the knowledge space on to content space becomes one of the most important goals in SEG development. To our knowledge, however, the mapping is a bottle-neck in SEG development as this has to be handcrafted by game developers closely working with education experts in most of the
2.2. RECENT TECHNIQUES FOR DEVELOPING SERIOUS GAMES

existing SEGs.

As argued by Damir et al. [19] based on their interview with education experts, game developers and players who involve in SEG, it was crucial to have a seamless connection between knowledge and game content spaces in SEG development. Moreover, they further emphasised that two spaces must be controllable [19] to allow for gaining the controllability in tailoring game elements that are likely affecting different kinds of the player’s experience, e.g. learning, enjoyment, motivation, engagement and so on. Moreover, Hussaan et al. [31] suggested that there are three components in SEG. Apart from learning and game resources, domain concept should be introduced to specify the relationships between learning materials to facilitate strategies in carrying out learning based on game resources. Nevertheless, this approach [31] emphasised that all of those components have to be carried by education experts via interactions with students or game players.

Alternatively, gamification [51] is a combination of learning and gaming that explicitly takes knowledge and game elements into account in development. The basic idea underlying gamification is directly embedding game elements (e.g. avatar, badges, levels and scores) into the learning process. Doing so, it expects students actively engaged in the learning process when they are situated in a game-like presentation of the learning materials. In this work [51], the connection between two spaces were handcrafted by both education experts and game developers, which was laborious and time-consuming. Similarly, Bellotti et al. proposed a generic approach for adaptive experience in serious games via building up the proper connection between knowledge and game content spaces [14]. In their approach, a serious game contained a hierarchical structure of types of game elements. At the leaf nodes, they allocated subsequent tasks by considering diversified connections between learning materials and game elements. Then, adaptation was carried out by personalising tasks’ sequence to an individual player to maximise their positive learning and positive affective experience [14]. However, the game design (in particular: mapping between two spaces) heavily relied on education experts, and it was infeasible to develop such serious games without involving education experts. Due to such an intense requirement, the cost of serious game development often rocketed. Technically, such an approach was also subject to limitation since the mapping task became extremely challenging when either of the two spaces is of a high complexity or size.
2.3 Assessment Techniques for an Adaptive SEG

Assessment is one of the important procedures in a learning tool such as this SEG is built for. An assessment measures the typical experience a player must obtain when using the SEG. To elaborate on this, this section gives the definitions of learning and enjoyment as the main purpose the players must achieve. In the corresponding section, each definition is focusing on its observability from the target object who experience it. Then, the next subsection explains basic methods currently available for assessing the knowledge and enjoyment in the education applications. The explanation of an assessment technique clarifies the simplest yet interruptive way to obtain data from target users. Hence, they are useful utilizations for our research such as allowing a user to label his/her game data with his/her reported experience.

2.3.1 What is Learning and Enjoyment?

*Learning* is a change in behaviour that lasts, is noticeable, and entails internal activities such as thinking, expression of attitudes and emotion [34]. Clearly, learning is a multidimensional activity where interpreting it is an arduous and exhausting work. The common way to measure learning is imposing data from the player [32]. However, since learning is a complex activity, it is still difficult to measure learning accurately. Even to demonstrate whether learning has taken place (i.e. learning or not) is a complicated, timely, expensive, and difficult process [52, 53].

However, we believe that when learning has acquired a knowledge, one applies it to solve a relevant task. For instance, a robot that *knows an obstacle* will try to find another path to reach a goal. In this example case, the robot has the *knowledge* of defining an *obstacle*. This knowledge is obtainable by the robot via trial and error, or other forms of learning. This example shows a change of behaviour in finding the best path to the goal when the robot learns the knowledge, instead of obstructing the obstacle when the robot did not *know* the obstacle. In serious games, a player learns a number of skills in progress (e.g. navigating, destroying an enemy and reaching the goal) as well as learning the *knowledge* embedded in the game scenario. Although learning in SEG sounds complex and difficult, it is observable from the *change* of the actors' behaviours.

Meanwhile, *fun or enjoyment* has a long and deep exploration by research in psychology, multimedia, and video games. Enjoyment is exhibited when a user is aware...
that his effort on doing a task is worthwhile because the skill is sufficient to overcome the challenge [36]. From a number of models that have been constructed, Flow theory [36] is probably the complete model that clarifies enjoyment universally. The theory reveals a flow state which is generated when the challenge of a task is suitable for the skill of the user. When the skill is significantly lacking to overcome the challenge, anxiety results, or conversely, boredom will be experienced. So, flow state can be described as a deep enjoyment of the user when performing the task in which a great amount of effort is worthwhile because the skill is sufficient to overcome the challenge [36].

Subsequently, Sweetser (2005) developed a subset of Flow theory called GameFlow which focused on the criteria of player enjoyment in the games. In the literature [54], the aspects are:

1. Concentration: the game should be designed to require the player to concentrate.
2. Challenge: the game must be challenging and importantly have sufficient balance with respect to the skill of the player.
3. Player Skills: player’s skill development and expertise must be supported by the game.
4. Control: players take full control of their actions.
5. Clear Goals: all goals in the game should be presented clearly at all times.
6. Feedback: players should receive clear feedback on their actions in the game.
7. Immersion: players should feel deep and effortless involvement during game session.
8. Social Interaction: by playing the game, there should be opportunities to have social interactions.

To ascertain that enjoyment is supported by the game, those criteria should be met by the game design as well as exhibited by the players (i.e. concentration, skill, control, immersion, and social interaction) during game sessions. The criteria in GameFlow is a useful foundation to assess the players.
2.3.2 Assessing Learning in Serious Games

A serious game (SG) is a tool to help a player learns a knowledge and at the same time offers enjoyment [2, 55]. Primarily, learning is the main goal of playing SGs. Therefore, it is important to note that SGs or educational games or game-based learning teach something to the player and assess his/her knowledge [17]. By assessing the player, we can assure whether the goal of playing a serious game was achieved or not [18]. An SG is practical when it provides assessment which has an appropriate design on the platform of education and game, and maintains immersion in the game. The next paragraphs explain some known methods for learning assessment in its legacy application.

Quantitative learning assessment is designed to serve different goals. A standard goal is evaluating individuals’ proficiency on specific skills or knowledge compared to others in the same population. This allows teachers to grade students into performance levels, but does not indicate how and when the knowledge was developed. Such assessment summarises the outcome of all learning since day one. However, in SG research, it is usually of more interest to assess the effectiveness of playing an SG and learning during a specific (often short) period, such as time consumption of a game session. By controlling variables (e.g. duration of a game session, a number game sessions), such a measurement can be designed to evaluate: (a) the relative scale of learning ability of a player among a population playing the same SG, or (b) the effectiveness of different instructional methods applied on a single population [25]. This type of assessment emphasises the change of a player’s knowledge in a specific scale, which requires at least two measurements conducted at different times.

A popular method in an education is to use pre- and post-treatment exams to evaluate the relative effectiveness of instruction on student learning. Applying this method in SG for a specific purpose is acceptable, such as observing the success of the use of an SG compared to other learning methods. In this project, this measurement is only intended to measure individual learning gains when playing an SG for a limited time, but not for end users. The results from both exams are then evaluated with an index \( g \), the ratio between the score difference of post- and pre-treatment scores and the maximum possible value of that difference [25], i.e., \( g = \frac{y - x}{100 - x} \), where \( x \) is the pretest score, \( y \) is the post-treatment score, and scores are of a range between 0-100. This half-century-old pre/post gain index independently utilised by Lazarsfeld who called it the effectiveness index [26]. This measurement method can quantitatively show the difference of a player’s knowledge before and after a teaching-learning method. In an
educational game, this assessment method is applicable to impose the player’s learning gain. However, we have to assume that players are compromising such interrupting activities other than playing the game.

Consecutively, what is a measured score represents? Due to the complexity of cognitive processes, there are much possible education and or cognitive variables underlying an observed score, most of which are difficult to be precisely determined. For example, a low score can be the result of a range of cognitive processes such as incorrect applications of a knowledge, applications of irrelevant knowledge, or unlucky guesses. In this project, these possible causes to a measured score will not explicitly be expressed in the model and the variations caused will be treated in general as uncertainties of random origins. Based on such limitations, a player’s knowledge state is then described with a score representation that has two states: measured-correct and measured-wrong. In one, the measured-wrong state represents in large the application of scientifically incorrect knowledge and the lack of appropriate knowledge. And in two, the measured-correct state represents in large the successful application of scientifically correct knowledge [56].

2.3.3 Assessing Enjoyment in Serious Games

Questionnaire is the known method to perform the enjoyment assessment, such as EGameFlow [27]. It is not surprising that a questionnaire has lots of thorough questions. It can help a player reports his skills. Pedersen uses four alternative forced choice (4-AFC) questionnaire which consists of self-reported experiences [35]. Robert uses a questionnaire with a Likert scale to estimate a player’s gaming skill [28]. Nevertheless, Likert scale is prone to many biases [57]. For instance, a participant may “afraid” of making extreme responses, a tendency to “please” the experimenter by making agreement statements, or reflecting themselves untruthfully reflected in the responses [57]. Hence, afc-questionnaire is the preferable choice for this project, because it allows a player to give a less biased and more ideal answer [58].

Originally, the alternative forced choice questionnaire is the 2-afc. The 2-AFC experimental design is commonly used to test speed and accuracy of choices between two alternatives given a timed interval [59]. The task is an established controlled measure of choice and is widely used to test a range of choice behaviours in animals and in humans. The basic components of a 2-AFC task are (1) two alternative choices presented simultaneously (e.g. two visual stimuli), (2) a delay interval to allow a response/choice, (3) a response indicating choice of one of the stimuli. Alternatively,
4-AFC is more efficient than 2-afc [60]. For instance, 4-AFC consists of four choices comparing Level X and Y. The example of choices are as follows:

- Level X is Harder than Level Y
- Level Y is Harder than Level X
- Both levels are Hard
- Both levels are Not Hard

With this questionnaire, a player is forced to give accurate or approximate answers in relatively quick respond [60].

### 2.4 Adaptation Techniques

In general, one can picture a set of interacting entities (tangible or abstract) which construct an adaptive system. These entities collectively respond to environmental changes or the dynamics in the interacting elements, as such is found in adaptation in biology. The key to these adaptive systems is feedback loops, such as the organisms and objects that build ecosystems; or in the game field, game engine, user interface, players, and game content. Commonly, artificial adaptive systems utilize negative feedback to preserve expected states. This means that any difference coming from the feedback triggers the system to balance its output accordingly.

In [61], adaptation systems saturates to a situation in which all stimulus suspend. In an adaptive system $AS$, a feedback $I$ is a stimulus for $AS$. As such is true if and only if the probability $P(AS \rightarrow AS' | I)$ that $AS$ must be altered (e.g. its process or entities) given the feedback signal $I$ is strictly greater than the prior probability of $AS$ modification independent to $I$:

$$P(AS \rightarrow AS' | I) > P(AS \rightarrow AS')$$  \hspace{1cm} (2.1)

Referring to Eq.2.1, let $AS$ be an unknown adaptive system subject to modification in time $t$ and let $I$ be an arbitrary stimulus for $AS$. One can assure that $AS$ is an adaptive system strictly when $t$ tends to infinity ($t \rightarrow \infty$) the probability of changing $AS$ to $AS'$ in a time step $t_0$ given $I$ is equal to the probability of $AS$ change to $AS'$ excluding the $I$. We can derive such explanations mathematically as follows:

$$P_{t_0}(AS \rightarrow AS' | I) > P_{t_0}(AS \rightarrow AS') > 0$$  \hspace{1cm} (2.2)
2.4. ADAPTATION TECHNIQUES

\[ \lim_{t \to \infty} P_t(AS \to AS'|I) = P_0(AS \to AS') \]  

(2.3)

Hence, for each instance of \( t \) results a temporal interval \( dt \) such that:

\[ P_{t+dt}(AS \to AS'|I) - P_{t+dt}(AS \to AS') < P_t(AS \to AS'|I) - P_t(AS \to AS') \]  

(2.4)

In conclusion, an adaptation is an iterative process whenever the probability of the content change due to a given feedback signal is greater than the probability of the content change without any trigger.

With regards to the definition above, the following subsections summarise earlier works corresponding to the most applicable adaptation methods and the learning strategy via repetitive recalls. The purpose is to introduce the existing adaptation methods and how they are related to our research. In addition, the SEG we are focusing applies in the low-level learning, i.e. memorisation, where it entails the same circumstance that the game holds: a repetition. Therefore, we argue that both purposes (i.e. gaming and learning), often described as contradicting goals, are actually complementary when a relevant learning method (i.e. spaced repetition learning technique) applies.

2.4.1 Player-centric Adaptation in SEG

Serious games have introduced an adaptation to support players’ learning by providing their needs. For instance, Magerko preferred to personalise an educational game based on the motivation of the player [30] in which the following paragraphs explain the three approaches.

The first method requests a player to manually specify his/her motivation type using a questionnaire. In some cases, this method is the simplest one to prepare by adding a simple questionnaire in between game stages and requires the player to complete it to progress the next game stage. The author and serious game developer should aware of the interruption caused by using this method. They should design the questionnaire carefully to minimise the distractions and preserve the focus of the player with respect to his learning and enjoyment. For instance, we must avoid embedding a questionnaire with too many question items. Despite that it may frustrate the player, it also causes a complexity of formulating the questionnaire results into the input variables for the serious game’s adaptation. Preferably, a simpler and direct questionnaire can minimise the distraction of the player and the input formulation. Alternatively, a game-like questionnaire can also be applied. But this requires more development of game elements representing the questionnaire inside the game.
Second approach: a player is manually customising the serious game based on his/her subjective preference. This method is commonly seen in entertainment game where a player configures the game that suits his/her preference. For example, the difficulty level, the next mission and much more. This method sounds a bit similar to the questionnaire approach; however, it offers a direct configuration to the content specification. In addition, the player must have an extra motivation to drive them adjusting the game stage when necessary. In many cases, players tend to configure the game once, such as in the commencement of the game, and strives to succeed all the missions under that configuration. Once he/she has accomplished them, he/she reconfigures the game when he feels it is convenient to do so.

The last method expects a player to play a preliminary game session as a way to recognize his/her motivation type. Although it is not disrupting the game session, there is a point that a player’s experience/preference may change over time that an adaptation handles such a case. Hence, relying upon the preliminary evaluation result may become obsolete up to a point that requires the player to repeat the preliminary game session. Alternatively, the preliminary session helps to launch the game that suits the player and the subsequent game sessions become the ongoing evaluations of the adaptation.

Therefore, a more convenient personalisation should take into account the ongoing game playing experiences of the player and keep monitoring it non-intrusively to maintain his/her focus [19]. Consequently, adapting the game stages automatically based on the assessment results.

As suggested in the former methods, adaptation is concerning the manipulation of the target game object to satisfy the player’s need. Various target adaptations are available and proposed in the existing approaches such as game elements [14, 19, 38] and textual message intervention [15, 16, 22, 23, 21]. From these proposed methods, they are all attributed a descriptive information to the target adaptation, enabling a more controllable generation of the object. The text-based intervention uses the semantic meaning of words that reflect a motivating advice [16]. Meanwhile, game element adaptation for SG carrying abundant challenges is our main interest. Because game element has plentiful potentials to produce diverse environments and experiences, rather than a text intervention in an SG. The adaptive SG that exercised this principle organised their adaptable content based on the corresponding information. In this regards, a hierarchical content structure makes the SG content is highly manageable for an adaptation. Then, modelling the adaptation is the critical step that generalises the mapping
2.4. ADAPTATION TECHNIQUES

between SEG content and the desired characteristic of the player (obtained by an assessment). A method in [19] suggested an observation of players conducted by tutors; alternatively, experts mapped the players and the relevant SG content based on its attributes [14, 38].

2.4.2 Spaced Repetition Learning

Spaced Repetition Learning (SRL) is a technique for efficient memorization by repeatedly testing a particular item to students rather than delivering numerous tasks in a short amount of time, each repetitive routine is expanded through time between repetitions as one recalls the education item. Some earlier research uncovered the potential of SRL for enhancing memorisation against the traditional learning method, e.g. [62, 63, 64].

A classic method of remembering an item is through acquiring (e.g. reading) the chunks of knowledge, while the SRL is basically repetitively testing oneself with the items to memorise. This SRL concerns the fact that humans’ memory declines over time, thus, interval-based frequent tests on a particular item will refresh the memory. The analogy is similar to a rechargeable battery, if the battery is left unused for a prolonged time, the charge will slowly degrade. Meanwhile, a repetition in learning is analogous with recharging the battery to a fully charged again.

SRL has been implemented in a computer-assisted language learning that allows players memorising vocabulary and grammars such as Duolingo [65] (a gamification of online language learning). In the non-computer-assisted learning, Spritzer found that SRL was efficiently helping students learning science facts [66]. These practices showed that examination (applied in SRL) is acknowledged as a learning tool, in that it opposed to the mostly known as measuring instrument of a student’s learning achievement.

Recently, there are various algorithms regarding the application of SRL such as SuperMemo or Anki [67]. Basically, the inputs of the spaced repetition learning are the performance of the recalling test, e.g. self-report of memorising strength, the correctness of the answer and time spent answering, and the estimation of memory decays. Then, a scheduling task estimates the occurrences of a particular test based on the interval between tests.
2.5 Rule-based Systems

Rule-based systems contain methods to store and alter knowledge to represent information in a useful way. Commonly, the term used in this project corresponds to systems involving developer-crafted or manually established rule sets. In contrast, we exclude the rule-based systems that apply automatic rule inference.

A rule-based system consists of rules or rule base to help users or clients draw a conclusion or make a choice. A rule base stores rules for the system. In that, a rule is constructed as an ‘if’ (i.e. condition, premises, antecedent) and a ‘then’ (i.e. conclusion, action, consequent) parts. A ‘true’ conditional test executes the ‘then’ statement which is known as firing the rule. These rules are commonly known as production rules, dissimilar to the conventional branching in programming. In a production system, there are pairs of if-then stored in the rule base and a ’controller’ that has a particular method to choose the rule for a certain situation. Once the chosen rule is set, the system applies it.

Typically, a rule-based method runs iteratively that starts by obtaining new information and keep them in the working memory. Then, the controller selects the matching rule from the rule base under which the new information should be tested. Consequently, the inference engine executes actions in the ‘then’ part of the successful ‘if’ test (as known as recognise-act cycle). The inference engine is then direct the output produced to the working memory causing modification of its content. As such marks the next iteration of the rule-based method.

Developing a rule-based system often administered in an iterative process. According to [68], the development process starts by acquiring bits of knowledge (if-then pairs) about a specific subject and keep them in the rule base. Experts and the system engineer are working together in this step to constitute the knowledge representation to be added into the rule base. In each iteration, the expert needs to ensure the consistency or complete information of the rule base. At this point, we do not need to worry about how and when to use the rules. Because the production system will carry out such duty. Given the small-scale rule base, a naive approach that checks each rule against known facts is feasible. Otherwise, the Rete matching algorithm [69], for instance, should be efficient. With this structure, knowledge is expandable by allocating additional rules at the end of the rule base. Moreover, relevant experts can interpret and understand them for necessary improvement or refinement. With the growing size of the rule base, an information in the working memory may match with more than one rules, we call this a conflict. In a rule-based system, one should consider a conflict resolution strategy...
2.6.MACHINE LEARNING

Figure 2.1: Rule-based system architecture (reproduced from various sources).

(CRS) to choose which rule to fire. The simplest yet favourable is by attaching each rule with a numeric priority. If two or more rules available for a data, choose the one with the highest priority value. However, if these rules have the same highest priority numbers, choose the oldest rule.

Rule-based systems often used in the case where problems have specific visible pattern and the developer has the knowledge to establish the rules.

2.6 Machine Learning

Machine learning is a method to construct intelligent algorithms that learn from existing data, henceforth generalise optimally, specifically, give predictions on new data. In Chapter 3-5 we will review existing techniques which make use of machine learning for the purposes of the adaptive serious game, and our own method described in each chapter that makes extensive use of it. As such we will provide a review of the relevant literature in this chapter.

In [70], phase-functioned neural network (PFNN) is the agent to control a 3-D character motion. Instead of manually code the characters motion in a 3D game, we can apply such an approach given the dynamic surface such as in the procedurally generated game arena. In addition, there exist approaches that utilising machine learning algorithms especially in the procedural game content generation. For instance in [28] and [29], unsupervised machine learning takes part to select representative games prior
to categorising the playable/unplayable games. Meanwhile, active learning algorithm is applied to differentiate good and bad games [28], [29]. These literature inspire us to apply them in serious game’s development; especially, the part where its content is somewhat similar to entertainment game’s. Currently, no serious games have applied unsupervised machine learning. Thus, there is an open challenge to adopt unsupervised machine learning approach from entertainment game into serious games.

Existing methods in adaptive serious educational game have utilised various machine learning algorithms (e.g., [71], [14]) and we apply them in our framework. In Section 2.7, we will describe supervised learning techniques that learn from labelled data. In Section 2.8 we will review unsupervised algorithms, especially clustering, which learns from the unlabelled data. In Section 2.9 we will look at reinforcement learning algorithms. And, finally in Section 2.10 and 2.11 we will briefly review resampling methods to produce relatively balanced training data and the relevant machine learning performance validation, respectively.

More elaboration of the algorithms covered in this section can be found in most machine learning books [72, 73].

### 2.6.1 Fundamentals

The aim of machine learning is to program computers using sample data or previous experience to solve a given problem [72, 74]. Thus, it is closely related or even overlaps with computational statistics which requires collections of data known as samples [72, 74]. The problem in statistics is optimising the use of samples from unknown probability distributions to help in making decisions from which distribution of some new samples belongs. Machine learning uses the statistics for training examples to produce a new function that decides or estimates a given new sample. Formally, a computer program learns from experience $X$ with respect to a number of outputs $Y$ and performance evaluation $P$, when the performance $P$ at outputs in $Y$, improves with experience $X$ [73]. The performance $P$ strongly correlates with the probability of an output $y$ given an input $x$. Hence, Bayesian theorem is the appropriate explanation of such a definition.

$$P(h|C) = \frac{P(C|h)P(h)}{P(C)}$$  \hspace{1cm} (2.5)

Reverend Thomas Bayes originally stated the Bayesian theorem in that it basically a way of understanding that the machine learning goal is finding the highest probability
2.7 Supervised Machine Learning

In this section, we will explain some details of supervised machine learning. Algorithms categorised in this technique are trained using training dataset comprise of ordered pairs of the form \((x, y)\). Where \(x\) be the sample input and \(y\) is the label or output of a target function \(f\) for it. Once trained with reliable and sufficient training samples, we expect the algorithm to confidently predict the output \(y\) via the function \(f\). A supervised machine learning has the ability to off-line learning a set of data to develop a generalised model such as a player’s affective experiences [35] in an entertainment game or content preference [28, 29]. Given the adequate samples, we will apply supervised machine learning to build a predictive model as an alternative assessment tool in the adaptive SEG.

We explain the common methods for developing a predictive model using either regression or classification under the supervised technique, depending on the typical output of the samples. Then, we introduce decision tree algorithm that is easy to interpret the decision rules, followed by Random Forests algorithm which contains weak decision trees. These Random Forests model were proven to perform better than an individual model, such as [75]. In addition, it generalises well even the samples are of...
a very high dimensional space. Especially, when we need to generalise an experience of the players which is originated from complex relationships between input variables. Subsequently, we also explain Artificial Neural Network (ANN) that we need it to act as an intelligent generation agent of the SEG’s content. This is due to the fact that it fits the conditions of the agent which include a small number of samples and potentially complex non-linear relationships between dependent and independent variables. This ANN will not be developed as an individual, instead, we will use it as the black box of the reinforcement learning.

2.7.1 Regression and Classification

Regression is a problem in estimating a continuous value for a new set of data. Given a new input vector, the regression function predicts the continuous value of the output based on the established linear model. Regression model may fit the problem where there is a function $f : X \rightarrow \mathbb{R}$, where $X$ be a set of input vectors describing the features in a game log such as the game duration (real value), total kills (discrete value), and the experience value (discrete value). The goal of the function is predicting the real-valued output, such as the measures of players’ gaming skill represented by game scores [76], $\hat{y}$, given previous data obtained in the form of $(x \in X, r \in \mathbb{R})$.

Buckley applied four general procedures to build the regression model of the score which includes: data collection via survey, input features extraction, input features grouping and score prediction modelling [76]. Training samples (476 games) were obtained via in-house survey involving 45 participants. Game events, keyboard presses and mouse actions were the input features of the samples. In addition, compression method was applied to measure the complexity of the keyboard and mouse inputs. Prior to building the prediction model, the input features were grouped for better analysis and representation with respect to score. Then, various regression models were built based on a group of input features. As a result, the best score prediction model came from the keyboard group input features with Spearman’s $\rho = 87.6$.

Meanwhile, in educational game research, regression can be applied to induce real-valued output, such as memorisation gain, engagement level or recall score [77]. However, the reported output must be taken care exclusively during the data collection phase either via a reliable measurement method and the selection of players who participated. Otherwise, the regression model will not appropriately picture the correlation between input and output.

Classification, on the other hand, is the problem in identifying the category for a
new set of data. Classification is a subset of supervised learning technique. This is due to it requires a training data set to model the classification which we call training or learning stage. The training dataset has a set of input vector $X$ and a label attribute. The components of input vector $X$ comprise of $n$-dimensional attributes $x_1, x_2, \ldots, x_n$. The training stage comprises of obtaining input vectors and their associated labels. Most importantly, the class label is discreet [74, 72]. The methods in modelling classification are to find the optimum decision boundaries which separate the classes. More formally, a classification problem involves the development of a function $\rho : A \rightarrow C$, where $A$ is a set of training examples and $C$ is a set of classes, such as predicting the winner in a match of a multi-player esports such as Multi-player Online Battle-Arena games (MOBAs) [78]. It classifies the winner based on the encounters which are identifiable when two or more heroes of the competing teams are in range to influence each other. For prediction purpose, the encounter is modelled as a series of events (i.e. time and location) involving the same type of hero. In the experiment, Schubert used 412 samples of DOTA matches. The logistic regression successfully predicted the winner by 78 percent accuracy.

The open challenges in serious game fields are categorising the players’ learning [71], motivation [30] or enjoyment [14]. Relatively the same procedures are applied for such purposes. During data collection a self-reported questionnaire is more relevant for affective-experience reporting, e.g., Four Alternative Forced Choice [79] to distinguish the enjoyment between a pair of games. Meanwhile, pre-post exams are relevant for measuring learning gain [71] as a way to label the corresponding in-game behaviour. Once adequate and reliable samples are obtained, the player’s experience model can be built accordingly. For instance, Manske et al. applied Dynamic Bayesian Network (DBN) to model the player’s factorisation knowledge. The selection of such algorithm was driven by the vast uncertainty of the player’s learning state represented by their gaming performance. In addition, the correlated subjects in the game were also the prominent reason for applying DBN. One DBN was built for each climb-the-mountain factorisation game. The network inside DBN was built from every player’s action representing his/her knowledge evolution (i.e. climb proceed). This shows that supervised machine learning has become the potential solution to generalise players’ experience from information that is carrying unknown relationships with the output.
2.7.2 Decision Tree

A Decision Tree is a flowchart-like tree structure where each **internal node** represents a condition of an attribute and each **branch** represents the path that should be followed based on the condition. The **leaf node** is the terminal node of the tree representing the final decision of the process which holds a class label [72]. Decision Trees can be applied for both regression and classification cases.

Given an input \( x \) in which output value is unknown, the attribute values are tested against the decision tree starting at the root node of the tree and traces downwards on a path determined by what conditions are met. Decision Trees are not specific to machine learning and are common in many fields, especially for exploratory knowledge discovery [72].

Decision Tree Learning is the steps involved in generating a decision tree by learning about the problem. Let \( D \) be a classification data set, with a set of classes \( C \) and attributes \( F \). Let \( V(F) \) provide all the values for the attribute \( F \). The purpose of Decision Tree Learning is growing a Decision Tree, which receives an input \( I \) consists of values for each attribute. At each node, the Decision Tree assesses the condition of an attribute value in \( I \). The leaf nodes give the solution constructed as a class \( c \in C \) to the query \( I \). There are many algorithms for growing decision trees, each with their particular advantages. As an instance, we will elaborate the essential process of the ID3 algorithm for classification tasks.

**Entropy** is the calculation of uncertainty of a random variable or set of data. The entropy for \( D' \subset D \) is specified as: \( \text{Entropy}(D') = -\sum_{c \in C} (p(c) \log_2 p(c)) \), where \( p(c) \) denotes the probability of class \( c \) in \( D' \), such as the frequency of occurrence. For \( v \subset \text{Values}(f \subset F) \), let \( D'_v \) denotes the number of samples in \( D' \) where the value for attribute \( a \) is \( v \). For Decision Tree Learning, we formulate the information gain for an attribute \( f \subset F \) with respect to \( D' \) as:

\[
\text{InformationGain}(D', f) = \text{Entropy}(S) - \sum_{v \subset \text{Values}(f)} ((|S_v|/|S|) \times \text{Entropy}(S_v)) \tag{2.6}
\]

This formula measures the reduction in entropy (i.e. decrease in uncertainty) that can be achieved if we know the value of an attribute. The ID3 algorithm is designed to generate a tree where at each node an assessment identifies the attribute with the highest information gain. The ID3 algorithm described in Algorithm 1 runs iteratively and initializes with \( D' = D \).
Algorithm 1 ID3 Algorithm.

1: **Purpose:** Generate a decision tree by learning the mapping between input vectors to a class of a training example \( D' \).

2: **Input:**
   - Training Data set, \( D' \), which contains of input vectors and their associated class labels;
   - The set of attributes \( I \) which forms the input vector.

3: **Output:** A decision tree.

4: Create a node \( N \);
5: if All samples in \( D' \) are of the same class, \( C \), then
6:   **return** \( N \) as leaf node with label \( C \);
7: else
8:   if \( |D'| = 0 \) then
9:     **return** a leaf node \( N \) labelled with the dominant class in \( D' \);
10: else
11:   With respect to the current data set \( D' \); identify the attribute \( f \in F \) in which information gain is the highest;
12:   Conditional test of the value of \( a \);
13:   **for all** \( v \in Values(a) \) **do**
14:     Generate a sub-tree \( v \) for each condition of \( a \);
15:     Recursive call to (1) in which \( D' = D_v \) to generate the sub-tree.
16:   **end for**
17: **end if**
18: **end if**
An educational game which operates simple and limited statistics may utilise Decision Tree as a quick and highly interpretable assessment model, such as categorising player’s motivation in S.C.R.U.B. game [30]. The collected data from players’ reported motivation can be useful as the training samples for a multi-class classification algorithm. However, SEG often built with complex information to be associated with the player’s learning or other experiences, such as Elektra serious game [22]. Implementing a decision tree algorithm may not be the best option for an assessment purpose. Because decision tree may overfit the training data if there is no careful pruning practised. Therefore, Random Forest (RF) algorithm which composed of an ensemble of weak decision trees is expected to tackle such an issue found in the Decision Tree algorithm.

2.7.3 Ensemble Learning: Random Forest

Ensemble learning method builds a number of classifiers and makes the final prediction of new samples by voting the predictions made by those classifiers [80]. Ensemble method can produce a highly accurate classification by combining those weak classifiers [80, 81]. There are various ensemble learning algorithms, such as boosting, Bayes optimal classifier, bootstrap aggregating, and much more. We explain an example of an ensemble learning algorithm in the following.

Random Forest (RF) [82] is one of the state-of-the-art algorithms in ensemble learning called bagging. Bagging was introduced by Breiman (1996) as an abbreviation of bootstrap aggregating [83]. It creates new training examples (bootstraps), creates prediction models from each bootstrap, and makes the aggregation of predictions. RF was inspired by bagging which creates a number of decision trees (tree ensemble). Decision tree consists of branched rules. When inputs feed the RF, each tree independently predicts an output. Then, RF makes a final prediction either by voting or averaging over decisions made by all trees. Voting works for classification problem while the average works for regression.

The benefits of using RF are as follows:

- Works for various types of prediction problems, especially for those which are unstable as a single model.
- It has relatively good accuracy.
- Resistant to outliers and noise.
• It handles multiple features.

• It produces estimation of error, strength, correlation, and feature importance (automatic feature selection).

• It is simple and parallel-process ready.

• It is fast.

The first step in RF is the \( B \) number of bootstrap samples. Let \( D = \{x, y\} \) be the original \( m \) training examples consist of independent variables \( x_j \) (\( j = 1, \ldots, n \), \( n = \) number of independent variables) and dependent variable \( y \). As many as \( B \) bootstraps \( (D_k, k = 1, \ldots, B) \) are created by randomly sampling from \( D \) with replacement. It means that training examples have the same probability to be selected repetitively. An example of sampling with replacement is as follows. Assume we have a number of marbles in a bag. Then sampling with replacement is done by randomly picking a marble, records the characteristics and values of that marble, then put the marble back into the bag. Therefore, there are possibilities that some training examples are selected more than once for being bootstrap samples.

The next procedure is the tree generation for each bootstrap. Trees in RF are grown using binary split to each node. The randomness applied in node creation where the candidates are \( f \) features selected at random (\( f < n \)). Then, the features with the best split are the node. The node creation process is repeated to the largest extent possible (until a node has a single observation) and no pruning to the tree. The benefit of randomness in node creation is twofold: (1) it seems to enhance accuracy, (2) the combined ensemble of trees gives various important ongoing estimates [82].

During a bootstrap generation, not all training examples are selected. According to Breiman [82], one-third of original training datasets were left out, we call these samples the out-of-bag \( OOB_k = \{x, y\} \). A combination of out-of-bag samples is very useful for making ongoing estimates such as generalisation error, variable importance, and variable correlation [82]. It gives internal estimates which are helpful in understanding classification accuracy and ways to improve it [82].

RF has been used in the categorisation of Quake game contents in Learning-based Procedural Content Generation (LbPCG) [28]. It successfully categorised contents into five difficulty level. To accomplish that, the collected samples were grouped into 16 sub-samples according to participants gaming experiences. Then, 16 RF classifiers
were built upon these sub-samples to create a two-level difficulty prediction. The accuracy of each classification then became the weight of the RF in the lower level that contributed to the final estimation of the difficulty level.

Meanwhile, SEG carries complex information and the outcome is undeniably abstract to be interpreted, e.g., learning and enjoyment. Moreover, there exist high occupancy of observable behaviour with the content inside the SEG that corresponds to such outcome. Instead of applying a sole algorithm which tends to overfit the training samples, RF has been proven as a high accuracy predictors using an ensemble of trees. It generalises well even with multi-dimensional features. Moreover, variable importance is automatically computed along with tree growing; thus, we can identify which features more significant for the prediction. It is also relatively fast in generating the model because of the randomness, which means less computational burden in generating the trees.

Based on our knowledge, research in SEG has not applied RF although it has the potential to improve the previous prediction models, e.g., predicting learning gain [71]. Our project applies RF as well which is explained in more details in Chapter 4. Because our approach is relatively similar with [28] [76]. Thus, a good prediction model can be constructed as well.

2.7.4 Artificial Neural Network

Artificial Neural Networks (ANN) [73] were inspired by the complex biological processes of the brain. However, not all specifications in the brain processes are modelled by the ANN. ANN is structured as a several layers of neurons. Neurons from one layer to the next layer are connected via weighted synapses. ANN are suitable to solve problems, especially the noisy training data which commonly found from the input sensors for real-world problems. The issues mainly exist in applying ANN are as follows: 1) longer time consumption to train them, and 2) the difficulty to understand the solution inside an ANN. However, these do not stop ANN to become the widely-used approaches in the video game research literature.

A weight is assigned to each neuron in an ANN’s inputs and forwards the sum through an activation function. Single Layer Perceptrons (SLP) are ANN consisting of neurons in one layer with an output that thresholds its inputs into a boolean value (0 or 1). An SLP only represent the problems considering linearly separable classification, such as AND, OR, NAND and NOR logic gates.

A more robust type of ANN is the Multi-Layer Perceptron (MLP). MLP consists of
a number of hidden layers with one or more neurons. Unlike the SLP, it can solve both linearly and non-linearly separable problems. One must ensure the activation functions are differentiable as this enables the influence of each weight on the error produced by activation function to be calculated. Hence, it can apply the popular back-propagation algorithm, which basically calculates the network’s error at the activation function. Then, it propagates backwards through the nodes computing the proportion each node and each node’s weights with respect to the error and adjusting them accordingly.

An example of artificial neural network application was the correlation between game logs with players experiences [35], i.e. enjoyment, challenge, frustration, predictability, anxiety and boredom (they may exist in entertainment or serious games). Supervised machine learning methods were applied to build the quantitative models of there player’s experiences. Evolutionary computation (i.e. Artificial Neural Network) was applied to predict the reported experiences in the 480 samples from 120 Super Mario game players. Pedersen applied two-stage player experience modelling. Initially, Single Layer Perceptron algorithms approximated the players’ emotional preferences. In order to identify the best features as the input vector for the SLP algorithms, four different feature selection methods were employed that include nBest, Sequential Forward Selection (SFS), Sequential Floating Forward Selection (SFFS) and Perceptron Feature Selection (PFS). These best features contributing to SLP’s prediction accuracy were then used to optimise the topology of the Multi-Layer Perceptron. Overall, the accuracy of estimating experiences rose around five to seven percent more accurate, e.g, Enjoyment prediction accuracy improved from 69.18% (via SLP) to 74.84% (via MLP).

SEG may also apply ANN in the same manner previously explained as an alternative solution. For instance, Kickmeier et al. assessed the player’s learning progress and motivational states based on Competence-based Knowledge Space Theory (CbKST) [84] and the observable behaviour of the player [16]. However, this approach applied logical rules via complex mathematical model [85]. It might be difficult to develop them because of the specialised expertise it requires. Alternatively, ANN can do that task as long as there are reliable samples provided, such as via survey. The simplest approach applies ANN to a group of skills of the CbKST to predict the learning and motivational state of each skill. The input vector composed of statistics of the observable behaviours. Then, the network in the hidden layers is optimised by training the ANN using the collected samples.
2.8 Unsupervised Machine Learning

In this section, we will review unsupervised learning techniques. These algorithms do not require labels in the datasets. Instead, unsupervised learning algorithms attempt the identification of the natural structure of the data. In this regards, applying unsupervised machine learning is more suitable especially when the samples are too difficult, complex, and the size is too large making it not feasible to be identified or annotated by the user.

In our case, such problem exists in the game content spaces of the SEG that it has a large space making the manual annotation or identification infeasible. The common property that separates these content spaces is the similarity or distance between data points. To our best knowledge, research in serious game fields had yet utilised unsupervised learning. On the other hand, research in entertainment games has applied it to categorise game content space, e.g., [29, 28]. Such approach is viable in our framework concerning the game content space (details of the implementation in Chapter 3.2). In the next subsection, we first explain the distance measurement and followed by the most potential unsupervised learning algorithm for our project.

2.8.1 Distance Measurements

Most unsupervised learning apply a similarity measurement of the data points to construct the clusters wherein the following sections will elaborate some of the clustering algorithms. Before doing so, however, we need to specify exactly how a distance metric between two data items can be established to measure the similarity. In this section, we will consider the input to our problem to be a parameter vector consisting of attributes representing features of a problem space. For example, the parameter vector may represent a car and each attribute some feature of the car such as size, colour, seats and so on. Our objective in this section is, given two such parameter vectors, to describe how similar they are. The immediate issue is that the components of the parameter vector could be of different types. For example, one attribute could be "transmission", which is a binary attribute since it can take on two values "manual" or "automatic". We may also have another value "miles travelled", which is a numeric attribute which may take on a real or integer or numeric value, perhaps from some restricted range of values. It is also conceivable there could be an ordinal attribute called "size", that can take on three values "small", "medium" and "large". The values have
an order to them in which attribute value has a set place among the other values. Finally, we could think of another attribute called "colour", which can take on one of the several values with no particular order, such as "silver", "black", "white" and "red".

In this section, we limit ourselves to the situation where the parameter vector has attributes of dissimilar types, as this is the more general case and the one which is most specific to the approaches we require later in this thesis. Our approach adopts the methods from pages 75-76 of [86]. In our understanding of their approach we assume that the attributes of each parameter vector are always available (no missing values), and we define the similarity metric as the following formula \( \sum_{f=1}^{n} d_{ij}^f \), where \( n \) is the total attributes and \( d_{ij}^f \) measures the difference between parameter vectors \( i \) and \( j \) for attribute \( f \). In more details, the similarity metric informs us the mean distance between each attribute pair in the two parameter vectors. It is immediately evident that we need to be cautious with the distance function \( d \), as we have to guarantee that no one or more attributes dominate the others because of the range of values they may obtain. This is why normalisation is essential.

Given \( x_{if} \) as the value of attribute \( f \) in parameter vector \( i \). For numeric or integer attributes, we define \( d_{ij}^f = |x_{if} - x_{jf}|/\text{range}_{f} \). Where \( \text{range}_{f} \) be the range of values an attribute \( f \) can take and as such \( d \), in this case, is resulting normalised distance in the range of \([0.0, 1.0] \). If the attributes have nominal/categorical and binary values, we define \( d_{ij}^f = 0 \) if \( x_{if} = x_{jf} \) or, otherwise, \( d_{ij}^f = 1 \). In other words, when two binary/nominal attribute values are the same, then no distance is considered among them; otherwise, we consider a distance of 1, regardless of what the two values actually are, which makes sense as there is no particular order for such attributes. For ordinal attributes, however, there is a notion of order and we have to take this into account. We define \( t_{if} \) as the translation of nominal attribute \( f \) in parameter vector \( i \). We define it as \( t_{if} = (o_{if} - 1)/(nf - 1) \), where \( o_{if} \) set the order/rank of attribute \( f \) in \( i \) and \( nf \) be the total number of attribute values for attribute \( f \). We consider the rank of an attribute value as an index, starting from 1 to the number of possible values. Once we have obtained \( t_{if} \) and \( t_{jf} \) for vectors \( i \) and \( j \) we can address the problem as found in a numeric attribute and set that the denominator as 1. Keep in mind that for all definitions of \( d \) we have used, the value returned is always in the range of 0 and 1, which ascertain that no domination of an attribute takes place.
2.8.2 Hierarchical Clustering

While the basic functionality of clustering is partitioning a set of objects into a number of exclusive groups, in many cases partitioning our data into groups at different levels or hierarchy. A hierarchical clustering technique groups our data objects into a "tree" or hierarchy of clusters. Representing data objects in this way is helpful for summarization and visualisation. For instance, a boy may organise his toys into a major group based on the type of toys. Then, he can further partition these main groups into smaller subgroups, such as based on the colour or size. All these groups construct a hierarchy or a tree of toys. In such a way, the boy can easily summarise or characterise his toys which help him searching a typical toy. In addition, the advantage of hierarchical clustering compared to other clustering methods is that it does not need to identify the $k$ value representing the optimal clusters. Instead, it constructs the tree of clusters from the training data points. As a result, one can flexibly choose the method to select the best clusters either based on the maximum clusters (if $k$ is known), the distance threshold, or the inconsistency method. Thus, revealing the clusters depends on the purpose of the analysis of the content space. We will elaborate this purpose more detailed in the next chapters.

The following subsections describe more details of a couple of hierarchical clustering algorithms used in our approach in this thesis.

2.8.2.1 Agglomerative Clustering Analysis and Divisive Analysis

Hierarchical clustering can be structured from the bottom-up way or a top-down. The bottom-up approach is called Agglomerative Clustering Analysis (merging objects) and the top-down fashion is known as Divisive Analysis (splitting clusters) [86].

An agglomerative clustering analysis (AGNES) constructs the clusters tree using a bottom-up strategy. It starts by allowing each object as a single cluster. These clusters are then merged, based on their distance matrix, to form recursively bigger clusters. In that two clusters have the smallest distance to each other, they are merged into one new larger cluster. The merging process runs recursively until all the objects are grouped in a single cluster or until one or more termination conditions are satisfied. The single cluster on top of the hierarchy becomes the root of the clusters tree.

A divisive clustering analysis (DIANA) applies a top-down approach that initially constructs the largest single cluster containing all the objects. This single cluster is the root of the hierarchy. Recursively, DIANA divides the cluster into smaller and smaller
sub-clusters until a cluster consists a single object or the objects within a cluster are relatively close to each other.

The output of either AGNES or DIANA is a tree of clusters or known as a dendrogram. A dendrogram has two axes: the x-axis represents the objects/cluster of the tree and the y-axis represents the height or linkage distance between clusters. Either AGNES or DIANA uses a distance metric (e.g., Euclidean or Manhattan) to perform the merging or splitting of the clusters. In the dendrogram, a cluster is observable via the horizontal line connecting its members. Visually, the dendrogram also organises clusters based on their linkage measures which are seen from the height separating the clusters. A linkage measure is a distance that separates one cluster to another. There are four known linkage measures [86] in the following equations.

Minimum distance: \( pdist_{\text{min}}(C_i, C_j) = \min_{a \in C_i, b \in C_j} \{|a - b|\} \) \hspace{1cm} (2.7)

Maximum distance: \( pdist_{\text{max}}(C_i, C_j) = \max_{a \in C_i, b \in C_j} \{|a - b|\} \) \hspace{1cm} (2.8)

Mean distance: \( pdist_{\text{mean}}(C_i, C_j) = |\mu_i - \mu_j| \) \hspace{1cm} (2.9)

Average distance: \( pdist_{\text{avg}}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{a \in C_i, b \in C_j} |a - b| \) \hspace{1cm} (2.10)

The equation \(|a - b|\) calculates the distance between two points or clusters \(i\) and \(j\). Variables \(\mu_i\) and \(\mu_j\) denote the means in clusters/points \(i\) and \(j\), respectively. And, \(n_i\) and \(n_j\) give the number of objects in clusters/points \(i\) and \(j\), respectively.

The use of minimum or maximum linkage are the two extremes to measure the distance between cluster. In general, if the objects are compact and approximately coherent, using the maximum linkage measure will produce well-separated clusters. Otherwise, the minimum linkage may be suitable. However, both linkage measures are sensitive to noisy data or the observations containing outliers. Therefore, the \(pdist_{\text{mean}}\) or \(pdist_{\text{avg}}\) compromise such sensitivity to noise or outliers. The mean linkage is the simplest to compute but only best for numeric or ordinal data. Meanwhile, the average measure is more robust that it can handle nominal and numeric data.

With a dendrogram, one can visually observe it to determine the best number of clusters. However, if the size of the tree is overly large, then, its visual observation becomes not feasible. In that case, we can compute the \(k\)-lifetime to search the optimal number of clusters. This \(k\)-lifetime is obtained based on the farthest distance between clusters excluding the single objects distances.
2.8.2.2 Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)

Previously, the AGNES and DIANA are the simplest approaches of the hierarchical clustering. However, both clustering methods are incapable to undoing the processes done previously, and the time constraint becomes an issue when the size of the data is very large. To solve such problems, BIRCH is introduced that it can handle a very large numeric data and it allows two-way processes.

BIRCH is a clustering method that combines hierarchical clustering at the early micro-clustering procedure and other clustering methods, such as partitioning clustering at the later stage (macro-clustering). It uses specific information representing the clustering feature (CF) and clustering feature tree (CFTREE). In such a way, the clustering method runs quite fast and scalable in large or streaming data, and the clustering is not fixed which means that we can generate the cluster incrementally and dynamically, given the incoming data [86].

Assume we have a cluster consists of \( n \) \( d \)-dimensional data. \( CF \) is an information containing the summary of the cluster which include: \( n \) denotes the number of points in the cluster, \( LS \) gives the linear sum of the \( n \) points: \( \sum_{i=1}^{n} p_i \) and \( SS \) be the square sum of the cluster members: \( \sum_{i=1}^{n} p_i^2 \).

\[
CF = \{n, LS, SS\}
\] (2.11)

As an essential part of BIRCH, \( CF \) is useful to derive important statistics of a cluster, i.e. the centroid (\( p_0 \)), cluster radius (\( R \): the average distance from member points to the centroid) and diameter (\( D \): average pairwise distance within a cluster).

\[
p_0 = \frac{LS}{n}
\] (2.12)

\[
R = \sqrt{\frac{(nSS) - (2LS^2) + (nLS)}{n^2}}
\] (2.13)

\[
D = \sqrt{\frac{(2nSS) - (2LS^2)}{n(n - 1)}}
\] (2.14)

With such summary using \( CF \) can avoid using too much space for storing the details of the points/objects in a cluster. Because it only requires a static size of memory for the \( CF \) information. In addition, if there are two separate clusters, \( C_1 \) and \( C_2 \), then, the merging of the two is done by adding \( CF_1 \) and \( CF_2 \) to create a new cluster \( C_3 \), see
\[ CF_3 = CF_1 + CF_2 = \{(n_1 + n_2), (LS_1 + LS_2), (SS_1 + SS_2)\} \tag{2.15} \]

Next, the CFTREE stores the CF vectors of the clusters structured as a tree with balanced heights. A non-leaf node ("parent") has sub-nodes or "children" where the parent node stores the sum of its children’s CF vectors. A CFTREE has three information to keep: 1) branching factor \((B)\) which limits the number of children for each non-leaf node and 2) threshold \((T)\) which limits the diameter of subclusters stored in the leaf nodes.

Additionally, BIRCH applies a multi-stage clustering technique that in the first stage it scans the data to construct the initial CFTREE which can be seen as a multilevel compression of the data in which clustering structure of the data is preserved. This is the stage where the CFTREE is incrementally built as new objects are inserted. If a subcluster’s diameter exceeds the threshold value, then BIRCH partitions the leaf node and probably other nodes. Once a new object has been added, the corresponding information propagates towards the root of the hierarchy. The threshold value in the CFTREE can be adjusted based on the available memory, then, the CFTREE is rebuilt accordingly. Subsequently, in the next stage, BIRCH applies a clustering algorithm to partition the leaf nodes of the CFTREE. At this stage, sparse clusters are removed (i.e. outliers) and compact clusters are merged into a larger one.

Yet, a consideration when using BIRCH is that one must assume each cluster has a spherical shape in which the notion of \(R\) and \(D\) defines its boundary. More importantly, due to the fact that there is a limited number of entries in a CFTREE, the constructed clusters may not be as natural as they are expected.

\section*{2.9 Reinforcement Learning}

\subsection*{2.9.1 Basic Theory}

Reinforcement Learning (RL) actually bridges the characteristics of supervised learning, where a training data set is trained using correct labels, and unsupervised learning where it clusters objects based on their similarities [72]. RL learns to perform different strategies and finds the one which works best [72]. It has a number of states, available actions that are executable for each state, and rewards. The idea is to provide experiences (states) to the model to learn from its actions which produce mistakes and correct
executions (produce rewards) [72, 87]. Basically, a reinforcement learning algorithm has a look up table that maps states and their value for each possible actions. Nevertheless, such an approach is not feasible in a case where the problem space is very large. Instead, an experience generalization [88, 89] is introduced as an alternative to the look-up table approach.

In the existing serious game research, RL was applied as an agent that controls the gameplay tasks sequence in Adaptive Experience Engine (AEE) [14]. A task in a serious game is the smallest component of the content tree. A player has a main goal $G$ to complete all the tasks in the serious game. Initially, this engine has tasks authored by experts and they have no particular order to present them to the player. In order to create a sequence, the agent performs an action by selecting a task from the content pool. At the end of each task completion, the AEE measures some performances of the player (e.g., the score, gaming duration, the level of interaction acted and game result). Then, it updates the states for the current sequence, such as the average score, average duration, etc. Based on the updated states, the agent decides whether the current selection of task should be rewarded or penalised. From this illustration, the objective of the RL is to find the optimum policy $\pi$ that maximises the cumulative rewards $R$. A policy specifies the best action $a \in A$ (i.e. selecting a task from a set of tasks) for every input (state $s$ of the problem space). Whenever practical, the agent stores its sequence generation details in a table consisting of state, action and value.

The question: how is this value measured?

First, the illustrated problem is a form of a delayed reinforcement learning, one often assumes that the time-separated future rewards are less valuable than immediate rewards. Hence, a temporal discount factor accommodates this principle, $0 \leq \gamma < 1$.

The immediate value of a reward, $r(s_i, a_i)$, occurring $i$ time units in the future, is taken to be $\gamma r(s_i, a_i)$. Assume we have a policy $\pi(S)$ that maps input vectors (i.e. a set of states) into actions (i.e. a pool of tasks), and let $r(s_i, a_i)$ be the reward that will be received on the $i^{th}$ time step after one begins executing an action $a_i$ starting in state $s_i$. Then, the accumulated rewards over all time steps by policy $\pi$ beginning in state $s_0$ is formulated in the Eq. (2.16).

$$V(s_i, a_i) = \sum_{i=0}^{\infty} \gamma r(s_i, a_i) \quad (2.16)$$

One reason for using a temporal discount factor is making the above sum finite. An optimal policy is one that maximizes $V(s_i, a_i)$ for all steps $i$. 

In general, let’s consider the case wherein the rewards, \( r(s_i, a_i) \), are random variables; henceforth, the effects of actions on the observable states are random too. In Markovian principle, for example, the probability that action \( a \) in state \( s_i \) will lead to state \( s_j; j = i + 1 \) is given by a transition probability \( p[s_j|s_i, a] \). Then, we will want to maximize expected future reward and would define \( V(s_i, a_i) \) as follows:

\[
V(s_i, a_i) = E\left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, a_i) \right]
\] (2.17)

In either case, we call \( V(s_i, a_i) \) the value of action \( a_i \) for input \( s_i \).

If the action prescribed by \( \pi \) taken in state \( s_i \) produces a new state \( s_j \) (randomly according to the transition probabilities), then we can write \( V(s_i, a_i) \) in terms of \( V(s_j, a_i) \), \( j > i \) as follows:

\[
V(s_i, a_i) = r(s_i, a_i) + \gamma \sum_{s_j} p[s_j|s_i, \pi] V(s_j, a_i)
\] (2.18)

where \( \gamma \) be the discount factor, \( V(s_i, a_i) \) gives the value of state-action pair under policy \( \pi \), \( r(s_i, a_i) \) is the expected immediate reward received when we execute the action prescribed by \( \pi \) in state \( s_i \) and \( p[s_j|s_i, a] \) is the probability that the environment transitions to state \( s_j \) when we execute the action \( a_i \) prescribed by \( \pi \) in state \( s_i \). Hence, for the optimal policy, \( \pi^* \), see Eq. (2.19) for the “optimality equation” chooses the best action.

\[
V(s_i, a^*) = \max_a [r(s_i, a) + \gamma \sum_{s_j} p[s_j|s_i, a] V(s_j, a)]
\] (2.19)

Bellman’s dynamic programming (DP) theory [90] assures us that there is one optimal policy at the minimum, \( \pi^* \), that satisfies this equation. DP also provides methods for calculating \( V(s_i, a^*) \) and at least one \( \pi^* \), assuming that we know the average rewards and the transition probabilities. If we knew the transition probabilities, the average rewards, and \( V(s, a^*) \) for all \( s \) and \( a \), then it would be easy to implement an optimal policy. We would simply select that \( a \) that maximizes \( r(s, a) + \gamma \sum_{s_j} p[s_j|s_i, a] V(s_j, a^*) \).

2.9.2 Q-Learning

It is difficult for an agent to learn an optimal policy \( \pi^* \) directly, due to the insufficient training samples of the form \((s, a)\). Instead, the available training data available to the agent is the sequence of immediate rewards \( r(s_i, a_i) \). Hence, the RL learns the values \( V(s, a) \) to find the optimal policy only when the agent has perfect knowledge of the transition probabilities function \((p[s_j|s_i, a])\) and the immediate reward function \( r \).
many practices, it is impossible for the agent or even the human programmer to predict the exact outcome prior to performing an arbitrary action to an arbitrary state [73]. Therefore, Watkins [91] proposed an RL that learns the $Q$ value instead of the value of the policy, or known as $Q$-Learning. He defined the evaluation function $Q(s, a)$ that maximises the discounted cumulative rewards, in that it can be achieved from the state $s$ and performing action $a$ as the first action. Then, the agent learns via sequences of episodes. The Algorithm 2 explains how Q-Learning learns the estimated Q-value ($\hat{Q}$) episodically. The update of Q-value in each step is provided by Eq. 2.20. In AEE, the same problem exist with respect to the insufficient training samples due to the online sequence generation. Therefore, AEE’s agent utilised Q-Learning with Linear function approximation [14].

$$\hat{Q}(s_i, a_i) \leftarrow r + \gamma \max_{a_{i+1}} \hat{Q}(s_{i+1}, a_{i+1}) \quad (2.20)$$

Note that the algorithm uses the current $\hat{Q}$ for the new state $s_{i+1}$ to estimate the $\hat{Q}(s_i, a_i)$ for the previous state $s_i$. It performs the action in the environment and observes the resulting state $s_{i+1}$ and reward $r$. Thus, we can view this as sampling these functions at the current values of $s$ and $a$ [73].

**Algorithm 2** Q Learning Algorithm [72]

```plaintext
function INITIALIZATION
    set $Q(S, a)$ to random values for all $s$ and $a$
end function

repeat
    initialise $S$
    repeat
        select action $a$ using greedy method or other policy
        take action $a$ and receive reward $r$
        sample new state $s_{i+1}$
        update $\hat{Q}(s, a)$ using Eq. 2.20
        set $s_i = s_{i+1}$
    until all steps in the current episode
until no more episode
```

Using the current $\hat{Q}$ values, $\hat{Q}(s, a)$, the agent always selects an action that maximizes it. Note that only the $\hat{Q}$ value corresponding to the state just exited and the action just taken is adjusted. And that $Q$ value is adjusted so that it is closer to the sum of the immediate reward plus the discounted maximum (over all actions) of the $\hat{Q}$ values of the state just entered.
Let us consider the existing approach in AEE’s agent sequencing the tasks in a serious game [14] if it applied a table-based approach. Suppose, the current task selected is task\(_0\). The maximum \(\hat{Q}\) value occurs for \(a = 1\), so the next task to choose is the much-simpler to perform (e.g., task\(_3\))—receiving no immediate reward. The maximum \(\hat{Q}\) for task\(_3\) is 5, and the learning mechanism attempts to make the value of \(\hat{Q}(\text{task}_0, 1)\) closer to the discounted value of 5 plus the immediate reward (which was 0 in this case). In this episode, there will be no changes in the table. The reader might try this learning procedure in the tasks pool with a simple computer program. Notice that an optimal policy might not be discovered if some tasks are not selected nor some actions not tried frequently enough.

To solve such an issue in table-based Q-Learning, we can apply a function approximator (e.g., a linear function or neural network) to replace the look-up table. Often, problems are difficult to model as a linear function. Thus, a neural network acts as the function approximator. The neural network (NN) accepts a state and an action and outputs the Q-value of the state-action pair. A neural network consists of input and output layers. Often, there are some hidden layers between the input-output layers. Nodes between adjacent layers are connected to each other. Each connection typically has a weight \(\theta\) that adjusts as learning proceeds. Q-Learning with NN does not update the Q-values in a table, instead, it iteratively updates these \(\theta\) of the neural network to produce better estimates of state-action values. The formula to estimate Q-value of the current state-action is:

\[
\hat{Q}(s_i, a_i) \leftarrow r_{i+1} + \gamma \max_{a_{i+1}} Q(s_{i+1}, a_{i+1})
\]  

(2.21)

The notations used are the same as in the Eq. 2.20. Given a transition \(<s_i, a_i, r_i, s_{i+1}>\), the update of Q-values follows these procedures:

1. Estimate all Q-values for every action given by the current state \(s\).

2. Do the same thing for the next state \(s_{i+1}\) and compute the maximum Q-value over the network \(\max_{a_{i+1}} Q(s_{i+1}, a_{i+1})\).

3. Update Q-value for action \(a\) to \(r_{i+1} + \gamma \max_{a_{i+1}} Q(s_{i+1}, a_{i+1})\) (max value from step 2). For the remaining actions, update the Q-value using the same calculation in step 1 to ensure there are no errors of the outputs.

4. Update the weights of all connections using backpropagation method.
Once Q-values are updated, the Q-Learning selects the action of the highest Q-value in the next iteration to exploit the best decision it produces. Otherwise, some portions of the overall iterations do not exploit this highest Q-value; instead, it explores less frequent actions to reinforce its prediction.

The learning problem faced by the agent is to associate specific actions with specific input patterns. Q-learning gradually reinforces those actions that contribute to positive rewards by increasing the associated Q-values. Typically, as in this example, rewards occur somewhat after the actions that lead to them, hence, the phrase delayed-reinforcement learning. One can imagine that better and better approximations to the optimal Q-values gradually propagate back from states producing rewards toward all of the other states that the agent frequently visits. With random Q-values to begin, the agent’s actions amount to a random walk through its space of states. Only when this random walk happens to stumble into rewarding states does Q-learning begin to produce Q-values that are useful, and, even then, the Q-values have to work their way outward from these rewarding states.

### 2.10 Resampling Methods for Imbalanced Dataset

Real life dataset rarely provides a balanced distribution between classes; especially, a survey of human participants. For instance, surveying on players from various gaming skills can be difficult to obtain [92]. This can be caused by the sparse preferences of game genre and the game under the case study could be of something new for participants. Therefore, Buckley administered the second data collection to balance the distribution of skills [76]. Similarly, serious games often produced a new gameplay for the participants. To some extent, this may affect their behaviour within the game or even the self-reporting experience due to a lacking knowledge about how to play the game. In order to avoid such unreliable game data, a practice session or, at least, a tutorial is strongly recommended to provide participants with the minimum level of knowledge [1][92]. Otherwise, the metric that measures the performance of the classification often affected by this imbalance, especially if the imbalance is significant.

During cross-validation of the classification model construction, we are typically want to see how well the prediction and how optimal the model generalise. Hence, the accuracy metric shows a disproportionate classification biased towards the majority class. To deal with this problem, a pre-processing of the data is required to ensure the training dataset has a relatively balanced distribution of classes. The following subsections
elaborate known re-sampling techniques in order to balance the dataset.

2.10.1 Undersampling

Undersampling is one of the most popular and simplest methods to balance the training dataset. Undersampling retains minority samples and removing the majority samples randomly until it has equal distribution. However, the more imbalanced dataset, the more samples from the dominant class, which potentially entail important information, will be discarded. Hence, the classifier is lacking representation of the actual population.

To overcome such issue, Zhang and Mani [93] proposed four controlled undersampling methods instead of the random undersampling. Instead of removal of the majority samples, the controlled undersampling selects a subset of the dominating samples based on the distances between some minority and majority samples. Four undersampling introduced in the article are NearMiss1, NearMiss2, NearMiss3 and Distant1. In NearMiss1, the selection goes to the dominating samples in which mean distances are the nearest to the three closest minority samples. In NearMiss2, the algorithm selects the majority samples in which average distances are the closest to the farthest three minority samples. In NearMiss3, for each minority sample, it selects $n$ nearest majority samples. The value of $n$ is determined by the desired ratio between minority and majority proportion we want to achieve. However, NearMiss3 is unclear whether the selection of the majority samples allows or disallows duplications. Finally, the Distant1 method selects majority samples whose average distances are the farthest to the three closest minority samples.

2.10.2 Oversampling

Basically, the oversampling method keeps the majority samples and duplicates minority samples until a balanced dataset reached. Oversampling method ensures there is no important information have lost during the process. However, the disadvantage is the increasing size of the training dataset. Hence, the learning stage requires more time for the construction of the prediction model and a higher memory consumption. In this case, we are restricted with the available computation power and memory when applying oversampling.

The simplest method in oversampling is the random oversampling. Under this
method, we need to provide a new container that will keep the training dataset. Initially, it contains the samples from the dominant class. Then, we randomly sample with replacement members of the minority class to be added to the training dataset. This sampling process repeats until the desired ratio of class distributions reached. Yet, such a ratio is a problem to solve. It depends on the initial proportion between the classes. Say, we want to achieve 50:50 proportion between binary classes. If, the initial proportion was a significant imbalance, such as 5:95, then, the result of oversampling contains too many duplicates which can bias the prediction.

### 2.10.3 Synthetic Minority Oversampling technique (SMOTE)

To overcome such problems exist in undersampling and oversampling, a resampling method called Synthetic Minority Over-sampling Technique (SMOTE) [94] creates synthetic samples based on the existing minority class samples. We over-sample minority samples by taking each member and introducing new synthetic samples between the line that connects $k$ number of the minority class nearest neighbours. The selection of neighbours from the $k$ nearest neighbours is random based on the desired number of over-sampling.

Based on the paper, we use five nearest neighbours. For instance, if we need to increase the minority samples up to 200%, we only select two neighbours from the five nearest neighbours ($k = 5$). Then, from the direction of each selected neighbours, we generate one synthetic sample. The generation of synthetic samples follows these procedures:

1. Measure the difference ($D_f$) between the feature vector (sample) under consideration and its nearest neighbour,

2. Multiply $D_f$ by a random number between 0 and 1, we denote this as $MD_f$,

3. Add $MD_f$ to the feature vector under consideration which generates a random point on the connecting line between two specific features.

Such approach effectively forces the generalization of the decision region of the minority class. We present the more detailed algorithm for SMOTE in Algorithm 3.
Algorithm 3 SMOTE Algorithm [94]

**Purpose:** Generating synthetic samples from the existing minority samples.

**Input:** Total samples from the minority class (s); The desired increase of minority samples %n; k number of nearest neighbours.

**Output:** (n/100) * S new synthetic samples.

1. If \( n < 100 \) then
   a. Randomize S:
      \[ S = N * S / 100, \text{ where } N = 100; \]
2. \( n = (\text{int})(n/100); \)
3. \( tot_f = \text{total features}; \)
4. \( S[i][]: \text{the array of minority samples}; \)
5. \( idS: \text{the index of generated a synthetic sample (starts from 0)}; \)
6. \( Sy[i][]: \text{the array of the synthetic samples}; \)
7. \( \text{for } i = 0 \text{ to } s - 1 \text{ do} \)
   a. Compute k nearest neighbours and keep the indices as NNS;
   b. Call function Resample\((n,i,NNS)\)
8. \( \text{end for} \)
9. \( \text{function Resample}(n,i,NNS) \)
   a. \( \text{while } n \neq 0 \text{ do} \)
      i. \( \text{rand} = \text{Generate a random number between 1 and k to choose the an index within the k nearest neighbours}; \)
      ii. \( \text{for } f = 0 \text{ to } tot_f - 1 \text{ do} \)
          a. Compute \( Df = S[\text{NNS[rand]}][f] - S[i][f]; \)
          b. Compute \( f_{gap} = \text{random number between 0 and 1}; \)
          c. \( MDf = S[i][f] + (f_{gap} * Df); \)
          d. \( Sy[idS][f] = MDf; \)
          e. \( idS++; \)
          f. \( n--; \)
      ii. \( \text{end for} \)
   b. \( \text{end while} \)
10. \( \text{end function} \)
Table 2.1: Confusion Matrix for a Binary Class Problem.

<table>
<thead>
<tr>
<th></th>
<th>Predicted Negative</th>
<th>Predicted Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Negative</td>
<td>True Negative</td>
<td>False Positive</td>
</tr>
<tr>
<td>Actual Positive</td>
<td>False Negative</td>
<td>True Positive</td>
</tr>
</tbody>
</table>

2.11 Prediction Performance Validation

Typically, a confusion matrix evaluates the performance of a machine learning algorithm. For a binary class problem, Table 2.1 depicts the confusion matrix. The columns are assigned for the Predicted class and the rows are the Actual class. This confusion matrix is a common evaluation matrix for supervised machine learning. So, we need to provide training samples and test samples when running a supervised machine learning. The Actual class values are obtained from the associated labels of the samples, while the Predicted class values are shown by the outputs of the machine learning algorithm. True Negative (TN) is the total samples in which actual labels and the predicted outputs are all negative. True Positive (TP) is the number of samples whose actual labels and predicted outputs are all positive. False Negative (FN) counts the number of samples with positive labels but incorrectly classified as negative. And, False Positive (FP) counts the samples labelled with negative value but incorrectly classified as positive.

The most common evaluation matrix to measure the performance is called \[ \text{Accuracy} = \frac{(TN + TP)}{(TN + FN + TP + FP)} \]. Once a machine learning algorithm has been trained, we can test its prediction using samples which have not been seen by the algorithm. Nevertheless, this is not a good option because it may be the case of an over-fitting of the model to the training dataset. It means that the model might be able to classify the samples in the training dataset very well; yet, it may not generalise well.

Hence, a reliable technique to measure the performance of the algorithm is by splitting the dataset into training and validation sets. We have to ensure that the validation set is sufficient, either from each class or from an acceptable range of values in a classification or a regression problem, respectively. Otherwise, an imbalanced validation set can negatively impact for the accuracy measure. For instance, the validation set contains 10 negative samples and 290 positive samples will not really show the actual accuracy for the negative class. Preferably, a cross validation (CV) iteratively measure the accuracy by using different training-validation sets. The most common approach is the \( n \)-fold CV. Initially, it partitions the dataset into \( n \) parts. Then, it generates the partitioned dataset into \( n \) duplicates. In each duplicate, the machine learning algorithm
is trained using samples from partition 1 to \( n - 1 \) of the dataset and validated with the
samples from partition \( n \). In the following iteration, the machine learning algorithm
uses the next duplicate of the partitioned dataset. Here, the validation set is shifted to
partition \( n - 1 \) and the samples from the other partitions become the training set. Such
process is iterating until the last duplicate of the dataset is used in which the first par-
tition becomes the validation set and the remaining partitions become the training set.
Finally, the accuracy metrics from these iterations are then summarised by averaging
them.

Unfortunately, using only one evaluation metric (i.e. accuracy) is not sufficient for
our case especially when the dataset is imbalance. Therefore, we also use additional
evaluation metrics which are less sensitive to imbalance dataset. The following equa-
tions of evaluation metrics are based on the confusion matrix of a binary classification
problem.

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{2.22}
\]

\[
\text{Recall} = \frac{TP}{TP + FN} \tag{2.23}
\]

\[
\text{Specificity} = \frac{TN}{FP + TN} \tag{2.24}
\]

\[
FPR = \frac{FP}{FP + TN} \tag{2.25}
\]

\[
F_score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.26}
\]

We also measure the area under receiver operating characteristic (AUC). This requires
the receiver operating characteristic (ROC) curve to be drawn first. ROC is a plot
that describes the performance of a binary classifier as its discrimination threshold is
varied. It is a 2-dimensional plot wherein x-axis is the false positive rate (FPR) and
the y-axis is the true positive rate (TPR or Recall). Each point in the plot is an instance
of the confusion matrix. A diagonal line from bottom left to top right of the plot
discriminates the ROC space. Instances above the diagonal line are good classification
result and points below the diagonal line are bad classification result. For instance, an
ideal point (instance) on the ROC curve would be (0,100); that is all positive samples
are classified correctly and no false positive. From those points, the ROC curve is
drawn and the area under the curve can be measured. In the normalised units, the
area under the curve (AUC) is the same as the probability that a classifier will rank
a randomly chosen positive sample higher than a randomly chosen negative sample.
(assumption: positive ranks higher than negative) [95].

On the other hand, unsupervised machine learning algorithms, especially clustering, validate the performance internally or externally. Internal validation techniques ensure the generated clusters from the perspectives of compactness, connectivity or separation [96]. Compactness metric evaluates how close objects inside a cluster which are usually measured using the variation of their distances. The lower the variation of within-cluster objects, the more compact the cluster (i.e. a better clustering). Connectivity metric measures how connected objects within a cluster according to their nearest neighbours. The value of connectivity is ranged from 0 to unlimited, but it must be minimised to indicate a good clustering. Separation metric determines how well the clusters are separated which can be measured from the distances between the centres of clusters or the pairwise minimum gap between objects from different clusters.

Meanwhile, external validations for clustering analysis include silhouette and cophenet correlation coefficient. Silhouette validation is formed as a plot that displays the distance between one object in one cluster to other objects from the neighbouring clusters. In this way, we can assess the number of clusters visually. The range of values of this silhouette plot starts from -1 to +1. The closer a point in the plot to -1 means that the point is located in between neighbouring clusters (i.e. a bad point for clustering). On the other hand, the closer a point to the +1 value it means the point is far away from neighbouring clusters (i.e. a good clustering point). Meanwhile, a cophenet correlation coefficient is useful for hierarchical clustering. It measures how well the dendrogram (i.e. clusters tree) keeps the pairwise distances between the original data points. The cophenetic distance between two points is represented by the height in the dendrogram. This height is constructed based on the linkage (i.e. average, centroid, complete, single, weighted, median or ward) firstly used to join the two objects. Equation 2.27 computes the cophenetic correlation coefficient \( c \). \( Y_{ab} \) denotes the distance between object \( a \) and \( b \). \( Z_{ab} \) denotes the cophenetic distance between object \( a \) and \( b \) specified by a linkage between both objects. The range of values of \( c \) is 0 to 1 representing the worst and the best clustering, respectively.

\[
c = \frac{\sum_{a < b} (Y_{ab} - \mu_y)(Z_{ab} - \mu_z)}{\sqrt{\sum_{a < b} (Y_{ab} - \mu_y)^2 \sum_{a < b} (Z_{ab} - \mu_z)^2}}
\] (2.27)
Chapter 3

Serious Educational Game Development Framework

In this chapter, we will elaborate our framework for developing serious educational game (SEG) which taking benefits of the different content spaces. We begin with the motivations behind the development frameworks for SEG as well as their underlying issues. Then, we elaborate our approach descriptively followed by the test case to prove the framework is applicable in actual development. Finally, we summarise the user feedback when playing the test-case SEG.

3.1 Motivations

In Chapter 2 we have elaborated some of the approaches to developing serious games that the cost and time consumption were the apparent issues. Alternatively, we are looking at a perspective that SEG has mainly two types of content namely the education materials and game elements. Our development framework in this chapter is to address the mapping issue by embedding annotated knowledge chunks into categorised game content/elements seamlessly during SEG development. With today’s networking technology, we can obtain the resources for learning of a knowledge (e.g., syllabus or handout). Such a resource organises units of knowledge and the sufficient learning instruction [10]. Our framework would capitalise on such information so that game developers can efficiently annotate the units of knowledge or an information retrieval techniques automatically acquires the information and annotate them into the knowledge units accordingly. Meanwhile, on the perspective of game elements, techniques
in [97, 98] divert the purpose of an existing commercial game for an education purpose. This approach exploits the properties of existing commercial games which are in common with learning, e.g., in order to play a game, a player has to learn game rules, objectives and strategy to success unconsciously, which is also required for learning in traditional education systems [97]. As an alternative, procedural content generation in entertainment game can generate game content automatically via algorithms, which requires far fewer expenses in the development stage. Moreover, the latest Procedural Content Generation (PCG) work [28] recommends that an appropriate use of the categorised game content can engage the players in a positive gaming experience. Based on these motivations, our framework would encourage the utilisation of PCG from existing entertainment games in developing the SEG. We argue that the mapping between two content spaces may better be done by embedding annotated knowledge units into categorised game content/elements.

Such a structure of serious game content containing knowledge and game content spaces enables a more controllable serious game generation. Yet, the existing SEG approaches instruct education experts to handle the process of deploying learning materials into a SEG. As a consequence, an expert is expected to deeply understand characteristics of both content spaces with regards to the potential linking properties between both spaces. Nevertheless, it becomes infeasible and not scalable in the presence of complex yet very large size of either content space. Alternatively, game developers can utilise the natural and inherent properties from each content space to address the mapping issue. Acquiring such properties are feasible since they are measurable given that each content space has attributes. Furthermore, the PCG techniques allow for a flexible control of game elements to embed knowledge chunks. Thus, we suppose that taking a maximum advantage of learning resources and making use of the latest PCG techniques could significantly lower the cost of SEG development. With the fact that both content spaces have detailed and appropriate descriptions, the developer can formulate different aspects between them, which serve a proper deployment.

### 3.2 Proposed Methodology

To address the aforementioned issues, we introduce an alternative SEG development framework (see Fig. 3.1). Initially, learning materials are separated from game elements. Regarding the learning materials, we need to describe the education materials in the annotation step. A reliable resource, such as a syllabus, provides the relevant
values of the learning materials’ attributes. Then, from the same resource or based on the specified characteristics, we can establish the strategy for delivering them in the SEG. Meanwhile, in the counterpart, we categorise the game content space in a couple of steps. The first categorization groups game content based on the level of difficulty. Subsequently, within each difficulty level, a clustering analysis groups game content based on the similarity. Consequently, the aspects underlying the descriptive learning materials and game elements can advise a developer applying their logic in formulating the mapping procedures between learning materials and game content. The outcome is an SEG content module comprised of playable game stages for learning.

The following subsections elaborate the proposed framework in more details.

### 3.2.1 Education Material Space

**Education material space** of an SEG refers to all the relevant units of knowledge to be learned by a player. In a case where there is no organization, Belloti’s [99] annotation method for tasks in a serious game has inspired us in structuring the SEG content. Yet, employing experts to annotate subjective attributes is no longer feasible to handle the growing size of learning materials. Especially for serious games for recalling abundant units of knowledge, such as astronomy, language vocabulary and geographical items.

We argue that the ideal properties for learning materials are originating from its inherent description provided by a reliable education resource (e.g., syllabus [100]) and the representation of the knowledge (e.g., text, image, audio or video). Hence, **annotation** operates in the natural descriptions of the learning materials with a minimum involvement from experts. Specifically in Algorithm 4, the knowledge resource (e.g. learning handout) provides a sufficient information regarding the unit of knowledge to be delivered for the learners. Then, based on the information retrieved from the handout, for instance, we can breakdown the education materials as small units. The smaller units of knowledge allow the learning task much simpler especially for a recalling purpose.

Subsequently, we must select the relevant properties based on the recalling purpose and/or their representation within the game (sourced from the same learning handout). These properties make one unit of knowledge distinct from the others. And they are descriptively identifiable as well as measurable that later helps the mapping more effectively. Given the available documented resources, an information retrieval technique –beyond our scope –automates the annotation process. In addition, a computer program that measures specifications of a content representation (e.g. text, image, audio
Algorithm 4 Knowledge Categorisation

1: **Purpose:** Annotation of Knowledge Space.
2: **Input:**
   - Raw Education Materials;
   - The knowledge resources or handout, e.g. a syllabus;
3: **Output:** Annotated Education Materials;
4: Get the reliable resource of the Education Materials;
5: Breakdown the knowledge space into the smallest unit \( (E) \) possibly delivered in a game session;
6: Add the properties of the education materials as the input vectors \( (I) \) based on the knowledge’s resource and their representations;
7: Use an information retrieval program to annotate each education material with the corresponding knowledge-related information from the resources;
8: Annotate the remaining properties of each education material based on the knowledge representation;
9: **for all** \( E \) **do**
10: **if** \( E \) is Related to Other Materials **then**
11: Search Prerequisites and/or Correlated education materials based on the learning handout;
12: **else**
13: Update the organisation of \( E \) based on the annotated properties;
14: **end if**
15: **end for**
3.2. PROPOSED METHODOLOGY

Figure 3.1: SEG development framework (reproduced from [1]).

or video) also annotates the attributes automatically. Such as the number of words of the text-based learning material or the length of an educational video.

Altogether, the education content space provides a comprehensive detail for initiating the strategy for delivering the learning materials. To accommodate that, we must organise them based on the delivery strategy (a term used in [14]). If no relationship exists (e.g. prerequisites of learning words are recognising the letter and their conjunctions) between education materials, an automated method (e.g. sorting) establishes the strategy based on the attribute values. Otherwise, a syllabus or a teaching handout can show the strategy explicitly. Consequently, with an established strategy, players is expected to recall the knowledge appropriately.

3.2.2 Game content space

Game content space of an SEG consists of all the playable content generated by an entertainment game engine, in our case: procedural content generation (PCG), to facilitate the learning defined via knowledge space for a player. PCG provides details of the game content in the parameters and it often produces a large space of game content. As a consequence, manually identifying the category for the content space is not feasible. Inspired by an entertainment game procedural content generation [28], we categorise the game content space into difficulty levels and similarity groups. Difficulty categorisation accommodates players with different gaming abilities [36]. Meanwhile, similarity categorisation benefits the plentiful game content choices to support repetitive sessions of learning.

Robert suggests [28] that the use of categorised game content can engage players in a positive affective experience via a proof-of-concept first-person shooter game. The content space has been categorised via difficulty classes learned from game examples annotated by developers [28]. For specifying difficulty levels, we can also
Algorithm 5 Rule-based Difficulty Categorisation

1: **Purpose:** Categorising game content difficulty level via manual observation.
2: **Input:**
   - Game content data set, $G$, which contains input vectors;
   - The set of attributes, $I$, which forms the game content’s input vector;
   - Total Difficulty categories, $c$;
3: **Output:** Rules table for categorising game content space based on difficulty levels;
4: for all $g \in G$ do
5:   GENERATE $g$ based on the parameters of the game content;
6:   PLAY and OBSERVE game $g$ with respect to its content feature values;
7:   ASSIGN a DIFFICULTY level;
8: end for
9: OBSERVE the impact of each feature of the game content to the DIFFICULTY level;
10: CREATE a new LIST of game content features ($I'$) and sort them descendingly based on the difficulty impact;
11: CREATE a vector $ClassFlag$ (size: $c$) as flags for each difficulty category;
12: SET the vector values of $ClassFlag$ to -1 indicating NO RULES associating with each difficulty category;
13: for all $i \in I'$ do
14:   for all $j \in c$ do
15:     if $ClassFlag[j] < 0$ then
16:       DECIDE the THRESHOLD value and the CONDITION that categorise $G$ into difficulty level $j$;
17:       if The rule FITS the difficulty level? then
18:         SET +1 in $ClassFlag[j]$ indicating the rule is set for difficulty level $j$;
19:       end if
20:     end if
21:   end for
22: end for
3.2. PROPOSED METHODOLOGY

adopt a rule-based approach (see Algorithm 5 for details). We propose this approach by considering the low-dimensional independent features of the game content space. In that, the developer can observe the generated games and s/he then manually creates the rules which categorise the game content appropriately. Technically speaking, we have to recognise the effect of the PCG’s controlling parameters to the difficulty level generated by playing and observing various games. In that, we should iterate to some games, observe and annotate each of them a difficulty level. Normally, three or five difficulty options \( (c) \) are sufficient. However, fewer categories should be more straightforward for classification development. Then, observe the effect of each input feature of the game to the annotated difficulty. Bare in mind to identify the difficulty-defining feature. Once we are confident in our observation, make an ordered list \( (I') \) of these affecting features. And, create as many as \( c \) flags \( \text{ClassFlag} \) with -1 indicating that the classes have no categorisation rules. Subsequently, iterate through the input features \( I' \) with -1 flag of vector \( \text{ClassFlag} \). Set the threshold values and condition of the feature under consideration corresponding to each difficulty level. If the current rule (i.e. threshold and condition) fits a difficulty category, keep it in the rules table and set the \( \text{ClassFlag} \) of the input feature to +1. Otherwise, iterate to next input feature until values in \( \text{ClassFlag} \) are +1.

Alternatively, if the parameters have relatively equal difficulty impact values, an active learning technique is applicable by allowing the developer as the oracle. Roberts [28] provides a thorough elaboration of the technical implementation of an active learning for the difficulty categorisation. Consequently, content categorisation naturally takes place with the specified difficulty levels.

Compared to the aim in [28], a different purpose of clustering analysis occurs here. Given the value of \( k \) as a total number of chunks of knowledge, the analysis identifies \( k \) groups of similar game content within each difficulty level. The goal is to provide a unique group of game content for each education material in each difficulty category. Henceforth, they prevent boredom developing when multiple repetitions of a game session are required. Given the total \( k \) clusters to form, a k-mean, k-medoid or hierarchical clustering techniques should fit if the game content space is not more than a medium size. If the size is large, we need a reliable and faster clustering technique such as Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH). Algorithm 6 provides the details of the procedure to accomplish this purpose.
Commonly, BIRCH has some tuning parameters that need initialization before generating the clusters. The cluster radius threshold ($r$) is a parameter to define the maximum radius that forms a cluster. Different values of $r$ can vary the quality of the clusters; therefore, we need to prepare a vector of $r$ values. Next, we also initialise the number of clusters ($k$) to be built by BIRCH. We must set the $k$ value the same as the total education materials in the SEG. The purpose is to prepare the candidates for the one-to-one mapping between a cluster in a difficulty category with an education material. Likewise, we also initialise the BIRCH’s branching factor ($b$) parameter that later describes the tree construction. For simplicity, we prefer the binary tree of the clusters by setting the $b = 2$. Then, the last initialisation is a variable to store the maximum mean silhouette score ($s = 0$). Commonly, a silhouette validation visualises a cluster’s consistency as a -1 to +1 score. The closer a point’s score to -1 indicates a bad point for the corresponding cluster. While the opposite score specifies a well-distanced point to the neighbouring clusters.

Afterwards, within each difficulty group of the game content space, the algorithm searches the constructed hierarchical clusters with the highest silhouette score ($s$). The search process iterates through all the $r$ values of the BIRCH clustering. If the current silhouette score ($s_i$) is greater than the maximum value of $s$, then, the currently built clusters have the better similarity categorised game content for the current difficulty level. Once the parameter tuning iteration finishes, the clusters with the best $s$ scores are the similarity categorised game content.

### 3.2.3 Mapping Education Materials with Game Content

**Mapping** between the aforementioned content spaces is the decisive step that assigns a unit of knowledge into game content based on their underlying properties. Often, educational game designers view knowledge subjects as the learning goals in the game. Inline with that, our framework can provide the straightforward mapping. Thus, the player has to address the learning task in a selected game mechanic, such as match-making, collection, shooting or narratives. In the developer’s viewpoint, identifying one or more game mechanics (as the platform for learning task) from the existing game content is within his/her reach. For instance, Pac Man game has collection game mechanics where dots are to be collected by the avatar. In the math version of Pac Man, i.e. Number Muncher game, dots are “replaced” by answers to a specific math question [3].

Ensuring that the selected game mechanics promotes learning to the subject of
3.2. PROPOSED METHODOLOGY

Algorithm 6 Similarity Categorisation using BIRCH algorithm

1: **Purpose:** Categorizing Game Content’s similarity in a difficulty level.
2: **Input:**
   - Difficulty Categorised Game Content Data set, $DC$, consists of $c$ difficulty categories;
   - The set of attributes, $I$, which forms the input vector;
   - Total clusters in a difficulty level $k$;
3: **Output:** A total of $k$ groups of similarity categorised game content ($SC_i$) in each difficulty group.
4: **Initialisation:**
   - Set the BIRCH parameter: a vector of cluster radius thresholds, $T > 0.00$;
   - Set the BIRCH parameter: total final clusters to built, $k = \text{total education materials}$;
   - Set the BIRCH parameter: branching factor, $b = 2$, to form binary trees;
   - Declare the maximum mean silhouette score ($s = 0$);
5: **for all** $DC$, difficulty categorised game content **do**
6:   **set the maximum mean silhouette score** ($s = 0$);
7:   **for all** $T$, **do**
8:       Run BIRCH clustering to produce a similarity categorised game content, $SC_i$;
9:       Measure the current mean silhouette score ($s_i$) of the clusters in $SC_i$;
10:      **if** $s_i > s$ **then**
11:         Use the current clusters as the similarity categorised game content in the current difficulty level, $SC_i$;
12:         **set the maximum silhouette score** with the current silhouette score, $s = s_i$;
13:      **end if**
14:   **end for**
15: **end for**
interest, it is recommended to choose the game mechanics that directly lead to the original game’s goal. In that, the learning task becomes primary in the game session. As a result, players can dedicate their learning process while playing the game. Alternatively, other existing game mechanics may also become the container of the learning task. In one hand, these game mechanic can promote knowledge acquisition. And the most important criteria: they have no contrasting perspective with the learning goal. To accommodate this, the developer must dig deeper into the game mechanics of the game content and s/he must be able to identify their importance in the game session.

Pointing to the underlying game mechanic(s), we may employ an arbitrary or sampling-based mapping between learning materials and game content as the simplest method. Yet, it may raise ineffective learning for different players. Because learning and playing in an educational game involve numerous factors [101]. For instance, random mapping has a greater possibility to deploy an uncomplicated recall materials with difficult game content. Hence, an amateur player potentially struggles to play difficult games striving to succeed the inherent challenges. Such a condition could distract a player’s main goal to recall the knowledge item. In other words, applying an arbitrary or sample-based mapping may produce imbalanced outcomes for the players.

Consequently, the mapping should serve specific conditions enabling appropriate deployment of education materials that promotes fair experiences for various players. The procedures for mapping the two content spaces are based on the measured conditions (e.g. statistics in a categorised content) between them. These conditions help the developer picture both content spaces in a high-level of perspective and how to map them appropriately. For now, we employ the developer’s knowledge to employ the in-depth properties of each content space. Inspired by the hierarchical structures of the SEG content [99], the fundamental mapping method assigns an education material as the root node of the hierarchy. Then, each unit of knowledge consists of a set of difficulty levels. Lastly, a unique cluster of game content is allocated in the leaf nodes of each difficulty level. Given the plentiful choices of game content in a unit of knowledge can prevent boredom growing when they need to repeat learning the same knowledge.

Before the mapping process starts, we need to clarify some descriptions. Let the education materials, $E$, be the learning goals. And $SC$ be the difficulty and similarity categorised game content. We limit our SEG product for the purpose of recalling the unit in $E$ space. Assume that the game which is generated via this mapping is like a pair of question and answers (i.e. correct and wrong choices). In the game, the
question is represented by the game objective to recall the correct answer. Meanwhile, the collectible or selectable game objects represent the answers. Each item in the SC space becomes a distinct challenge for accomplishing a game objective. So, the mapping question is: given a unit of knowledge, \( E_i \), to be recalled, what is the subset of game content (\( e \subset E \)) that provides the appropriate challenge? Additionally, we recommend the following specifications to perform the mapping.

1. Assume that the recalling tasks start from the simplest/easiest to the most complex ones. So, the mapping begins by ordering the member of \( E \) based on the delivery strategy stated in the corresponding resource. Alternatively, if the knowledge units are independent, we need to order them based on one or more properties of the \( E \).

2. At this point, we need to identify the measurable information that bridges the mapping between the two categorised content spaces. This is an abstract duty that a developer is capable of. Because, the memorisation is the purpose of the SEG, the quantity of a cluster may act as the retry opportunity of the potential failures which relatively correspond to the complexity of a learning task, \( e_i \). In this case, the easiest knowledge unit to recall may need the least retries, in other words, a cluster with the smallest size. In contrast, the most complex recalling task may need the largest cluster to provide more opportunities caused by frequent failures potentially occurring.

3. With such procedures, the mapping becomes straightforward ensuring that a learning material has the game content from all difficulty levels wherein one unique similarity cluster for each level. Algorithm 7 provides the pseudo-code for the mapping procedures in more details and self-explanatory.

Using this algorithm, we can deploy learning materials and game elements automatically, even when both have large spaces. Alternatively, an additional bridging information can drive a more precise mapping concerning the player’s experiences.
Algorithm 7 Mapping Algorithm.

1: **Purpose:** Mapping an Education Material with a set of Game Content.
2: **Input:**
   - Units of the Knowledge Space \((E_i \in E)\), ordered based on the strategy for delivering them in the game;
   - Similarity Categorised Game Content Data set of all difficulty categories \((c)\), denoted by \(SC_j (j \in c)\);
   - Input vector of the Education materials, \(IE\);
3: **Output:** Two additional attributes in \(SC\) containing the id of \(e\);
4: **Initialisation:**
   - Add two new attributes in all of the game content to store the education material’s id (EID) and its corresponding cluster’s statistic (CS), respectively;
5: **for all** \(DC\), difficulty categorised game content **do**
6:   **for all** \(k\) clusters in \(SC_j\), where \(j \in c\) **do**
7:   Measure the desired statistical information of the \(k^{th}\) cluster, e.g.: total game content in a cluster;
8:   Store the measured statistic in all the members of current cluster’s CS attribute;
9: **end for**
10: SORT the clusters in \(SC_j\) in an ascending order of the CS value;
11: **for all** \(k\) clusters in \(SC_j\), where \(j \in c\) **do**
12:   Assign the EID value in all the members of the \(k^{th}\) cluster with the value from the \(E_k\) (the \(k^{th}\) education material);
13: **end for**
14: **end for**
3.3 Developing a Test-case SEG for Recalling: Chem Dungeon

In this chapter, we demonstrate our proposed method to shift the purpose of an existing entertainment game into an educational game by implanting learning goals. Theoretically, our approach should enable a common developer combining separated education subjects and any types of entertainment game content. In the next subsections, we elaborate an implementation of our method inspired by an existing serious game, Chem Fight, including the solutions tackling the practical challenges.

3.3.1 Overview of an Existing SEG for Recalling Chemical Compounds: Chem Fight

Matt Palmerlee developed the Chem Fight, a turn-based educational game that confronts a single-player versus a Non-Player Character (NPC) in a chemical compound battle. In that properties of known 20 chemical elements from the Periodic Table (PT) and the compound-forming logic construct the game rules (conditional branching for game events). The Chem Fight game is open source under MIT licensing\(^1\).

Each player has a number of lives (red heart icon) representing the continuation status in the game, energy (blue flash icon) and Atom Bucks (yellow dollar sign) as a limiting tool to owning some atoms. The following paragraph explains the game mechanics with clarifications\(^2\).

The game is parted into a number of rounds until one of the players loses all their lives. Each round consists of a purchasing mode, an attacking mode by one player and the opposite player is set the defending mode to survive the attack. In a purchasing mode, each player is allowed to obtaining atoms from the periodic table at a price specified by its atomic number (e.g. Hydrogen [H] with atomic number 1 costs one Atom Bucks). On the attacking turn, one player selects a single atom as an attacking element. At this moment, only the valence electron count is visible for the opposite player (defender) and he/she is given an opportunity to figure out the atoms for countering the attack. In the following turn, the selected defending and attacking atoms are in a battle mode that the formed compound determines the result of the current

\(^1\)accessible online: http://js13kgames.com/games/chem-fight, the source code is available online: https://github.com/mpalmerlee/ChemFight, and both were accessed on 22 September 2018.
\(^2\)available online: https://github.com/mpalmerlee/ChemFight
round. If these atoms create a chemical bond with the attacking atom (we call this a successful defence), the defending player receives rewards composed of a number of Atom Bucks and Energy Units. Otherwise, the defending player receives a penalty (a decreasing energy level) for those unbounded defending elements. However, if the player has insufficient energy, their health decreases in proportion to the deficit. This game rule has a purpose of discouraging players from just defending with arbitrary or all elements they own each turn. In another case, when no compound is formed, the attack is successful. Once one game round finalised, players earn some Atom Bucks for purchasing additional elements.

3.3.2 Chem Dungeon Development Process

Chem Fight was developed from scratch by the developer. The gameplay was designated by the atom properties and the chemical compound forming logics. In essence, there was a transfer of knowledge about atoms and compound forming via gaming. Such a knowledge transfer in Chem Fight inspired us applying our proposed approach in Sect. 3.2 to develop a new serious game: Chem Dungeon. It has entirely different game mechanics compared to the original (Chem Fight) but a similar core game rule: a battle-like compound forming between atoms. As for the development purpose, we used the library of the knowledge to represent the education material space. In addition, an existing rogue-like game\(^3\) is employed to represent the game content. In the following subsections, we describe how our development framework is applied to create a Chem Dungeon from both content spaces.

3.3.2.1 Specification of Education Materials: Chemical Compounds

The end product of an educational game was intended to promote the memorization of chemical compounds for players. In our case, the education material space contained 100 compounds formed by at least two atoms. A compound is represented by alphabet characters informing the symbol, name and the bonding atoms. For instance, two Oxygen atoms construct an \(O_2\) representing the air compound. In the game, one atom appears as a game object with a text-based atomic symbol, e.g. \(K, Na, F\). Otherwise, if some atoms of the same type are involved it appears as a concatenation of strings between the total atom and the atomic symbol, e.g. \(3O, 2Cl, 2H, 6B\).

\(^3\)available from http://www.kiwijs.org/, and last accessed on 22 September 2018
According to Algorithm 4, we obtained the compounds and the atom information from the original Chem Fight game’s compound model class. Then, we verified the atoms’ information from the compound model class via The National Center for Biotechnology Information website. Once verified, we added the properties according to the forming atoms and compound representation. Attributes of the forming atoms (atom-1 and atom-2) include \textit{atom-1-number} (discrete), \textit{atom-2-number} (discrete), \textit{total-types-of-atom} (discrete) and \textit{total-atom} (discrete). Attributes associated with compound and atom representations include: \textit{total-character-symbol-1} (discrete) and \textit{total-character-symbol-2} (discrete). Subsequently, a javascript program retrieved necessary data from the original game’s class and measured the total characters of the involving atoms, then, annotated the attributes automatically. For instance, \textit{H}_2\textit{O} comprised of two Hydrogen and one Oxygen atoms. The annotated values of this compound are \textit{atom-1-number}=1 (H), \textit{atom-2-number}=8 (O), \textit{total-types-of-atom} \textit{H}_2\textit{O} is 2 (one H and one O), the \textit{total-atom} is 3 (two H + one O), \textit{total-character-symbol-1} and \textit{total-character-symbol-2} are both 1.

Subsequently, due to the compounds were uncorrelated between one another, the default strategy of memorising them was sequentially based on the annotated properties. To our knowledge, recalling them should be driven by the complexity of each compound. In other words, the more complex the representation of a compound, the more difficult the memorisation and we can put it in the later learning goal. Therefore, based on recall priority, compounds were ordered based on \textit{total-types-of-atom}, \textit{total-atom}, \textit{atom-1-number} and \textit{atom-2-number}, \textit{total-character-symbol-1} and \textit{total-character-symbol-2}, respectively. The resulting organisation set \textit{H}_2 (composed of two Hydrogen atoms) as the easiest compound to remember and the last one to recall was \textit{CaB}_6 (formed from one Calcium atom with six Boron atoms). To help us identify the sorted units of knowledge, an attribute \textit{EID} is assigned with numeric values from 1 to 100.

3.3.2.2 Specification of Game Content: Rogue-like Maze

We have confirmed that the game content space was segregated from the education material space. As an overview, we generated game content using parameter values that consist of \textit{maze-id} (categorical), \textit{enemyType} (0: random-move enemy, 1: simple-move enemy, 2: smart-move enemy), \textit{numberOfEnemy} (1-5), \textit{numberOfBullets} (1-5).

\footnote{available online: https://github.com/mpalmerlee/ChemFight} \footnote{available from: https://www.ncbi.nlm.nih.gov/, and last accessed on 22 September 2018}


Table 3.1: The conditions to categorise game content based on difficulty level.

<table>
<thead>
<tr>
<th>Difficulty</th>
<th>enemyType</th>
<th>numberOfEnemies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy</td>
<td>Random</td>
<td>&lt;4</td>
</tr>
<tr>
<td>Medium</td>
<td>Random</td>
<td>&gt;3</td>
</tr>
<tr>
<td></td>
<td>Simple</td>
<td>&lt;3</td>
</tr>
<tr>
<td>Hard</td>
<td>Simple</td>
<td>&gt;2</td>
</tr>
<tr>
<td></td>
<td>Smart</td>
<td>any</td>
</tr>
</tbody>
</table>

By default, the game content space comprised of 48600 playable games.

In difficulty categorisation, we generated more than 30 different games, played and experienced their elicited difficulty. Based on our observation, three levels \( (c = 3) \) of challenges were sufficient to separate the game content. To our best knowledge, the parameters enemyType and numberOfEnemies distinguished the difficulty quite noticeably within the set of conditions in Table 3.1. As a result, 22365 of game content were categorised as Easy, 15660 were Medium-level game content and 10575 of content were of the Hard difficulty level. Fig. 3.2 pictures three different levels of difficulty. The image on the left is identified as an Easy game. The cause was merely a single obstacle from an enemy which moved randomly, but we moved the avatar freely without much concerns colliding the enemy. Similar challenges were created by game content in which the numberOfEnemies was less than four. The remaining two images on the right are Medium and Hard difficulty levels, respectively. A medium game with four or five enemies which move randomly was more restricting the avatar’s movements; thus, required a bit of strategy to accomplish the goal and survive from the enemies. When the type of enemy was changed to the one with a simple movement, the difficulty level relatively remained if there were less than three enemies on the arena. Otherwise, the difficulty jumped to the most difficult one. Above all, the game content with one or more smart enemies demanded a high level of tactical practice in decision-making because these enemies were capable of traversing the shortest path to the avatar.

The similarity categorisation had a purpose of providing a selection of similar game content for each learning material within each difficulty category (DC). To accommodate this, we set \( k = 100 \) representing the total clusters of similar game content inside a difficulty group. Given that maze-id attribute did not describe a maze explicitly, five numeric attribute were elaborated measuring the maze’s numberOfPaths, numberOfCorners, numberOfCrosses, numberOfDeadEnd and complexityMeasure. Aside from this, awareness was raised due to the large size of game content space. Therefore, Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) which is fast
3.3. DEVELOPING A TEST-CASE SEG FOR RECALLING: CHEM DUNGEON

Figure 3.2: Exemplary game content of different difficulty levels (left-right): Easy, Medium, Hard (reproduced from [1]).

and flexible even with very large samples (details available in [102]) tackled such an issue.

According to Algorithm 6 we configured BIRCH with \( k = 100 \) and setting the branching factor \( b = 2 \) constructs a binary tree of game content space for easier visualisation. And we set a vector of cluster’s threshold radius \( T = 0.1, 0.11, ..., 1.0 \). Iteratively, the BIRCH produced 100 clusters and the corresponding silhouette scores evaluated the optimum threshold radius value. The result of BIRCH on game content space under normalised values attested to Low, Medium and High difficulty groups using a threshold radius of the clusters, \( T_l = T_m = T_h = 0.02 \), to reach the highest silhouette of 0.23, 0.2 and 0.23, respectively. Overall, 300 clusters identified and equally divided into 3 difficulty levels are ready for deployment with the education materials.

3.3.2.3 Mapping Education Material and Game Content Spaces

Prior to mapping process, we manually identified, analysed and appointed a number of potential game mechanics to fit our SEG construction. We adopted the learning process of multiple choice question in an exam into the SEG. Consider two groups of atom(s) (say, group A and group B) that their combination forms a chemical compound. Group A represented a question, while the B group became the correct answer. Group B will be located with other groups of atoms (say group C, D, E) that do not form chemical compound when combined with group A. This was to represent multiple choices of the question. In the game, group A was visible as the avatar’s atomic shield. Meanwhile, there were options of game mechanics for the group B (represents choices), we ordered them based on the interest level in the existing game content:

1. Fighting an enemy.
2. Collecting coins.

3. Collecting bullets for additional ammunition.

4. Collecting potion to restore the lives.

According to our knowledge about the game, fighting enemies in order to combine their atoms had a conflict of interests. In one hand, fighting has a connotation of destructive actions to the opponent. In contrast, the avatar’s mission is collecting the correct atom(s) that construct a chemical compound. Based on this views, we excluded the first game mechanic as the learning task platform. Meanwhile, the remaining three game mechanics had the purposes that potentially promote rote learning. However, we considered collecting bullets less important as they were the prerequisite for shooting actions. Similarly, potion collection was lacking interest in the game unless being the last option to resuscitating lives. There was an alternative to achieve that by accumulating gaming experience level to the maximum value by either via shooting obstacles/enemies, killing enemies or collecting coins. Such alternatives to reach one goal often introduced in games to engage strategic actions in addressing the main mission. Thus, we preserve potion-collecting as is. Other developers may transform its functionality as a hint corresponding to the learning mission in the game.

In our case study, the most relevant game mechanic to endorse rote learning was solely the coins collection. In the SEG, we converted coins into atom objects that interact with the avatar’s atomic shield. Moreover, the abundance of atom objects in the maze can assist the repetitions required for strengthening the rote learning. Meanwhile, the enemies and incorrect atom objects were the obstacles in the educational game.

Provided that the game content was prepared for the mapping process, former steps in our framework successfully arranged the chemical compounds from the simplest to the most complex to memorise. And the game content has been categorised into difficulty levels and similarity clusters (SC). Given the attribute values in learning material and game content spaces, the core mapping requirement ascertained no duplicates of game content for multiple compounds. Following the procedures in Algorithm 7, the units of knowledge (E) were already sorted from the knowledge categorization step.

Then, we added two blank new attributes (EID and CS) into each game content. Yet, we defined additional criteria that ensure an appropriate mapping based on our notion of possible learning conditions between simple versus complex compounds. In principle, a player may require a higher number of game content choices to recall complex compounds. Because a more frequent failure may be experienced by the
3.3. DEVELOPING A TEST-CASE SEG FOR RECALLING: CHEM DUNGEON

Figure 3.3: Mapping result regarding total games in a cluster (reproduced from [1]).

player. Consequently, a slight difference in the game elements for recurrence of a recalling goal might accustom the player to those game stages without the fear of boredom growing. In that event, the player could address more of his/her focus on the learning mission. Based on these criteria and expectations, the statistic details of the clusters resembled those aforementioned conditions including the quantity of game content (represents the number of repetitions, denoted by $CS_0$) and the sum of standard deviations of game content features (represents the variety of games, denoted by $CS_1$) under non-normalized parameter values.

Iteratively within each difficulty level, a program measured both statistical information of each cluster and annotated them into attributes $CS_0$ and $CS_1$. Subsequently, the clusters were sorted in an ascending order based on the attributes values of $CS_0$ and $CS_1$. Hence, the attribute EID of each sorted cluster were annotated with ascending numbers from 1 to 100 representing the numeric ID of the units of knowledge space.

Mapping priority started with the total game content in a cluster. Fig. 3.3 shows the growing number of game content from the simplest to the most complex compounds. Then, the sum standard deviation of the assigned clusters (Fig. 3.4) shows a very slow decrease of the game content variation. In addition, the resulting content module was kept in a database tables.

3.3.3 SEG Game Stage Generator

Given that Chem Dungeon is a brand new game, a game generator should appropriately prepare different players to have adequate comprehension with the game. Fig. 3.5
shows the default procedures (non-adaptive) in the SEG game session. Initially, a new player should enter the practice game session which contains the educational game with dummy learning materials. Alternatively, an existing player may enter the practice game session for updating his/her Player Level. Once the practice session ends, the achieved score (see Eq. (3.1)) is used to roughly estimate the difficulty level suitable for the player $V$.

$$\text{score} = \sum_{i=1}^{n} \alpha_i a_i^+ - \sum_{i=1}^{l} \beta_i a_i^- \quad (3.1)$$

Where $a_i^+$ be the $i^{th}$ positive game action (e.g., successful moves, accurate shots, successful battle) and $a_i^-$ be the $i^{th}$ negative game action (e.g., lives lost, failed moves). A value of $n$ counts the number of positive game actions while $l$ measures the total negative game actions. Values of $\alpha_i$ and $\beta_i$ set the $i^{th}$ weights for positive and negative game actions, respectively. Then, the threshold values of score categorise a player into a particular level $V$. In addition, various weights (if known by the developer) on particular actions may yield a more accurate scoring.

In the first instance, a new player starts playing the practice game stage with the first education material ($E_0$). Meanwhile, an existing player may progress the next game stage containing the subsequent recalling goal according to his/her game session record (List of Played SEG). Based on $V$ and $E$, the generation engine searches through the content module for the subsequent education material, selects a difficulty category.
3.3. DEVELOPING A TEST-CASE SEG FOR RECALLING: CHEM DUNGEON

Figure 3.5: SEG game session’s procedures (reproduced from [1]).

based on the $V$ threshold and the corresponding game content cluster, as game content candidates. For a new player, the candidates are all game content in the selected cluster, except the played game content. Subsequently, a centroid-based selection chooses the nearest game content relative to the centroid $g_x$. The $j^{th}$ feature centroid ($g_x^j$) is measured by (5.3). Whereas, $x_j^i$ be the $i^{th}$ game content’s $j^{th}$ feature in the pool and $m$ be the number of game content candidates.

$$g_x^j = \frac{\sum_i^m x_j^i}{m}$$ (3.2)

Finally, the game engine generates the selected game stage for the player to play it and accomplish the recalling goal.

3.3.4 Chem Dungeon Game Mechanics

This section demonstrates the game descriptions and instructions to play the newly-developed SEG as observed in Fig. 3.6. The game arena is basically a maze consists of pathways, walls, intersections and dead-end alleys. There are also spawn points for the game’s actors and an exit gate (initially closed and hidden) at the bottom-right of the arena. Actors in the game composed of an avatar and some opponents. The avatar carries an atomic shield (represents an atom that is ready for bonding with others) wherein the atom’s details are shown nearby the avatar’s spawn point. When the avatar is hitting an atomic mine (blue shield), it will inform the compound-forming result or
Figure 3.6: Chem Dungeon’s game stage layout (reproduced from [1]).

an atom properties at the top-centre of the game arena. On the right side of the game (from top to bottom) contains lives (heart icon), experience in a red bar, ammunition (number), bonds made (number) and the remaining time (90 to 0 seconds). Inside the maze, bullets (yellow object), atom objects (blue object) and live potion (red object) are collectable for the avatar. Each bullet collected adds some ammunition for the avatar. A live potion can restore the avatar’s life to full.

The objectives of playing Chem Dungeon are getting the correct atom to form a compound and entering the exit gate within the 90-second time limit. Initially, the avatar is spawned in its home and the enemies start in the diagonal paths of the maze (bottom-left to top-right). The avatar can walk in 4-degrees of freedom: left, down, right, up controlled by keyboard keys $a$, $s$, $d$, $w$, respectively. When wandering in the maze, the avatar should avoid colliding with an enemy or an atomic mine. In fact, it will lose one life when colliding with an incorrect atomic mine or a strong enemy. Alternatively, the avatar can shoot an atomic mine to unblock the path. If the bullet hits a strong enemy, it transforms the enemy to a weak mode (white-coloured character).
Then, a weak enemy re-spawns back to its home when captured by the avatar, thus, clearing another route. Accordingly, the avatar can explore and compile the correct atom (mine) which forms a compound with its atom (shield). At this point, an educative message pops up which informs the chemical compound details. Consequently, this game state should encourage players to read and retain the corresponding knowledge in their memory. When the avatar has collected the correct atom object ten times, the exit gate reveals to open. Finally, by entering the exit gate, the avatar gets a Victory. Otherwise, losing all lives or out of time issues a Defeat.

The following illustrates some helpful hints for players to play the game. Although each game contains different atom choices, its aim is to form one compound (repeatedly). Beginner players often apply a trial-and-error method and are fully aware not to waste all their lives. Therefore, the player ought to carefully read the text message corresponding to the latest result of the compound-forming effort. Meanwhile, whenever one life left, a player must recover it back to full by collecting a potion. Or, alternatively, by accumulating game experience (xp-bar) via accurate shots and capture weak enemies. Once the XP reaches a full bar, one additional life is the reward. However, such an endeavour should consider the remaining bullets/ammunition and the 90-second time limit. These constraints impede players abusing such tactical practices merely for entertainment while neglecting the main purpose of playing the game: retaining compound forming in his/her memory.

3.4 User Feedback

Under our development method that mix of education materials and game content, a new game called Chem Dungeon was created. Therefore, we ran a survey about the player’s experience when playing the game. We administered the survey only for players at least 18 years old and computer literate.

Fig. 3.7 depicts the procedures for the survey initiating with a Consent Agreement, Demographic questionnaire, Practice (training) session and Pre-game Exam (randomly chosen learning materials). Afterwards, a player plays a pair of game stages containing one learning goal questioned in the Pre-game Exam. The game stage is set to a difficulty level according to his/her level measured from the Practice game session. Following each pair of games, the player reports his/her fun/enjoyment from the latest played games and answers questions in the Post-game Exam. In addition, each game session produces a game data for further analysis.
CHAPTER 3. SERIOUS EDUCATIONAL GAME DEVELOPMENT FRAMEWORK

The consent form verifies a player’s participation in the survey. Meanwhile, the Demographic form records participant data, including age, location, player-id, email address and a unique code for players to re-enter the survey. A four alternative forced choice (4-AFC) questionnaire demands a player to compare his/her enjoyment between both games [35, 79]. Question wording for the reported enjoyment appears as follows: a) Game N+1 is more FUN than Game N, b) Game N is more FUN than Game N+1, c) Both Games are FUN and d) NONE of the Games are FUN. Meanwhile, Pre and Post-game Exams employ Multiple Choice Question (MCQ) design [103, 71].

Subsequently, a player may revisit the training session if he/she requires improving his/her gaming ability before continuing to the next section of the survey. Alternatively, he/she may opt to directly play a new pair of games initialised by completing another pre-game exam, or just quit the survey.

3.4.1 Analysis of User Feedback

Fifty players participate in the survey that ran for three months and resulting a total of 540 reports obtained. Averagely, players reported ten gaming experiences, while four of them played only a pair of game stages and the remaining 85% played between 4 to 14 game stages. Only one player played and reported 15 pairs of game stages. To analyse the reported experiences, we chose a z-test to infer the difference between contrasting affective experiences, e.g. proportion of enjoyment versus no enjoyment.

In terms of the reported Enjoyment, 352 reports confirmed that the game stages were entertaining while 188 game stages reported they were not enjoyable. Table 3.2 summarises three z-tests evaluating $H_0$ against $H_a$. The null hypothesis $H_0 : \pi = 0.5$, where $\pi$ indicates the proportion of FUN reports. Given the 0.01 significance level, two
3.4. USER FEEDBACK

Table 3.2: Z-test on Proportion of Reported FUN.

<table>
<thead>
<tr>
<th>Indicators</th>
<th>$H_0 : \pi \neq 0.5$</th>
<th>$H_a : \pi &gt; 0.5$</th>
<th>$H_a : \pi &lt; 0.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1</td>
</tr>
<tr>
<td>99% conf. intervals</td>
<td>0.59-0.74</td>
<td>0.6-1.0</td>
<td>N/A</td>
</tr>
<tr>
<td>$H_0$ status</td>
<td>Rejected</td>
<td>Rejected</td>
<td>Rejection</td>
</tr>
</tbody>
</table>

z-tests reject the null hypothesis while 99% confident the proportion of FUN reports (0.652) is greater 0.5 proportion.

By looking into play-logs, slight differences were observable from various gaming activities that separate the reported FUN and NOTFUN. We suspected that this is due to the fact that the enjoyment is a very subjective matter. In one hand, a player may feel 'entertained' because the game was easy enough for him/her. Meanwhile, another player may experience the same thing when the game content was more difficult to play for his/her skill. Such a factor is common that different players could have various perceptual/cognitive experience in response to the same stimuli. Therefore, a thorough questions were suggested to accommodate various factors that correspond with the enjoyment [27].

Regarding the learning performance of the players, each question item in an exam represented a unit of knowledge. Thus, pre and post-game exams produced binary values indicating prior knowledge and recalling results, respectively. The difference in scores between pre and post-game exams produced three types of learning performances, i.e. unchanged, improvement and decay. Nevertheless, only the unchanged and improved learning performances will be used. Because, the negative score (decay) is likely to be produced from arbitrary answers or random guess [56]. In addition, we excluded the improvement of knowledge regarding the known education materials, because the exam was not designed for that. As results, 309 reports involving not known prior knowledge were partitioned into 219 game sessions which were reported to improve players’ learning performances, while 90 sessions failed to improve the players’ learning performance. For this case, the same z-tests analysed the proportion of improved learning performances ($\pi$) for the null hypothesis and alternative hypotheses. Table 3.3 summarises three z-tests results. Given the 0.01 significance level, two z-tests reject the null hypothesis while 99% confident the proportion of Improved Learning reports (0.694) is greater 0.5 proportion.
Furthermore, we investigated the recorded gaming data corresponding to \textit{learning} and \textit{not learning} outcome and found that gaming activities seem correlated to the learning outcome. In general, a game session where players recalled the education materials has more gaming activities than a game session where players did not recall the education materials. In fact, the total time spent in reading the successfully collected compound corresponding to \textit{learning} actions took around 15 seconds on average. In contrast, the \textit{not learning} actions mostly spent less than three seconds on the same action. Overall, the total actions in \textit{learning} game sessions was approximately twice as many as the \textit{not learning}. The reason behind this was driven by straightforward learning goal of this educational game, in that it was designed to collect as many bond-able atoms as possible.

The statistical analysis shown in Table 3.3 and Table 3.3 affirms that the Chem Dungeon game was considered successful from the players’ perspective regarding their learning and enjoyment. Such result is consistent with Pavlas’ testimony [101] that learning as the main goal is supported by enjoyment in the game.

On the other hand, we also looked into the relationship between learning outcome and affective experience reported by the survey participants. Table 3.4 summarizes such information collected from all the game sessions. It is clear from Table 3.4 that our SEG allowed more players to gain positive learning outcome together with Fun as there were just about 50\% out of 309 reports fell into this category. This confirmed that the application of our SEG development method may lead to an SEG that fits the specifications described by Abt: in a serious game, learning may be primary but other experiences involved should not be overlooked [2]. Furthermore, it also clarified that serious games involve learning and entertainment dimension as a unity during game sessions [104, 105, 106, 107]. While the learning in serious games is the dominant objective, we take into account the importance of enjoyment as well. Overall, the experimental results reported above indicate that, to a great extent, the produced educational game potentially elicit positive affective experience and many players will

---

Table 3.3: Z-test on Proportion of Learning Improvement.

<table>
<thead>
<tr>
<th>Indicators</th>
<th>$H_0 : \pi \neq 0.5$</th>
<th>$H_a : \pi &gt; 0.5$</th>
<th>$H_a : \pi &lt; 0.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>0.00000</td>
<td>0.00000</td>
<td>1</td>
</tr>
<tr>
<td>99% conf. intervals</td>
<td>0.65 to 0.73</td>
<td>0.66 to 1.0</td>
<td>N/A</td>
</tr>
<tr>
<td>$H_0$ status</td>
<td>Rejected</td>
<td>Rejected</td>
<td>Rejection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Failed</td>
</tr>
</tbody>
</table>
3.5 Summary

Chem Dungeon was developed successfully to demonstrate that two separate content spaces, i.e. education materials and an entertainment game’s content, can be combined. We can generate the Chem Dungeon by controlling the parameter values of the SEG content module and the produced educational game stage is playable.

During the development stages, the available resources of the learning materials provide advantages in two folds. In one, the learning materials have a natural description carried by the properties facilitating a developer to organise them semantically. Second, computer applications can automatically retrieve those attributes’ values with light intervention from experts and with a concern merely for the learning materials’ size. With respect to the game content space, applying PCG in our approach unlock an alternative route towards a speedy development of SEG. As a result, it enables an automated categorization utilising proprietary algorithms, a rule-based method and machine learning algorithm. Hence, those specific descriptions underlying either content spaces facilitate a developer in establishing the mapping requirements based on his/her knowledge. Moreover, the newly produced SEG under our method was reported to supporting players’ learning and entertaining them. However, our approach has limitations and simplifications in some aspects that the following paragraphs will elaborate.

In our framework, we process learning materials and game content separately, then, integrate them in the mapping phase. However, the process prior to mapping does not imply a ‘separated’ process. Because the developer has to make an appropriate selection of the game to host the learning process. In other words, the target knowledge, to some extents, inspires the game selection. In facts, there were qualifications to select the game in the followings. Our method only applies to games that are controllable via parameters and the type of knowledge is limited (explained in the next paragraphs). Then, we need to identify game mechanics suitable for acquiring the knowledge under

Table 3.4: Reported experiences: Learning Performance vs. Enjoyment.

<table>
<thead>
<tr>
<th></th>
<th>NotLearning</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotFun</td>
<td>42</td>
<td>65</td>
</tr>
<tr>
<td>Fun</td>
<td>48</td>
<td>154</td>
</tr>
</tbody>
</table>

obtain it when they engage in learning via game playing.
consideration. We also need to ensure that we can modify the existing game object, under which the appropriate game mechanic of the chosen game, with the learning goal. Unfortunately, our approach does not allow very high-dimensional features in both spaces. Because the redundant, unimportant or noisy elements in both spaces potentially befall. An example of the unimportant element is the choice of font for the game’s information, as long as they are readable. If high-dimensional features are inevitable, a feature selection method should sufficiently choose the optimal input variables. Furthermore, our concern arises when the game content space is smaller than the learning units. As a consequence, our approach maps \( n \) learning units to one game content. This causes an inappropriate experience for the player. As far as being a gamer, we expect a dissimilar game set for the new objective inspired by the learning goal. The same game challenge between two or more different objectives potentially leads to boredom. Therefore, our approach requires the game content space to have at least the same size as the learning materials.

Aligned with such a requirement, the game content space in our test-case was significantly greater than the learning materials. Hence, our approach forced the mapping between a hundred learning units to more than forty thousand game content. Because there were no appropriate natural categories of the game content with respect to total learning units. Only the maximum of 0.23 out of 1.0 silhouette score measured in the clusters. In facts, that results were not satisfactory to our expectation. Conceptually, we can solve that by artificially constructing \( k \)-groups \((k = \text{total learning units})\) of game content. Initially, a clustering technique sets the centroid or medoid for the groups. Then, we add game content iteratively into each cluster which is bounded by the threshold distance to the corresponding centroid or medoid. During each iteration, we need to check the silhouette score whether an additional game content is necessary to be added into the group. As such a step has a purpose to maintain a good separation between the constructed clusters.

Additionally, we understood that our development framework was limited to knowledge that can be learned via rote learning. Nevertheless, there is an opportunity to practice our framework for a higher order of learning, such that found in the understanding category. In that, we may embed some learning types under this category that includes grouping, identification, recognition, selection or translation to create an educational game. Because we can learn them via repetition which allows straightforward mapping with the game content. Unfortunately, the other learning methods in the understanding category like describe, discuss, explain and report may have a small
applicability to our development framework. Because they require learning techniques beyond repetition. To facilitate that, the game content should provide a proper platform for such a complex learning. For instance, a game content with embodied narration to control the game’s story and experience [7, 37]). Alternatively, we can also take into account multi-player game genre where a player can collaborate with other players to demonstrate complex learning process. For such a purpose, Kim et.al. introduced a meta-cognitive learning support in the game [108] which may play important roles when one involves a complex knowledge in the SEG development. Hence, the more complex learning of the knowledge to be combined with a game content space, the less applicability of our framework to develop the SEG. To address this issue, the framework may have to involve experts in the categorisation of learning materials and the mapping process.

For now, our framework fits suitably with the existing action game genre, i.e. roguelike and maze. We recommend our framework for the other game genres like action-adventure, adventure, logic (e.g., puzzles) and trivia. Because their gameplay and game mechanics support straightforward mapping with the rote-learning knowledge. We can estimate that relatively the same portion of modifications is required to the target game content. Suppose the game content is a platformer game known as Mario Bros. and the knowledge to be learnt consists of chemical compounds. We can modify the game mechanic to require the player collecting atom items (instead of power items). Once the avatar entered a warp-pipe, the game shows the chemical compounds from the collected atoms. Similarly in a puzzle game called Tetris for instance, one can substitute puzzle objects into boxes containing atom symbols. Whenever the adjacent atom boxes form a chemical compound, a corresponding knowledge-related information pops up. Then, the involved boxes are re-allocated in a basket outside the game arena (for rehearsal). Fundamentally, a wide range of game genres is applicable to our framework. Notably, the game content with simple mechanics. A game genre such as Real Time Strategy (RTS) or Role-playing Game (RPG) may be too complicated to be applied in our framework. Because we have to pre-process the game content (e.g., modifying game mechanics, scenario, game rules), quite significantly compared to the simpler game genres to support the mapping process.

In addition, our framework focuses to help developers create single-player educational games. This makes typical players, especially Achievers and Killers types [109], optimally learning-playing in this way. We are considering multiplayer game setting, at least making the test-case SEG “feels” like a multiplayer game. This can be done
by adding a feature in the SEG, such as players ranking based on their overall learning tasks achievements, scores or other measurement methods. Alternatively, applying a multiplayer game content into our framework will produce a serious game with a higher degree of interactions between players. For instance, communication and interaction, as well as a competition between them. Learning in this type of game may benefiting the most for players who love to socialise and explore in the game. They can exploit the communication feature and environment to engage themselves [109]. The biggest challenge rests in the developer’s hands for creativity to ensure that the multiplayer facilities are optimally employed for players’ learning. For example, the developer can adjust the game’s scenario that encourages players getting involved in a chat room (inside the multiplayer game) to collaboratively solving a learning task. It is recommended to apply collaborative learning in a multi-player game environment through knowledge engineering process that promotes a higher degree of cognition [110].
Chapter 4

Non-intrusive Assessment in SEG

In Chapter 3 we have developed the Chem Dungeon content module that the game engine can generate playable game stages of this new SEG. In this chapter, we build the next module into the SEG that enables it to assess a player during the game session without interrupting his/her gaming immersion. Chapter 2 briefly explains some existing works related to our approach. Based on that, we begin this chapter with the motivations that drive our approach followed by the proposed methodology. Then, we provide the analysis of the collected data and the developed prediction of the player’s experiences.

4.1 Motivations

From the aforementioned methods of known learning and enjoyment assessments in a serious game (see Chapter 2), most of them have to interrupt the gaming session if they are directly applied. This is due to the fact that existing assessment methods are non-game-based evaluations, such as using a self-reported questionnaire or a written examination. Hence, this type of assessment potentially distracts the gaming immersion because of the contrasting experiences shifting from fun gaming to a serious reporting. In the meantime, an assessment must continuously monitoring the player’s achievements when a serious game is adaptive [19]. Therefore, there is a need of a reliable assessment method in serious games (SGs) that maintains the gaming excitement and immersion.

Three types of assessment methods applicable in SGs [17, 55] are as follows: (i) completion assessment, (ii) in-process assessment, (iii) teacher evaluation. In the first, a completion assessment can be regarded as an accomplishment of a mission’s goal,
which is reflected an acquired knowledge or skill. In other words, the final result of a mission is linear with the player’s knowledge acquisition. Nevertheless, it entails a question of how the success in a mission was achieved, whether through true dedicated learning, a coincidence, or through a cheating. Regarding the second, the teacher’s evaluation involves an observation of a player’s behaviour in a game session, especially those aspects that are difficult for a computer to capture and compute, such as emotion and motivation. However, this method is too laborious and requires specialized expertise of the teacher. Lastly, an in-process assessment monitors a player’s in-game behaviours and a function translates them into the desired outcome. This method is more suitable for SGs since it analyzes the reason, timing, and methodology in performing the tasks during the game sessions. The reason for this is the fact that through playing an SG, the player’s behaviours contain abundant information, e.g., knowledge acquired and affective states [7], learning style [14], motivation [22, 30].

Pedersen in his research revealed that in-game actions can reflect the affective experiences of the player [35]. The research findings show that players enjoyed a feast of running, numerous enemy killings and collected coins were indicated by in-game actions such as the number of times the player killed opponents, the total run button being pressed, and the proportion of time spent running [35]. This proves that although those affective experiences were abstract, they were observable in the game actions of the player. We agree that learning, as one of abstract experiences of the players, is also observable via the in-process assessment. Yet, the problem in applying an in-process assessment is the difficulty of identifying the characteristics of attitudes or behaviours or achievements during the game session that are reflecting the player’s experiences (i.e. learning and enjoyment) reliably. Because, the game mechanics, game plays and game rules all play important roles in differentiating the play-log features the reflect an experience of the player between one game and the other. Therefore, we need a technique to generalise the relationships between an experience of the player and the in-game data. Inspired by previous methods (e.g., [35, 28]), we administered a survey to collect data from players labelling their in-game data. Then, we apply supervised machine learning algorithms using the collected survey data containing training samples to generalize enjoyment and learning outcomes via the in-game data.

The main contributions of the proposed approach are: 1) we develop non-intrusive assessments for serious games, and 2) we identify prominent features that can be useful for future development in similar projects.
4.2 Methodology

In general, we apply supervised machine learning method to predict a target outcome/experience of the player. To achieve this, we need sufficient examples of data which consist of input features and label. In one hand, input features (in gaming) usually originate from statistics calculated by the game activities. Statistics from key presses and mouse events [76] are the source of raw independent variables. Most commonly, input variables sourced from the in-game events, such as jumps, kills, play duration, etc, are more interpretable for game playing analysis. By getting deep into the game rules, gameplay and game mechanics, or through intense observation of game sessions we can identify such potential input variables. Additionally, expressions from the player are also the source of information taken from a microphone or camera. In many game cases, players speak (or even scream) to themselves that correspond to the game situation they are facing. For instance, players often being expressive (i.e. voice and facial) when killing the enemies they are targetting. Nevertheless, identifying all potential input variables becomes the challenging process to apply machine learning to our approach. On the other hand, one may obtaining the label or dependent variable via manual annotation when feasible. Alternatively, a survey or data collection phase allows participants annotate the data. Then, we need to pre-process the collected data by excluding unreliable, noisy or outliers samples. Sometimes incomplete samples contain key information about the target experience. Thus, we may assign values to the missing labels or input features. Subsequently, we choose a machine learning algorithm to predict the target experience, commonly known as training phase. Sufficient justification based on the typical samples’ output (classification or regression) and advantages of the learning method should select the appropriate algorithm. Alternatively, one can run a competition between different machine learning algorithms.

Then, we need to build prediction model(s) by training them using the samples. In that, we need to measure the performance of these prediction model(s) using appropriate evaluation functions such as confusion matrix (classification) or mean-squared-error (regression). Instead of a one-go prediction modelling, a cross-validated (CV) technique, such as k-fold CV, should minimise the bias of the produced models. This CV method divides the collected data into k partitions. There are k iterations of training phase using a different combination of training and test samples. From these iterations, we summarise the performances of the prediction models. With this technique, we search the best parameters of the machine learning algorithm. By iterating through values of the parameters, the aggregated performances in CV technique show how well
a machine learning algorithm predicts an output. Then, the final prediction model is
the trained machine learning algorithm using the parameters of the highest prediction
performance.

For our case, we propose a method that has been successfully applied in the field
of entertainment game. Conceptually, it should work for other games and any corre-
sponding experience. In general, we treat the SEG as a platform where players play
and learn with it. We believe that different players behave differently with the game
set driven by their internal conditions (e.g., learning, enjoyment, motivation, etc). This
behaviour is recordable as play-log during a game session. However, it is difficult to
measure such abstract experiences directly from the log without proper models. In that,
a model is a generalised relationship between play-log with an experience. To achieve
that, we apply a survey where each player self-reports his/her experience with the last
game session. This report acts as the label of a game log. In a survey, we can obtain
sufficient samples of the labelled game sessions. If necessary, we can pre-process the
collected samples to make them reliable. Then, a machine learning method should be
able to model the relationship between game log with the experience. Given a new
game log, the model should predict the experience confidently.

4.2.1 Enjoyment Prediction Modelling

As previously mentioned, we apply an approach successfully implemented in [28] and
[35], which correlates some reported enjoyment with their corresponding in-game be-
haviours and game content features. In a survey, each player reports his/her enjoyment
after a game session, such as via Likert scale in [28]. However, this measuring scale
often made the participants inconfident to make extreme rates. Hence, samples did not
represent the experiences accurately.

Alternatively, four alternative forced choice (4-AFC) compares a pair of games
whether one of them is more entertaining, both of them creates enjoyable experiences
or neither are fun. This method requires a participant to play a pair of games. Then,
s/he must compare the enjoyment between the latest two games. The wording of this
questionnaire is as follows:

- Game N is more Fun than Game N+1
- Game N+1 is more Fun than Game N
- Both games are Fun
4.2. METHODOLOGY

- Both games are Not Fun

Each reporting session creates two binary-labelled samples. Say, the player plays \( game_0 \) and \( game_1 \) subsequently. The first answer creates samples of \( game_0 \) and \( game_1 \) with labels TRUE and FALSE, respectively. Otherwise, the next answer labels the same samples the opposite values. Meanwhile, if s/he reports both game were Fun; then, \( game_0 \) and \( game_1 \) have TRUE labels. Otherwise, if the last answer was picked; then, \( game_0 \) and \( game_1 \) have FALSE labels.

Once all samples collected, we use them as the training samples for the binary classifier to predict the enjoyment. However, the binary classifier must use the parameter values that maximises its cross-validated enjoyment prediction. Sometimes, it is also necessary to experiment with a reduced set of independent variables of the classifier. Thus, a simpler enjoyment classifier is built, yet a maintained or improved classification performance. We elaborate the enjoyment modelling in Section 4.5.1.

4.2.2 Learning Prediction Modelling

Regarding the change of learning, the same data collection applies (i.e. survey). However, participants play one game each session which is sandwiched between a pre-game test and a post-game exam. The purpose of two exams are measuring changes of knowledge before and after a treatment (i.e. playing a game). Each exam is a knowledge question with four choices. However, the order of choices are shuffled between exam and the question wording is modified. The setting is intended to minimise the effect of a past question. Hence, a player responds to the post exam question based on his/her knowledge from playing the game.

A game stage in the test-case SEG contains a learning task; thus, both exams ask the same knowledge. The score in each exam is a binary value representing a correct (1) or wrong (0) answer. These produce one sample containing game data and the label (i.e. the difference between pre- and post-game exam results). Such a label represents the change of knowledge of a player (we call this a learning performance) as a product of an in-game learning treatment. Learning performance (LP) is basically the class label categorising a player who is learning (memorising) or not learning the education material in a game session. One can quantify its value explicitly by measuring the difference between the results of pre- and post-game (L0 and L1) exam scores (Eq.(4.1)).

\[
LP = L_1 - L_0 \tag{4.1}
\]
In our case, the values of $L_0$ and $L_1$ are binary. Therefore, there are four possible LP values shown in Table 4.4. Despite the four possible classes of Learning Performance, our project only used the three of them. First, if pre- and post-game got 0 scores, then, the corresponding sample is categorised as LP0 representing a player who failed to learn due to no prior knowledge. Second, if a positive difference was found between post- and pre-game exams, then, the corresponding game data is categorised as LP1 thus representing a recognition of new knowledge unit has occurred. And third, if a player initially memorised the education material prior to playing the game and correctly answered the post-game exam, the corresponding game data is categorised as LP2 where no learning exists due to no change of memorisation, but it is a successful recall. The one category excluded is when a player made a pre-game score higher than the post-game score (i.e. a decaying learning performance). We assume such results may be caused by arbitrary answers either in a pre-game or post-game exam, thus, samples with a negative learning category are acknowledged as outliers.

The collected samples contain three class values. Therefore, the next phase aims to build a three-class classifier that predicts learning performance. Prior to that, we need to iterate through a set of values of the classifier’s parameters. Then, we choose the LP classifier of the best cross-validated classification performance. If necessary, we experiment with a reduced set of input vectors to shrink the classifier yet maintain or improve its prediction accuracy. See details of the experiments with LP prediction in Section 4.5.2.

4.3 Data Collection

4.3.1 Survey Design

One of the aims of our project is to be able to predict the learning performance and enjoyment of a player when playing the serious educational game (SEG). Our approach is through applying supervised machine learning that requires us to obtain the labels (i.e. learning and enjoyment) for specific in-game data. To accommodate this we use a survey technique (onsite and online). Participants from both surveys are invited and monitored during the sessions. This is to make sure that they can complete the survey in a controlled environment.

Fig. 3.7 in Chapter 3 depicts the procedures of the survey design. In general, there are three groups of procedures, including:
4.3. **DATA COLLECTION**

1. **Introduction.** At this point, a participant begins with a consent form, demographic questionnaire, practices with the game, and reads the tutorial.

2. **Questionnaire and Exam.** There are three types of questionnaire in this part of the procedure. The first is a pre-game exam, which will assess a participant’s initial knowledge of a number of education materials (chemical compounds) embedded in the game stages. This pre-game exam runs once in the survey before the game stages are generated. The next type is self-reported enjoyment, which compares the latest two played game stages. And finally, a post-game exam is conducted at every completion of a pair of game stages. It asks two questions with respect to the education materials embedded in the latest pair of game stages. The results of this part become the labels for the corresponding in-game data.

3. **Game Session.** In between Pre-game Exam and Self-report Enjoyment there are a pair of game stages a participant must play. A game stage is an educational game with an objective to recall a learning material, which is questioned in the pre and post-game exams. The result of each played game stage is a game data.

We conduct the survey in multiple sets in which one set consists of survey groups 2 and 3. We expect participants to play as many sets as possible. With multiple sets of survey, each participant can become consistent with their experiences, i.e. learning and enjoyment, and gaming behaviour. In addition, we invite participants within two different survey sessions (1 month session A and 2 months session B) separated by 2 weeks period. The difference lies in the selected SEG for session A (to make sure 150 SEG have been played at least once), while in Session B the SEG are randomly selected from the list as a way to balance the distribution of the reported experiences.

### 4.3.2 Survey Procedure Description

The survey design depicted in Fig.3.7 comprises several steps a participant should complete in order to provide a consistent dataset. We explain these steps as follows:

1. **Consent.** This consent form consists of the ethical application regarding the survey on human participants. It requires the participants should be at least 18 years old and computer literate. A participant can sign the consent form to proceed with the survey. The online participants may authorize the consent form
by browsing the document on the survey page and clicking the ‘agree’ button to proceed.

2. **Demographic.** This is a questionnaire that consists of the participant’s email address, age, gender, two-dimensional game experience and gaming frequency, which will be stored in a database table: `surveyplayer`.

3. **Practice Game Session.** Game rules in this session are exactly the same as the main education serious game, except the education content is dummy and the session is repeatable. A game session is limited to 90 seconds for a game stage exploration. The purpose of this session is to introduce the basic game mechanics and rules for the players. With sufficient experience of the educational game, this allows the players to focus on learning the education material in the SEG. This session also records the player’s behaviours as score achievement starting from 0 to unlimited. The scoring scheme details are explained in Chapter 3. An estimated ability of a player’s gaming is the average score achieved from the practice game sessions.

4. **Tutorial.** A text-based introduction, game objects, game mechanics and game rules are presented here. This tutorial page is re-callable when a player pauses the game stage.

5. **Pre-game Exam (PreGE).** Some questions related to the player’s initial knowledge about the education materials in the SEG. Before answering the questions, the participants must state how many game stages pair they want to engage in: at least one pair and as many pairs as possible. One pair of game stages consists of two different learning tasks. The question type for testing the knowledge of the education materials is a multiple choice question (MCQ). In our project, a question item is asking about a chemical compound created from two or more chemical elements in the periodic table. The player must select the options that answers the question correctly. We retain all answers of the PreGE in the database (table: `exam`). Additionally, a new row of post-game exam’s result in the same table is also created in the same database table to later be updated when the post-game exam has been completed.

6. **Game N.** This contains the same educational game in which a game stage has an embedded education material that was included in the pre-game exam. It
has specific information (hint) about whether a known compound (correctly answered from pre-game exam) is the game’s objective. The purpose of this information is to preserve the initial knowledge of a player thereby ensuring their consistency. In this way, a correctly guessed or true-correct answer in PreGE will lead to playing activities from a player who knows how to identify the correct compound in the game. This can also reduce ambiguous player behaviour with respect to his/her initial knowledge. In other cases, if a player does not know the correct compound in PreGE, he/she will play the game without needing the hint. This will therefore compel the player to explore the game to find the correct compound.

Every action made by the player is recorded sequentially as string literals; we call this play-log. When the game ends, either due to a Victory, Defeat or Time End, the game parameters and playing activities will be concatenated into one stream of string and stored into the database (table: surveyplaylog).

7. **Game N+1.** The same as Game N, except a different education material is selected from the list of answered items in PreGE.

8. **4-AFC Self-report Enjoyment.** See Section 4.2.1 for details of the wording.

   All answers from this questionnaire are stored in a database (table: surveyfun) which labels the corresponding play-log.

9. **Post-game Exam (PostGE).** There are two question items in this PostGE that test the knowledge of the player about the correct chemical compounds he/she found when playing the latest pair of games. The wording of the question is different from PreGE, but, it is asking about the same materials from the latest pair of games. And, the choices are also randomized to reduce the effect of the questioned items in the pre-game exam. All answers will update the corresponding PostGE row previously created during the PreGE session. The difference in the results between PostGE and PreGE labels the corresponding play-log.

### 4.3.3 Database Design

This section explains the design of database used for our project as well as the current survey. We use MySQL database to link up with the web-based SEG we have built and the corresponding questionnaires. All tables use the primary keys ‘id’ and they are
useful to build the relationship between tables by way of foreign key(s). The relational database design can be seen in Fig. 4.1.

In the database we store the parameters of categorised gaming contents in a table named game_cat. Separately, edu_materials table retains the learning materials to be embedded into the game. A table (edu_game_map) links items in game_cat and edu_materials to create a fully functional configuration of the SEG. Later, an SEG will be generated based on selected rows that specifies the desired parameters.

We created other tables for survey purposes. A database table surveyplayer stores demographic information of the participants as well as some details of their survey and progress. During PreGE, random education materials are selected for a participant’s session and the exam table will save the result of PreGE and then create a PostGE row for him/her. That table is the bridge to select the details of education materials and game parameters to proceed. The PostGE row will be updated when a PostGE has been completed by a player. A database table surveyplaylog records players’ played
### 4.4. DATA ANALYSIS

<table>
<thead>
<tr>
<th>Playing Routines</th>
<th>Total players</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Never</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Very rarely</td>
<td>21</td>
<td>42</td>
</tr>
<tr>
<td>Once a month minimum</td>
<td>13</td>
<td>26</td>
</tr>
<tr>
<td>Once a week minimum</td>
<td>9</td>
<td>18</td>
</tr>
<tr>
<td>More than once a week</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td><strong>Overall Players</strong></td>
<td><strong>50</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>

game parameters, play-logs and his/her measured player level. It also saves the id of the reported experience linked to the database table `surveyfun`, which stores the self-reported enjoyment.

#### 4.4 Data Analysis

In this section we summarise various details of the data collected from our survey of the experiences of the participants. We begin with an overview of the survey participants and the resulting data, and the description of variables in the data set.

#### 4.4.1 Overview of Survey Results

The survey restricted the participants to be at least 18 years old and computer literate, but not limited to a particular gender. There were 50 participants who joined the survey through on-site or online schemes. In the on-site survey scheme, most of the time we visited the participants and a few of them came to the place we provided. Dominantly, participants preferred the online scheme because they could choose the most convenient time and place to complete the survey. The participants were divided by gender in that 84% of the participants, or 42 from 50 participants, were male players dominating the survey. Meanwhile, the remaining eight participants (16%) were female because most of the invited female candidates were not confident enough doing the game-related survey. This gender domination clearly showed that game-related activities were more interesting for male players, although the fact is that the educational game in the survey was not a gender-driven game and some rewards were provided for completing a number of sets of the survey.

Meanwhile, the various gaming experiences of participants are listed in Table 4.1. Players who played two-dimensional games fewer than once a week were novice
Table 4.2: Survey Participants and Samples by Sessions.

<table>
<thead>
<tr>
<th>Description</th>
<th>Total</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Players participated only in Session A</td>
<td>8</td>
<td>17</td>
</tr>
<tr>
<td>Players participated only in Session B</td>
<td>24</td>
<td>46</td>
</tr>
<tr>
<td>Players participated in Sessions A+B</td>
<td>18</td>
<td>38</td>
</tr>
<tr>
<td>Samples from Session A</td>
<td>134</td>
<td></td>
</tr>
<tr>
<td>Samples from Session B</td>
<td>406</td>
<td></td>
</tr>
<tr>
<td>”NOTLEARNING-A” Samples</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>”NOTLEARNING-B” Samples</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>”LEARNING” Samples</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>”NOTFUN” Samples</td>
<td>165</td>
<td></td>
</tr>
<tr>
<td>”FUN” Samples</td>
<td>375</td>
<td></td>
</tr>
</tbody>
</table>

gamers who actually needed several practice gaming sessions before playing the main serious game and thus provide a reliable play-log. Therefore, the practice gaming sessions will be provided as an introduction for new players of the adaptive serious educational game in this project. In the table, just about more than half of the participants (52%) were novice gamers who play 2D games less than once a month (21 players,) and five players were never tried playing this type of game. At the same time, amateur gamers comprised 44% of those who participated of which 13 players play this kind of game at least once per month and nine players play them more frequently within a month. And merely two expert gamers joined the survey. This numbers inform to us that the survey participants may not fulfilling our expectation of collecting a balanced number of participants with respect to their gaming routines. However, it does prove that providing a practice game session is an important factor in obtaining reliable play-logs from various players. Because, the SEG we built is a new kind of game that sufficient skills gained from a practice game session enables any level of player perform relatively consistent.

We ran the survey in two sessions (i.e. Session A and Session B). The first session was intended to collect data from players playing a number of selected pairs of game stages. The pairs of game stages were selected based on the difference between the education materials of the two games. For instance, Game N embedded CO (Carbon Mono Oxide) and Game N+1 contained CO2 (Carbon Dioxide). Meanwhile, the last session had the purpose to improve the balance of data distribution. Table 4.2 lists the participation in the sessions. The list shows that 18 players joined the survey Sessions A and B, eight people only joined in Session A and most (24 players) participated in Session B only. In the first session we were unable to achieve a good balance
between all the reported experiences (i.e. three classes of learning performance and binary labels of enjoyment). Therefore, we invited the former and some additional participants in the second session to improve the classes distribution.

Table 4.2 also lists the amount of samples (i.e. a sample consists of play-log from the played game and learning and enjoyment reports) collected from each session. The number of games completed by the players is ranging from a minimum of two games to a maximum of 30 games. From those participants, we collected a total of 540 play-logs labelled with learning and enjoyment. We collected 134 samples from Session A, while nearly three times more samples (406) were collected from the second survey session. Regarding the learning samples, there are three sample distributions. In the first, only 90 samples were labelled NOTLEARNING-A, which was given when pre- and post-game exam results were zero. In the second, 231 samples were labelled with a NOTLEARNING-B representing a 'not learning report' due to a player knowing the education material before and after playing the game. In the third, 219 games were labelled as LEARNING which represents the process of recalling the education material indicated by the positive difference between post- and pre-game exam results. Likewise, the enjoyment classes have an unbalanced distribution wherein 165 samples were labelled as NOTFUN compared to the 375 samples with a FUN label. These numbers of class distribution are of concern to us in that the imbalance distribution may be affecting the predictive model performance, such as in a biased confusion matrix.

4.4.2 Independent Variables

As aforementioned in 4.2 section, an abundance of input variables are available for this approach. While raw-data statistics are too complex and hard to picture, the in-game event statistics are more reliable for, especially, post-processing (e.g. interpretation and analysis). Meanwhile, input data from external sources are also considered for inclusion. For example, a camera could be the focal source to assess students knowledge during a learning session. However, only specific knowledge are assess-able via facial expressions, such as psychology, human behaviour, affective experiences, etc. Pedersen [35], Robert [28] and Buckley [76] proved that affective experiences play important roles in gaming activities. Their approaches used in-game data to predict the experiences. Alternatively, a camera can capture such experiences from the facial expressions or gestures of the player. As a consequence, an automatic expression
CHAPTER 4. NON-INTRUSIVE ASSESSMENT IN SEG

prediction demands a strong computing power. When the game is intended for an intensive observation of a patient’s psychophysiology, experts are taking benefit of the video-recorded data.

Similarly, auditory data from microphone could also be a good source for an assessment purpose. An attached microphone captures time-series auditory features such as amplitudes, decibel, tone or words coming from their expressions. These types of data are rich with information. A game designed for language learning, for instance, exposes information from the microphone to assess players ability in pronouncing a word or a sentence correctly. There also a possibility to correlate spoken words or shouts from a game player to his/her affective experiences such as enjoyment, satisfaction, etc. For instance, a player loudly shouts 'Yesss' to express his/her satisfaction. Most multiplayer games allow players to communicate strategies, information, gratitude or insults via microphone to their fellow players or even their competitors. This audio communication added a distinct functionality from non-multiplayer games. In this case, audio information is open for research to correlate it with specific experience. For instance, spoken words are sent to a speech-to-text engine to produce a series of text. Then, a sentiment analysis algorithm reads these texts to estimate the positive or negative experience of the player. One can accumulate the predicted sentiments over a period of time or until a game session ends to aggregate the overall experience of the player.

Therefore, identifying all possible independent variables for a game that monitors the player should consider the main purpose of the game. This becomes the precaution of unimportant data inclusion that leads to unnecessary computing complexity. So, the principle in our assessment module is efficiency and effectiveness. This means, the dimensionality of the independent variables should be minimum and they are prominent to the assessment target. Our method chose independent variables based on the developer’s knowledge about the SEG via thorough observation to the game sessions. From all possible in-game statistics, the developer chose the most potential ones corresponding to either learning or enjoyment. We followed the approach in [30] where in-game statistics are not too large to be observed by the developer. Alternatively, we can identify all possible in-game data, considering the valuable information in them via a workshop involving the developer and relevant experts to extract candidates of independent variables. The similar method was applied in [19] that enables different perspectives to look for specific in-game data to represent an experience of the player.
In a workshop, participants must be given the same comprehension of the target experience and details of the game under consideration. Then, participants observe each game session together. A game session is a ‘scripted’ scenario in the game that a player should play and show the game events to the participants. During a game session, participants must list as many information sources as possible to become the candidate for the independent variables. Once game sessions completed, the collected list is then shown to all participants for a discussion to select the best independent variables by removing redundant or unimportant ones. Although it is potentially more expensive and time-consuming, the selection of in-game data should be the reliable source to picture game-playing experiences. As an alternative, we may use a feature extraction or feature selection technique to statistically selects the most important ones that correspond to the desired player’s outcome, such as [76]. This process ensures that there will be no loss of information from activities in the game.

In general, we have divided input variables into two groups: game content and play-log. In the former group there are maze features that include NumberOfPaths, numberOfCorners, numberOfCrosses, numberOfDeadEnd and the complexityMeasure, and game object features consist of enemyType, numberOfEnemies and numberOfBullets. The feature: numberOfPaths is the total walk-able cells in the maze. We have represented the counts of edges in the maze, i.e. identifiable from every three adjacent walkable cells forming a 90-degree angle, as the feature numberOfCorners. We have counted the intersections in the maze and retained the value in the feature numberOfCrosses. Another form of cell in the maze is the cul-de-sac path (an alley that ends up at a wall) in which the total is kept in the feature numberOfDeadEnd. All values of these features are discrete except the continuous values in the feature complexityMeasure. This complexityMeasure acts as a high-level feature associated with the maze properties [111, 112, 113]. Meanwhile, the feature enemyType has three ordinal values representing the behaviour of the enemies with respect to their difficulty to beat them. The values are as follows:

1. Enemies with random movements,
2. Enemies with a simple path tracking towards the avatar,
3. Enemies with an A* closest path search towards the avatar.

Features numberOfEnemies and numberOfBullets counts the total enemies (i.e. 1-5) in the maze and the available bullets (i.e. 1-5) collectible by the avatar, respectively.
Meanwhile, the play-log group consists of 14 features of in-game actions and achievements. The *movesOK* and *movesFail* features count all steps made by the avatar including the successful ones and the ones that hit the wall, respectively. Features *shotHit* and *shotMiss* count the shot that hit and missed, respectively, for either the enemies or the atoms. The *battleOK* and *battleFail* features count the number of collisions between the avatar versus an enemy in which the former feature causes the enemy’s death and the latter kills the avatar, respectively. During the game session, the avatar is allowed to collect the bullets to shoot an enemy or an atom to open paths. We have retained the collected bullet as a feature called *bulletsTaken*. Similarly, the avatar can collect potions to refill the remaining *lives* to full again allowing additional trials for completing the game stage. There are learning-related play-logs explained in the following lists:

- *bondOK* is a feature accumulating the successful bonding between the avatar’s chemical element with a *coin* representing a chemical element(s).

- *bondFail* accumulates the unsuccessful bonding attempts.

- *bondReadTimeAvg* counts the average time consumed when reading the pop-up information with respect to the successful bonds.

When activities result in negative outcome, i.e., failed battles (*battleFail*) or failed bonding trials (*bondFail*), the avatar lost one of its lives and we keep it in *totalLives-Lost* feature. At the same time, we have also added a common entertainment game’s achievement called *experience* (XP). It is a linear accumulation of positive activities deduced by the fraction of negative play-logs. Lastly, the feature *result* marks the end of the game session with either a *victory* or a *defeat*.

### 4.4.3 Dependent Variables

Generally, there are three learning classes and two enjoyment labels. In this thesis we use the terms and definitions explained in this section.

We have named the knowledge achievement of a player as *learning performance* (LP) in which three categories are explained as follows:

- **LP0** categorises the corresponding game data as unsuccessful learning (recall) due to no prior knowledge of the education material.
• **LP1** represents the improved knowledge of a player who previously did not know the education material, then, he/she successfully recalls it in the game session.

• **LP2** is a category where a player who previously knew the education material and he/she is able to recall it in the game automatically.

Meanwhile, enjoyment is a binary class represented by *true* or *false*. To make it clear in some explanations, we have used the notation of a *true* enjoyment as FUN and *false* enjoyment as NOTFUN.

## 4.5 Prediction of Experiences

Given the dataset collected from the survey is ‘not clean’ yet, we performed a pre-processing before using it to train a machine learning algorithm to predict an output. When applying a machine learning algorithm, we needed to set some parameters that produce the optimum prediction performances. To identify these parameters, we used a *k*-fold cross validation method in which performances are measurable via some metrics defined in Chapter 2. However, whenever the measured performances show values sensitive to the imbalance distribution of the training sets, we may need to re-sample the training set until it reaches an expected distribution. In the following sections we explain the development of enjoyment and learning performance predictions.

### 4.5.1 Prediction of Enjoyment

From Table 4.2 we can see the collected enjoyment reports were 165 NOTFUN versus 375 FUN. Before using the samples as a training dataset, we performed a visual observation on the samples’ consistency. In that we assumed all labelled samples must be of the completed game sessions and additional assumptions took part, such as the following explanations. If a sample indicated an incomplete termination of the game, such as no values in the play-log features, it shall be removed. If a sample has too few values but the game objective was not accomplished due to running out of time or losing all the lives, is an obvious NOTFUN category. Because, the player did not sufficiently experience what is so-called an enjoyment. On the other hand, if a sample contained very high counts in his/her play-log features representing the positive actions (e.g., total of successful bonds) and low values in the negative actions (e.g.,
total of failed moves) it was apparently labelled FUN. This is because such a sample fit the definition within Flow Theory that a sufficient match between skill (shown by the positive activities in the game) and challenge (i.e. the game session) potentially led to an enjoyment [36, 101]. Our observation identified that 18 samples were falsely labelled as NOTFUN (instead of FUN) and 23 samples were labelled FUN instead of NOTFUN. After readjusting these inconsistent samples, the distribution of enjoyment became 188 and 352 reports of NOTFUN and FUN, respectively.

We chose Python’s Random Forest (RF) algorithm to build the predictive model of enjoyment. Because, RF generalised well even with a very high dimension input vector. However, the RF required the appropriate values in its parameters to predict accurately. Normally, the RF has more than ten tuning parameters but we found out that only four of them were significant for the RF’s decision trees construction. These four tuning parameters in our experiments were: total trees in the forest (n_estimators), maximum input features for a tree’s construction (max_features), minimum samples for tree splitting (min_samples_split) and minimum samples in the leaf nodes of the tree (min_samples_leaf). The following paragraphs explain the tuning method in an experiment. Initially, we set the vector of values for the first tuning parameter and left the remaining parameters by their default values. Then, we ran a 10-fold cross validation on the RF of each given value of the first parameter. From each iteration, we averaged the metrics scores from the corresponding cross validation results. As a result, the RF with the maximum overall metrics scores derived the first parameter’s optimum value. Subsequently, the next parameter tuning applied the optimum value(s) of the earlier parameter(s) and left the remaining parameter values by their default. Based on these experiment procedures, we ran the following experiments.

The first experiment trained the RF using the enjoyment dataset and used the default values of the parameters, except the n_estimators=100. The purpose was to gain an overview of whether the imbalance distribution of the enjoyment labels produces an imbalance confusion matrix. As a result, our concern was legitimate that the True Positive (TP) was near perfect at 0.95 and the True Negative (TN) was merely at 0.41. This confusion matrix stayed imbalanced even when we randomly tuned the RF. Consequently, this result led to the further experiment as described in the following paragraphs.

To solve this issue, we resampled the enjoyment training set using the SMOTE method because it tackles issues that exist in the other re-sampling methods, i.e. biased new samples (over sampling) and missing information (under sampling). The goal of
the resampling was to create the training dataset containing a balance distribution of its classes. The most balanced training dataset was produced when we set the SMOTE that used the five nearest neighbours and infused 89% additional minority samples (i.e. NOTFUN). The resampled enjoyment training set contained 352 NOTFUN and 353 FUN sample distributions. Then, we trained the RF that uses the same values of the tuning parameters from the former experiment. As a result, the confusion matrix is more acceptable than the former experiment. This was indicated by the TP and TN rates at 0.88 and 0.77, respectively. Based on this result, we ran another experiment to train an tune the RF using the resampled dataset.

In the last experiment, we applied the same procedures and the training dataset was the SMOTEd enjoyment dataset. As a result, the RF was optimum when it contained 474 trees, five maximum features in a tree, two minimum samples for splitting the tree branch and one minimum sample in the leaf node. Fig. 4.2 shows the averaged classification metrics of the RF predicting the enjoyment. The $x$-axis consists of the metrics we used in the training stage. Meanwhile, the $y$-axis is the normalized value from 0.0 to 1.0 representing the lowest to the highest rates, respectively. Internally, the Out Of Bag (OOB) scored 0.83 and about $\pm 0.005$ standard errors. On the confusion matrix, the TP and TN were 0.89 and 0.81 respectively, and the other metrics were also above 0.8 rates which indicated the RF can appropriately predict the enjoyment of a player via the game data. In other words, the assessment of a player’s enjoyment can be automatically and confidently performed during a game session without interrupting the gaming immersion.

From the graph we can also see the standard errors, such as TP and TN errors
that are $\pm 0.05$ and $\pm 0.09$, respectively. In order to analyse this phenomena we collected test samples from the CV iterations. Then, we processed an in-depth analysis concerning the feature characteristics between the correctly predicted samples versus error samples within each cross-validated prediction model. So, we assigned each feature with the rank of importance based on the RF importance rate (see Table 4.3). This rank helped the order of analysing the features that caused the error. The observations were manually done and we summarise of the prediction results in the following list:

1. Samples with true prediction of a NOTFUN (i.e. TN) contained very limited rewarding/positive activities, especially the feature $\text{bondReadTimeAvg}$ was close to zero, a low $\text{XP}$ score and the feature $\text{bondOK}$ was near zero, but there were higher measures on failures, e.g. $\text{bondFail}$, $\text{battleFail}$ and $\text{totalLivesLost}$.

2. Samples with correct Enjoyment prediction (i.e. TP) were driven by positive activities, especially a longer time reading the compound information $\text{bondReadTimeAvg}$, a high $\text{XP}$ score, a high $\text{bondOK}$ and often wandering the maze more than average (i.e. visible from the features $\text{movesOK}$, $\text{shitHit}$ and $\text{battleOK}$); however, they produced fewer counts in negative activities such as failed bond, failed battle or lives lost.

3. Many game situations were falsely expressed as enjoyable moments (known as False Positive or FP), despite the fact that players did not experience the game sufficiently. They were also lacking of moves, with too few positive outcomes and the main objectives were not achieved (e.g. total successful bond was close to zero and a Defeat due to running out of time). The aforementioned samples should have been labelled as NOTFUN. And, earlier we have mentioned that we can easily spot the extreme input values in these samples. We suspect these samples were produced by players who spent a very short length of time with the SEG, which could be due to them having been overlooked in the training game session. Hence, they set the standard of enjoyment lower than average. Or, their attention might might have been diverted towards something else other than the logged game data itself, such as attractive game graphics, the engaging ’new’ game play or the ’first-time’ gamers (novices) who became over-excited.

4. Some samples also incorrectly frustrated or bored the players. Most of these samples showed very high positive and negative play-log features. For instance, many occasions contained high accumulations of successful bond and sufficient
4.5. PREDICTION OF EXPERIENCES

Table 4.3: Features with Importance Rank for Predicting Enjoyment.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Feature</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bondReadTimeAvg (0.1151)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>XP (0.0973)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>bondOK (0.0791)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>movesOK (0.0696)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>bulletsTaken (0.0602)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>shotHit (0.0582)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>complexityMeasure (0.0546)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>bondFail (0.0515)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>totalLivesLost (0.0449)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>battleOK (0.0431)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>numberOfBullets (0.0415)</td>
<td></td>
</tr>
</tbody>
</table>

It becomes clear that enjoyment data obtained from a survey inevitably contains noises. Because enjoyment is a subjective expression in that different players have certain standards of so-called enjoyment. To minimize errors, the prominent solution is by administering a preliminary survey to the candidate participants by introducing the standard of enjoyment in the Flow Theory [36], game-Flow model [54], Flow model for serious games [101] or e-Game Flow model [27]. The content introduced in the preliminary survey should fit the SEG used for the data collection. In addition, the survey should also ensure the diversity of the players, so that the machine learning algorithm can generalize the enjoyment model more accurately.
4.5.1.1 Important Features

This subsection explains in more detail about the important features that were formed from the importance rank of the random forest (Table 4.3).

The ten top features contributed 67% for the prediction of enjoyment surpassed by the average time of a player reading compound information (\textit{bondReadTimeAvg}) by 11.5%. Many players enjoyed the game session when their memory of the correct atom was reinforced by spending a few seconds reading the pop-up information regarding the compound. Thus, it allowed them to recollect the correct atom collection repeatedly (\textit{bondOK}). In other words, the feature \textit{bondReadTimeAvg} could act as the initiator of good progress in the game (represented by feature \textit{XP}) that made the game exciting. Accordingly, it was not surprising that features \textit{XP} and \textit{bondOK} were the next important features by 9.7% and 7.9%, respectively.

Those actions resulting in good progress have a direct relationship with the total of successful moves (\textit{movesOK}), \textit{bulletsTaken}, \textit{shotHit} and \textit{complexityMeasure}. When a player recognised the bonding atom, the decision making became straightforward. This type of player navigated the avatar (\textit{movesOK}) confidently and he/she shot wrong atoms and enemies accurately (requiring sufficient \textit{bulletsTaken} and producing \textit{shotHit}), which were regarded as obstacles for reaching the correct atoms. In fact, the maze’s complexity (\textit{complexityMeasure}) has some effect on these features in that it also acted as one of the obstacles to achieving the game’s goal. Therefore, the aforementioned four features were the 4th to 7th most important features.

On the other hand, players who had no prior knowledge about the compound and the involved atoms might adopt a trial-error strategy collecting different atoms until the correct one is found. Such trials on various atoms were shown by a feature named \textit{bondFail} which corresponds with the curiosity of the players within the game that, to some extent, excited them. As a consequence, the death rate was also increased due to each additional failed bond (\textit{totalLivesLost}).

The last feature in the top ten was the total of successful battles shown by feature \textit{battleOK}. It revealed that some players were not in favour of fighting enemies, but alternatively moved the avatar through the \textit{loop paths} to avoid collision with them. In addition, this strategy produced a significant effect on lowering the importance of the \textit{battleFail} feature. Because, the game rule itself accentuated the compound thus forming rather than fighting the enemy. For that reason, most players have considered the battle was less significant for the enjoyment prediction.

Unsurprisingly, most features outside the top ten ranking had an importance level
4.5. PREDICTION OF EXPERIENCES

of less than 4% dominated by the game content descriptions, i.e.: numberOfBullets, numberOfEnemy, numberOfCorners, numberOfCrosses, enemyType, numberOfPaths and numberOfDeadEnd, measuring a total of 21.7% importance for the enjoyment prediction. This was due to the fact that most players were not fully recognised the game content descriptively; rather, they prefer to experience the game content or accepting the challenge offered. For instance: the player will not count the number of crosses in the game; instead, he/she prefers using the cross-paths as facilities to avoid a chasing-enemy. Meanwhile, movesFail was rather ignored due to the sole consequence it produced, i.e the enemies move one step ahead.

Subsequently, the least three features were rated just above 5% combined. The number of potions taken in the game was an alternative strategy for regaining lives. In fact, accumulating xp-bars seemed more exciting for most players because it involved many different activities, such as accurate shots, colliding weak enemies and collecting correct atoms. Meanwhile, the end result of game stages became somewhat ignored due to the lack of follow-up scenario concerning that game stage’s result. In fact, each game stage was regarded as an independent game. If the games were correlated with each other, such as via a story line, it might have entertained the players even more. Lastly, the shotMiss feature (total of inaccurate shots) failed to entertain the players because of the abundant ammunition the players can collect. It allowed players making a lot of shots that missed become less apparent for their enjoyment.

These ranked features imply that an information seeking (i.e. bondReadTimeAvg) becomes prominent yet exciting to perform in the game session, especially for Explorer players [109]. They prefer to dig deep in to the game’s events to satisfy their curiosity. In that, such an educational game player should paramount the curiosity of solving the learning tasks. Yet, the produced game plays encourage players to interact with the game more. Thus, features ranked in the second to the seventh are relevant to achiever-type who enjoy the game via various in-game achievements [109]. Killer-type players should also find a privilege in our educational game since they can kill the enemies more frequently than the total enemies. Because the killed enemies are not removed from the game, instead, they respawned back to their home. Unfortunately, socialisers may not find this educational game exciting for them because there not exist game plays to socialise either with other players or NPC.

We need to clarify that these features are rated based on their enjoyment reports
under 4-AFC questions. The questions were designed to compare the (general) enjoyment of a couple game session. Hence, the reports are lacking clarity in which different types of player can express their enjoyment specifically such as via e-game flow questionnaire [27]. However, this type of comprehensive reporting requires prepared players. Such players must have a uniform perception of each questionnaire items, such as via workshop. As a result, enjoyment dataset can be more specific for each type of player as well as it rectifying the enjoyment model to each player category.

4.5.1.2 Enjoyment Classifier with Reduced Set of Features

In the previous section, we analysed the important features produced by the random forest. Based on that, we experimented developing RF using reduced sets of features. The RF was configured by the optimal parameter values that produced the highest classification metrics. Our purpose in this experiment is to find out whether we can build a more efficient classifier by using a smaller number of features. In the experiment, we provided an ordered list of features based on the importance rank (see Table 4.3). Then, we built an RF using a set of features from the list. The set of features were iteratively reduced by removing the least important one from the list. Within each iteration of the RF generation we measured its classification performance, which included AUC, FScore, Precision, Recall and Specificity under the 10-fold cross validation.
4.5. PREDICTION OF EXPERIENCES

Table 4.4: Learning Classes.

<table>
<thead>
<tr>
<th></th>
<th>(L_0 = 0)</th>
<th>(L_1 = 0)</th>
<th>(L_1 = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(L_0 = 0)</td>
<td>LP0</td>
<td>LP1</td>
<td></td>
</tr>
<tr>
<td>(L_0 = 1)</td>
<td>N/A</td>
<td></td>
<td>LP2</td>
</tr>
</tbody>
</table>

Overall, the classification performed quite steadily (with respect to the AUC and F-score) when the RF was trained using 17 to 22 most important features. Then, the performances slowly decreased until the top 12 features were used in the RF. At the same time, a significant drop of classification rates was produced when the RF was trained using less than the 12 most important features. Let us say, a 0.8 threshold value for each metric determined an acceptable performance, then, the trained RF inputting 17 features was the most efficient yet effective way to predict the enjoyment. In this case, the RF excluded the five least important features as follows: `enemyType`, `numberOfDeadEnd`, `potionsTaken`, `result` and `shotMiss`. The last two paragraphs in the previous subsection clearly explained the reasons these features were ignorable for predicting enjoyment.

Given this result of the experiment, we can conclude that the predictive model as an assessment tool for the SEG can accurately predict the player’s enjoyment via game data. Moreover, it still as confident as we expected when the predictive model of enjoyment uses 17 important features. Thus, there will be no interruption of the game sessions of the players.

4.5.2 Prediction of Learning Performance

From the survey we collected, the LP dataset containing 90 samples was labelled as LP0, 219 samples reported as LP1, and 231 samples had LP2 labels. Based on this class distribution, we ran the same experiments wherein configurations and descriptions were relatively the same when developing the predictive model of enjoyment. Except, we applied the one-against-all performance metrics due to the three classes in the output.

Given the original dataset, we initiated an experiment to observe the imbalance effect on the area under Receiver Operating Characteristics (ROC) curve (AUC) of a classification against the remaining classification performances. The experiment involves a Random Forest with 500 trees and the other parameters are set to default. The result measured AUC of 89%, 70% and 67% for LP0, LP1 and LP2, respectively. Such results proved the imbalanced distribution of the LP classes did not negatively impact
the classification. However, the classifications of LP1 and LP2 were a bit under performed. Therefore, we tuned the RF by iterative search within each parameter. As a result, the classifications of LP1 and LP2 improved to 73% and 70%, respectively, by employing 249 trees, 3 maximum features, 12 minimum samples and 1 minimum leaf of the RF.

According to these performances, the RF’s LP classification was acceptable as indicated by AUC scores above 0.7. In other words, the predictions were above the random guess (i.e. AUC = 0.5). However, we were a bit concerned with the LP1 and LP2 classifications. We suspected there were some inseparable values of the features between LP1 and LP2 samples, such as the rates of correct bond (\textit{bondOK}) and accurate shots (\textit{shotHit}). We understand that these features were too general and needed to be specified more descriptively, as explained in the following.

First, the logging mechanism captured details of actions. But the dataset generation was first prioritising the smaller number of input features by generalising some detailed actions into a higher level play-log feature. Therefore, the resulting classification of LP1 and LP2 urged us to use the raw log data in some of the high-level actions. For instance, we divided the \textit{shotHit} feature into \textit{shotHitUnbondingAtom}, \textit{shotHitBondingAtom} and \textit{shotHitEnemy}. We expected that these more detailed features can distinguish LP1 and LP2 better because shooting actions act as the strategy to opening paths towards the correct bond. For instance, the high rate of shots at the incorrect atoms could be driven by the player’s initial knowledge about the atoms that form the chemical compound; thus, a LP2 category should be given. On the other hand, if the player has high counts on both \textit{shotHitUnbondingAtom} and \textit{shotHitBondingAtom} it could reflect an exploratory behaviour that implemented trials and errors to identify the correct and incorrect atoms in the game stage. Hence, we interpreted this as one of the learning effort and highly related with the LP1 class.

Second, the raw play-log data were stored as a sequence of actions in a game session. We did not create play-log data as a sequence-based dataset because it contained abundant input features; thus, it will significantly increased the complexity of the classification. Hence, given that the LP1 and LP2 separation underperformed, we partly utilised the sequence of the raw play-log to improve the classification while maintaining a low-dimensional input vector. With respect to the goal of the game (i.e. collecting correct atoms), we partitioned the play-log into two groups: \textbf{before} and \textbf{after} the first collected correct bond \textit{bondOK}, denoted by $*_0$ and $*_1$, respectively. The asterisk indicated the partitioned feature name. Based on our knowledge about the game
4.5. PREDICTION OF EXPERIENCES

As a consequence, the training dataset dimension grew to 30 independent variables. In addition, we added 67 samples that were discarded due to inconsistent labelling by the participants by treating them as samples with missing labels. Then, we manually re-labelled them based on our believe given the corresponding play-log values. As a result, we obtained 607 training samples for training the predictive model of LP. See Fig.4.4 for the distribution of the LP classes. Under 10-fold cross validation, we tuned the RF’s parameters based on the AUC, F-1 score and OOB metrics because they produce robust performance measurements, although the class distribution was imbalanced.

Consequently, the RF classification improved significantly using 66 trees, 8 maximum features, 7 minimum samples and 1 minimum leaf. This was indicated by the overall metrics well above 0.8 out of 1.0 including AUC0, AUC1 and AUC2 scoring mechanics, the actions before the first correct atom collected determined whether the player knew the correct atom or not before entering the game. There were six play-log features that we partitioned as follows:

1. $movesOK \rightarrow movesOK_0 + movesOK_1$
2. $battleOK \rightarrow battleOK_0 + battleOK_1$
3. $bondFail \rightarrow bondFail_0 + bondFail_1$
4. $bondReadTimeAvg \rightarrow bondReadTimeAvg_0 + bondReadTimeAvg_1$
5. $shotHitUnbondingAtom \rightarrow shotHitUnbondingAtom_0 + shotHitUnbondingAtom_1$
6. $shotHitBondingAtom \rightarrow shotHitBondingAtom_0 + shotHitBondingAtom_1$

As a consequence, the training dataset dimension grew to 30 independent variables. In addition, we added 67 samples that were discarded due to inconsistent labelling by the participants by treating them as samples with missing labels. Then, we manually re-labelled them based on our believe given the corresponding play-log values. As a result, we obtained 607 training samples for training the predictive model of LP. See Fig.4.4 for the distribution of the LP classes. Under 10-fold cross validation, we tuned the RF’s parameters based on the AUC, F-1 score and OOB metrics because they produce robust performance measurements, although the class distribution was imbalanced.

Consequently, the RF classification improved significantly using 66 trees, 8 maximum features, 7 minimum samples and 1 minimum leaf. This was indicated by the overall metrics well above 0.8 out of 1.0 including AUC0, AUC1 and AUC2 scoring mechanics, the actions before the first correct atom collected determined whether the player knew the correct atom or not before entering the game. There were six play-log features that we partitioned as follows:

1. $movesOK \rightarrow movesOK_0 + movesOK_1$
2. $battleOK \rightarrow battleOK_0 + battleOK_1$
3. $bondFail \rightarrow bondFail_0 + bondFail_1$
4. $bondReadTimeAvg \rightarrow bondReadTimeAvg_0 + bondReadTimeAvg_1$
5. $shotHitUnbondingAtom \rightarrow shotHitUnbondingAtom_0 + shotHitUnbondingAtom_1$
6. $shotHitBondingAtom \rightarrow shotHitBondingAtom_0 + shotHitBondingAtom_1$
Figure 4.5: Optimal Performance of Random Forest Predicting LP classes.

0.98, 0.94 and 0.95, respectively. Meanwhile, the FS0, FS1 and FS2 scores were 0.88, 0.86 and 0.83, respectively, which were not too far from the OOB score of around 0.85.

The random forest accurately classified LP classes, although the fact is that learning was a complex experience. The RF obtained these acceptable classification performances provided because the SEG has a the straightforward mission objective that is closely related to the learning goal. In our case, the mission objective of the SEG was to collect an atom that forms a chemical compound and the learning goal was to recall the chemical compound. In addition, players were more prepared with their understanding about the game mechanics, and the rules obtained from the Practice Session helped them provide a reliable dataset. Moreover, unlike the previous model of a player’s knowledge where a complex relationship is seen between elements of knowledge [71], our approach highlighted the single objective in a recalling game is a good example to make the classification of the player’s learning more effective.
### Table 4.5: Features with Importance Rank for Predicting Learning Performance.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Feature</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bondReadTimeAvg_0</td>
<td>0.253</td>
</tr>
<tr>
<td>2</td>
<td>bondOK</td>
<td>0.112</td>
</tr>
<tr>
<td>3</td>
<td>bondReadTimeAvg_1</td>
<td>0.107</td>
</tr>
<tr>
<td>4</td>
<td>shotHitUnbondingAtom_1</td>
<td>0.097</td>
</tr>
<tr>
<td>5</td>
<td>bondFail_0</td>
<td>0.088</td>
</tr>
<tr>
<td>6</td>
<td>movesOK_0</td>
<td>0.073</td>
</tr>
<tr>
<td>7</td>
<td>XP</td>
<td>0.052</td>
</tr>
<tr>
<td>8</td>
<td>shotHitUnbondingAtom_0</td>
<td>0.03</td>
</tr>
<tr>
<td>9</td>
<td>complexityMeasure</td>
<td>0.019</td>
</tr>
<tr>
<td>10</td>
<td>movesOK_0</td>
<td>0.018</td>
</tr>
<tr>
<td>11</td>
<td>totalLivesLost</td>
<td>0.015</td>
</tr>
<tr>
<td>12</td>
<td>result</td>
<td>0.014</td>
</tr>
<tr>
<td>13</td>
<td>shotHitBondingAtom_0</td>
<td>0.013</td>
</tr>
<tr>
<td>14</td>
<td>bulletsTaken</td>
<td>0.011</td>
</tr>
<tr>
<td>15</td>
<td>numberOfCorners</td>
<td>0.01</td>
</tr>
<tr>
<td>16</td>
<td>bondFail_1</td>
<td>0.009</td>
</tr>
<tr>
<td>17</td>
<td>shotHitEnemy</td>
<td>0.009</td>
</tr>
<tr>
<td>18</td>
<td>numberOfEnemy</td>
<td>0.008</td>
</tr>
<tr>
<td>19</td>
<td>shotHitBondingAtom_1</td>
<td>0.008</td>
</tr>
<tr>
<td>20</td>
<td>movesFail</td>
<td>0.007</td>
</tr>
<tr>
<td>21</td>
<td>numberOfCrosses</td>
<td>0.007</td>
</tr>
<tr>
<td>22</td>
<td>numberOfBullets</td>
<td>0.007</td>
</tr>
<tr>
<td>23</td>
<td>shotMiss</td>
<td>0.006</td>
</tr>
<tr>
<td>24</td>
<td>battleOK_1</td>
<td>0.005</td>
</tr>
<tr>
<td>25</td>
<td>numberOfPaths</td>
<td>0.005</td>
</tr>
<tr>
<td>26</td>
<td>battleFail_0</td>
<td>0.004</td>
</tr>
<tr>
<td>27</td>
<td>potionsTaken</td>
<td>0.003</td>
</tr>
<tr>
<td>28</td>
<td>numberOfDeadEnd</td>
<td>0.002</td>
</tr>
<tr>
<td>29</td>
<td>enemyType</td>
<td>0.002</td>
</tr>
<tr>
<td>30</td>
<td>battleOK_0</td>
<td>0.002</td>
</tr>
</tbody>
</table>

### 4.5.2.1 Important Features

This subsection explains details about the important features that were obtained from the importance rank of the random forest (Table 4.5).

Within the top 7 of the important features, each feature contributed at least 5% dominated by bond-related features. The feature $bondReadTimeAvg_0$ affected the LP classification by 25% importance. This feature’s value indicated that players spent a bit longer reading the corresponding information of the first correct atom collected to ensure they memorised it in the first place. This is consistent with the findings that attention span plays an important role to prolong the memory of a certain object [114]. Consequently, they continued collecting the correct atom more frequently in the maze, which was shown by the second-ranked features $bondOK$. The third important feature ($bondReadTimeAvg_1$) contained information that some players needed to read again the pop-up hint to reinforce their memory of the correct atom. Once a player well-memorised the correct atom, the path directing towards the correct atom’s location was opened by shooting the incorrect atoms, represented by the feature $shotHitUnbondingAtom_1$. Then, the incorrect atoms collected before the correct atom $bondFail_0$ indicated a player did not possess the initial knowledge of the pair atom that forms the chemical bond. So, the higher the counts of this feature, the greater effect for the LP0 or LP1 categorisation. Otherwise, the player was to be categorised as LP2 because they knew the incorrect atom beforehand. Features $movesOK_1$ and $XP$ in the sixth and seventh important ones clearly showed that they separate the category of LP quite distinctively.
The more frequent the good navigations after the first bond led to higher values of XP feature and movesOK, which was the positive indicator that the player knew what he did.

Meanwhile, the 8th to 15th features had an importance rank between 3% to 1%, respectively. Only two bond-related features did not noticeably influence the classification which include shotHitUnbondingAtom0 and shotHitBondingAtom0. The shotHitUnbondingAtom0 had nearly three times more importance than feature shotHitBondingAtom0 because it corresponds with the prior knowledge of the player. A player with a relevant prior knowledge reliably shot more incorrect atoms that impeded the avatar to directly reaching the correct atom; thus leading to the LP2 category. Meanwhile, if the player did not know the correct atom, he/she will shoot the correct atom in the first place (an indication of LP0 or LP1 class).

On the other hand, the remaining 15 least important features have a less than 1% importance rank each and the sum importance was less than 10%. With respect to this fact, the RF will still confidently classify LP classes via 15 important features. Therefore, in the next subsection we will see the classification performances when the random forest uses a reduced set of features.

4.5.2.2 Learning Performance Classifier with Reduced Set of Features

In the previous subsection, the important ranks produced by the random forest classifying the learning performance showed the prominent features for RF’s LP classification. Based on that, we experimented some RFs using a reduced set of features. We configured the random forest for this experiment that consisted of 66 trees grown based on 8 maximum features, 7 minimum samples and 1 minimum leaf. The experiment’s goal was to identify a more efficient classifier with a smaller number of features. In the experiment, we used the ordered list of features based on the importance rank (see Table 4.5). Then, we grew the RF using a set of features from the list, which were iteratively reduced by removing the least important feature one at a time. Within each iteration of the RF generation, we measured its classification performance which included AUC and FScore under the 10-fold cross validation (CV).

At a glance, the graphs show there are no significant differences of classification performances among the RF that use a full set of features until the one trained with 9 important features. The small decreases do not fall below 0.93 AUC scores and 0.8 F-scores. However, there seems a degrading trend when the RF was trained using less than 13 most important features. Therefore, to ensure the classification accuracy
Figure 4.6: Learning Performance Classification using Reduced sets of Features.
to be optimal, we prefer to use 13 most important features when predicting learning performance of the player.

Based on these experiments, we have built a predictive model that classifies the LP classes confidently, even using 13 important features. When we apply this predictive model into the SEG, it will assess the player’s knowledge of an education material seamlessly without interrupting the game session of a player.
Chapter 5

Adaptation in Serious Educational Game

In this chapter, we explain our final and the most important component of the project: adaptation module. Chapter 2 explains some existing works related to our method. From there, we summarise the motivation and this is followed by our proposed approach for the adaptation. We explain our adaptation approach per sub-module and that has specific functionality for the adaptation which includes the initialization, player’s experiences prediction, game missions optimizer and game content adaptation based on the experiences of the player. Then, we simulate the adaptation using the synthetic player model.

5.1 Motivation

The assessment result of the player’s learning towards a unit of knowledge and the enjoyment in the corresponding game stage of the SEG informed the match between the challenges and the ability performed by the player. If the assessment results were negative, then an adaptation is necessary by generating a more appropriate game stage for the player. However, the SEG contains two types of content in which the characteristics are significantly different. Asking the player to hand-pick the education material is possibly feasible with a compromised interruption. The feasibility issue arises when the player has to manually search the game content of the corresponding learning task. Especially, when the size of the game content space is so large that a manual search frustrates the player and it is going to be time-consuming.
An alternative adaptation method is by generalising the mapping between an experience of the player and a pair of learning task and game content, via supervised machine learning task. Nevertheless, we argue that each player has a distinct mapping of such challenges and skill. Hence, it does not make any sense asking the player to play some game stages to build the adaptation model for him/herself. Because, there is a great chance that the player has to complete all the learning tasks in order to make the adaptation model reliable and it is very time-consuming as well. So, such an off-line learning strategy to model the adaptation is not the best choice in our case. In fact, an on-line adaptation model is more suitable because it personalises the SEG specifically for each player on the run.

In addition, the recall type of learning employed in the educational game aims to endure the memorization remains for a long time. One of the methods to achieve such an aim that matches the typical game natures is Spaced Repetition Learning (SRL) [63, 115]. The match comes from the same repetitions employed by the SRL and the game stage is the platform that encourages repetitions. Basically, the repetitions of a stage (that consist of a particular education material) are based on the current evaluation of recall of the corresponding education material. Unlike the SRL application in Duolingo [65] or other non-game learning environments(e.g., [67]), the evaluation of learning in SEG is more sophisticated than the game stage’s end result: Victory or Defeat. Instead, the predictions of learning and enjoyment should be the controller of the SRL.

On one hand, the repetition provided by the SRL in SEG should consider the player’s enjoyment besides the learning. This is because, in the game’s nature of the SEG, a repetition can become sensitive to the player’s enjoyment. For instance, an over-repeated game stage may occur due to failure in accomplishing the learning task, thus, frustration can grow rapidly. Or, a player might get bored when he recognises the completed game stage was repeated. Therefore, the predictive model (EP module) built in the previous stage of our research helps with monitoring the player’s learning and enjoyment non-intrusively when a repetition strategy (SRL) takes place. Hence, such constant monitoring of a player’s learning and enjoyment should allow the appropriate personalisation of the SEG to help the player improve his/her achievements.

On the other hand, the sequence of game stages generated for a player based on the predicted outcomes affects how he/she put in the efforts of learning and gaming. For instance, a player in a new game stage \( (g_{t+1}) \) takes the benefit out of the experiences
5.2. **PROPOSED ADAPTATION TECHNIQUE**

he/she obtained from the previous game stages \(g_{t-n}, g_{t-n+1}, \ldots, g_t\). This typical situation is similar to the condition of a player’s attitude towards an item question (e.g. in an examination) which is affected by his/her responses from the previous questions [116]. Based on this belief, the personalisation of the SEG is expected to optimise the positive experiences (learning and affective) by creating a particular sequence of game stages for each player. Thus, the **first problem** we have to tackle is personalising the sequence of game stages that optimises the achievements of the player with respect to his/her learning and enjoyment. In correspondence with that, there exist candidates for game content of a game stage who can not be overlooked because a player may respond differently towards different game content settings. A heuristic similarity search may be feasible if the search space is small. On the other hand, it would be too exhaustive if the search space is large. Therefore, the **second problem** to solve is an effective yet efficient game content search strategy based on the player’s experiences.

### 5.2 Proposed Adaptation Technique

Our adaptive SEG consists of modules presented in Fig. 5.1. It requires the organisation of SEG content by **categorising the content** into two spaces: education materials and game content, followed by a **content mapping**, which ensures that a chunk of knowledge has a unique selection of game content for all difficulty levels. Then, the adaptation module we propose here focuses on the Game Mission Optimiser (GMO) and Experience-driven Content (EdC) tackle the first and second problems, respectively, which were mentioned in the Motivation section. In brief, GMO generates a mission that consists of a sequence of learning tasks (i.e. education materials) based on the predicted learning and enjoyment of the player. GMO is inspired by the entertainment game design in which its mission often contains a number of tasks to complete. Organising objectives in such a way helps the player to recognise and control his achievements more clearly, given the limited tasks held in a mission. If there is no grouping of learning objectives, we are concerned the players may lose his/her persistence with the SEG because the players feel a massive burden in completing an unknown number of objectives. And most importantly, the GMO fits the hierarchical structure of the mapped content spaces of the SEG. Because GMO works in the top level of the content space of the SEG; namely, education materials. Then, hierarchically within each selected learning task the GMO has assigned in a mission, EdC searches the best game content. The adaptivity is continuously progressing, given
that the player’s experiences are assessed by the Experience Prediction (EP) non-intrusively via gaming data, and passes them to GMO and EdC. Iteratively, the GMO receives a pair of inputs from EP’s prediction and the played game stage(s); it then updates its knowledge about the pair and optimise the future generation of the learning tasks. Subsequently, as a search method, EdC takes into account the predicted experiences from EP and the similarity of game content candidates corresponding to a learning task. More details of the GMO and EdC are explained the subsections 5.2.3 and 5.2.4.

The main contributions of the adaptation approach presented in this project are summarized as follows: a) we propose an application of SRL in a game; b) under our proposed methods, we develop an adaptation as a proof-of-concept into the SEG, Chem Dungeon, to demonstrate the usefulness of our proposed approach; and c) we simulate this adaptation method using synthetic players because of the intense simulated experiments that human players can not handle.

In the next subsections we explain the procedures and the elements involved in the adaptive SEG. We begin with the initialisation of a fresh start of SEG game session for a player. Then, we briefly explain the EP model as the non-intrusive assessment tool consisting of the input and output descriptions. Subsequently, we elaborate the methods of GMO and EdC.

### 5.2.1 Initialization

In the first instance of the SEG, a player is required to play a practice game session at least once. The purposes of this practice game session are to accustom the player with the game rules and estimating the players gaming skills based on the score. Details about the practice game session and the score formulation are provided in Section 3.3.3.

Next, the GMO initializes a random sequence of game stages containing education
materials and game content (GC) of the first game mission. EdC searches the default GC in the difficulty category defined by the score threshold for the corresponding education material of a game stage. The default GC is the one located in the centroid of a cluster in the center of the difficulty category. Once found, the parameter numbers of each game stage are sent into the GE to generate the game stage for the player to play it. Once a game stage has been played, the EP model predicts the learning performance (LP) and enjoyment (FUN), and then feeds them into GMO. Within the iteration of a game mission, GMO learns to optimise the next game stages based on the predicted LP and FUN.

5.2.2 Experience Prediction (EP)

We have built the EP module that predicts the learning performance (LP) and FUN of a player in the completion of a game stage. The input consists of game content features ($GC_i, i = 1, \ldots, 8$) and the player’s gaming behaviour or as known as play-log ($PL_i, i = 1, \ldots, n$ and $n$ for LP and FUN are 30 and 22 respectively). Chapter 4 provides thorough explanations of the development of experiences prediction.

The FUN prediction is a binary classification problem. At the same time, the LP prediction is a three-class classification problem that indicates the player’s recalling abilities as follows:

- LP0 indicates a player who failed to recall the unit of knowledge,
- LP1 shows an improvement of recalling an unknown education material for the player,
- LP2 is a player who already knew the education material and is able to recall it perfectly during the game session.

To simplify the input for the adaptation module, we encode both experiences into six numeric experience values (denoted as $XV$).

1. LP0 and NOTFUN,
2. LP0 and FUN,
3. LP1 and NOTFUN,
4. LP1 and FUN,
5. LP2 and NOTFUN,

6. LP2 and FUN.

If necessary, the XV value is decodable into the FUN value by inverting the modulo conversion of the XV value ($Fun = |(XV \% 2) - 1|$. We can decode the LP value using the following procedures:

**Algorithm 8 Decoding LP from XV.**

- **Purpose:** Decoding the LP value from the XV,
- **Input:** A numeric XV value ($xv$),
- **Output:** A numeric LP value ($lp$),

if ($xv > 0$)\&($xv < 3$) then
  return: $lp = 0$
else
  if $xv < 5$ then
    return: $lp = 1$
  else
    return: $lp = 2$
  end if
end if

5.2.3 Game Mission Optimizer (GMO)

GMO is responsible for organising the game stages in a mission based on the predictions of learning outcome and enjoyment (i.e. passed from EP). The foundation for GMO is the game structure of SEG shown in Fig. 5.2 inspired by the entertainment game structure. This structure ensures a mission consists of a number of different game stages that helps reduce the frustration of a player due to repeated learning goals.
in a mission. If a player failed a learning task, the corresponding game stage will be
regenerated in the other mission. Specifically, the GMO sorts a mission’s game stages
based on a selected property of the embedded education materials either in ascending
or descending order. To be more precise, we need to tackle some problems in ac-
complishing the GMO goal appropriately, which the following paragraphs will explain
based on Algorithm 9 and Fig. C.1 (in Appendix).

First, we need to set the mission’s length, \( m \), (i.e. total learning tasks) so that it
should neither include too many nor too few learning tasks in a mission. This consis-
tent with the principle of Flow Theory, a player can optimally finish a task when the
challenge is within his/her capability [36]. In addition, it should be set to a value that
allows feasible memorisation of a human on new units of knowledge such as suggested
by [117, 118]. However, this problem is beyond our reach for now. So, we limit our
project to solve this problem by allowing a player to choose the mission length manu-
ally [23] or by setting a fixed number of game stages in a mission. The value of \( m \) is
then initialising the GMO. For the reason that such a fixed number of learning tasks is
important for the consistency of the input of the GMO. As a consequence, we include
this at the end of the Practice Game session as a numeric input. Since this is part of the
initialization of the SEG, we are quite sure it will not interfere with the gaming expe-
rience of the player. As part of the initialisation, count the size of input vector of the
learning tasks (\( SI \)). Then, enumerate the set of actions (\( a_n \)) for the sequencing agent
in MbM and SbS (size: \( 2 \times SI \)). This enumeration is necessary for the Reinforcement
Learning algorithm to sort the learning tasks based on an input feature in an ascending
or descending order. In that, the first half of action values \( a_n \) create an ascending order
of learning tasks. Meanwhile, the remaining values of the \( a_n \) sort the learning tasks in
descending order.

Second, once the mission length (\( m \)) is set, the GMO selects some \( m \) education ma-
terials (\( e_0, \ldots, e_{m-1} \subset E \)) for the game stages of the new mission. We call this process:
\textit{education materials sampling}. However, we prefer not to randomly sample \( e_i \) from
the available education materials \( E \). Because, a random sampling potentially makes
the progression of a player in the SEG become uncontrolled which can cause nega-
tive affective experiences to arise. For instance, a frustration due to over-repetition
of the same education material in which the player has failed recalling it in the pre-
vious game stages. Or, boredom, because of over-repetition of the recalled education
materials in the previous game stages. A study of workers’ psychology in regards to
repeated work [119] is a similar case, to some extent, due to the issue of an unmanaged
Algorithm 9 Game Mission Optimiser

1: **Purpose:** Optimising the sequence of learning tasks of a mission.
2: **Input:**
   - Learning tasks, $E$, which contain input vectors;
   - The set of properties, $I$, which forms the input vector of a learning task;
   - Predicted learning performance of a learning task from EP Module, $LP$;
   - Total learning tasks in a mission, $m$;
   - Property-driven learning tasks sequence ordering, $a_n$;
   - The type of sampling method $F$;
   - The type of sequencing agent $RL$;
3: **Output:** The ordered sequence of learning tasks in a mission;
4: $SI ← \text{length}(I)$; \hspace{1cm} ▷ Count total input variables of $E$
5: $LT ← \text{List}(row = m, col = SI)$; \hspace{1cm} ▷ Learning tasks slots.
6: $a_n, n ∈ \{0, 1, ..., (2*SI) - 1\}$; \hspace{1cm} ▷ $a_n$ is the sorting order index;
7: while !LearningTasksCompleted() do
8:   $LG ← \text{List}(row = 4, col = 3)$; \hspace{1cm} ▷
   4 groups (Recognition, Retry, Exploration and Recall), each group: \{ID, GS, GP\}
   as group’s index(0,...,3), size and proportion;
9:   $GE ← \text{List}(row = 4)$;
10:  $GE[0] ← E[LP = 1]$;
14:  $totalGS ← \text{length}(E)$; \hspace{1cm} ▷ Size of $E$.
15:  $CG ← 0$; \hspace{1cm} ▷ Candidates of $LT$.
16:  for all $g ∈ LG$ do
17:      $g[GS] ← \text{length}(GE[g[ID]])$; \hspace{1cm} ▷ Updating data in LG.
18:      $g[GP] ← (m*g[GS])/totalGS$; \hspace{1cm} ▷ Set current group size.
19:      $CG ← (CG + g[GP])$; \hspace{1cm} ▷ Count current group proportion.
20:  end for
21:  for all $g ∈ LG$ do
22:     if $g[GP] > g[GS]$ then
23:        $g[GP] ← g[GS]$;
24:     end if
25:  end for
26:  if $F == \text{PGF}$ then
27:     PGF();
28:  else
29:     MPF();
30:  end if
5.2. PROPOSED ADAPTATION TECHNIQUE

31: \( i \leftarrow 0; \)
32: \( \text{for all } e \in E \text{ do} \quad \triangleright \text{Storing learning tasks in the slots.} \)
33: \( \quad \text{for all } g \in LG \text{ do} \)
34: \( \quad \quad \text{if } g[ID] == e[ID] \text{ then} \)
35: \( \quad \quad \quad LT[i] \leftarrow e; \)
36: \( \quad \quad \quad i \leftarrow (i + 1); \)
37: \( \quad \quad \text{end if} \)
38: \( \quad \text{end for} \)
39: \( \text{end for} \)
40: \( \text{if } RL == \text{SbS then} \)
41: \( \quad \text{SbS();} \)
42: \( \text{else} \)
43: \( \quad \text{MbM();} \)
44: \( \text{end if} \)
45: \( \text{end while} \)
46: 
47: \( \text{function } \text{LEARNINGTASKS_COMPLETED}() \)
48: \( \quad \text{for all } e \in E \text{ do} \)
49: \( \quad \quad \text{if } e[LP] < 2 \text{ then} \quad \triangleright \text{LP value of learning task } e \text{ HAS NOT been recalled.} \)
50: \( \quad \quad \quad \text{return } \text{False;} \)
51: \( \quad \quad \text{end if} \)
52: \( \quad \text{end for} \)
53: \( \quad \text{return } \text{True;} \)
54: \( \text{end function} \)
55: 
56: \( \text{procedure } \text{PGF()} \)
57: \( \quad i = 0; \)
58: \( \quad d \leftarrow |m - CG|; \quad \triangleright \text{Count the difference between mission length with the total candidates} \)
59: \( \quad \text{while } CG! = m \text{ do} \)
60: \( \quad \quad d \leftarrow |m - CG|; \)
61: \( \quad \quad \text{if } CG > m \text{ then} \quad \triangleright \text{An excess of candidates.} \)
62: \( \quad \quad \quad \text{if } LG[3 - g][GP] \geq d \text{ then} \)
63: \( \quad \quad \quad \quad LG[3 - g][GP] - = d; \)
64: \( \quad \quad \quad \quad CG - = d; \)
65: \( \quad \quad \text{else} \)
66: \( \quad \quad \quad \text{if } LG[3 - g][GP] > 0 \text{ then} \)
67: \( \quad \quad \quad \quad LG[3 - g][GP] - = 1; \)
68: \( \quad \quad \quad \quad CG - = 1; \)
69: \( \quad \quad \text{end if} \)
70: \( \quad \text{end if} \)
else  \(\triangleright\) A deficit of candidates.

if \(LG[i][GP] < LG[i][GS]\) then
    if \(LG[i][GS] <= m\) then
        \(LG[i][GP] += 1;\)
        \(CG++;\)
    else
        \(LG[i][GP] += d;\)
        \(CG+= d;\)
    end if
else
    end if
end if

\(i++;\)

end while

end procedure

procedure MPF()

\(LG \leftarrow \text{Sort}(LG, key = GS, order = \text{descending});\) \(\triangleright\) Descending order of \(LG\) based on \(GS\) value.

\(G \leftarrow LG[0];\) \(\triangleright\) Store the MOST populated group to \(G\).

\(g \leftarrow LG[0];\) \(\triangleright\) Store the LEAST populated group to \(G\).

\(d \leftarrow |m - CG|;\) \(\triangleright\) Count the difference between mission length with the total candidates

if \(CG < m\) then \(\triangleright\) A deficit of candidates.

if \((g[GS] > 0) \&\& (g[GS] >= d)\) then
    \(g[GP] += d;\)
    \(CG+= d;\)
else
    \(g[GP] += 1;\)
    \(CG+= 1;\)
end if
else \(\triangleright\) An excess of candidates.

\(G[GP]--;\)
end if

end procedure

function DECODEINPUTIdORDER(a)

\(Id = a - m - 1;\)

\(Order = 0;\) \(\triangleright\) Ascending order.

if \(a > (m-1)\) then
    \(Order = 1;\) \(\triangleright\) Descending order.

\(Id = a - m - 1;\)

end if

return \{Id, Order\};
end function
5.2. PROPOSED ADAPTATION TECHNIQUE

function UPDATE STATES() 

for all $i \in 0,\ldots,(\text{length}(I) - 1)$ do 

for all $j \in 0,\ldots,(m - 2)$ do 

dEI[i][j] ← |LT[j][i] - LT[j + 1][i]|
end for 
end for 

for all $i \in 0,\ldots,(\text{length}(I) - 1)$ do 

minEI[i] ← min(dEI[i]);
maxEI[i] ← max(dEI[i]);
medEI[i] ← median(dEI[i]);
µEI[i] ← $\sum_{e=0}^{m-2}dEI[i][e] / m$
end for 

return $s_k \leftarrow \{\mu EI, \text{minEI}, \text{maxEI}, \text{medEI}\}$
end function

procedure SBS() 

θ ← 0.3;  \triangleright Set the exploration rate of the Sequencing agent.

RL ← QLearning(NeuralNet, θ);  \triangleright The Sequencing agent.

for all $k \in 0,\ldots,(m - 1)$ do  \triangleright Loop within learning tasks LT.

$a_k \leftarrow \text{RL}[a]$;  \triangleright Get the action selected by the Sequencing agent.

LastLT ← {LT[0],...,LT[k]};  \triangleright Get the past learning tasks.

TempLT ← {LT[k + 1],...,LT[m - 1]};  \triangleright Get the next learning tasks.

Id, Od ← DecodeInputIdOrder($a_k$)

TempLT ← Sort(TempLT, key = Id, order = Od)  \triangleright Sort TempLT based on $a_k$.

LT = {LastLT, TempLT};  \triangleright Renew the learning tasks order.

$s_k \leftarrow \text{UPDATE STATES}()$;

G[k] ← EdC(LT[k][ID]);  \triangleright Get game content using EdC module based on the learning task’s id.

Playlog ← Game Session;  \triangleright The game session generates playlog of the player.

LP_k ← EP(Playlog);  \triangleright EP predicts the player’s learning performance via playlog.

$r(s_k, a_k) \leftarrow (LP_{k-1} - LP_k)$;  \triangleright Update reward.

RL ← {s_k, a_k, r(s_k, a_k)};  \triangleright Sequencing agent learns the sequence of learning tasks.
end for 
end procedure
procedure MBM()

R ← 0; \>
Set the accumulated rewards.

θ ← 0.3; \>
Set the exploration rate of the Sequencing agent.

RL ← QLearning(\text{NeuralNet}, \theta); \>
The Sequencing agent.

a_k ← RL[a]; \>
Get the action selected by the Sequencing agent.

LastLT ← \{LT[0],...,LT[k]\}; \>
Get the past learning tasks.

TempLT ← \{LT[k+1],...,LT[m-1]\}; \>
Get the next learning tasks.

Id,Od ← DecodeInputIdOrder(a_k) \>
Sort TempLT based on a_k.

LT = \{LastLT,TempLT\}; \>
Renew the learning tasks order.

s ← UpdateStates() \>

for all \( k \in 0,...,(m-1) \) do

G[k] ← \text{EdC}(LT[k][ID]); \>
Get game content using EdC module based on the learning task’s id.

\text{Playlog} ← \text{Game Session}; \>
The game session generates playlog of the player.

LP_k ← \text{EP(Playlog)}; \>
EP predicts the player’s learning performance via playlog.

r_k ← (LP_{k-1} - LP_k); \>

R ← (R + r_k)

end for

end procedure
over-repeated learning. Both these negative affective experiences are the products of an inappropriate match between skill and challenge [36, 120].

In order to minimise such situations, GMO organises a mission into four game stage groups representing the important purposes of playing SEG. First, the (Recognition) group is allocated for recalling the previously recognised education materials (successful game stages: classified as LP1). Second, the Retry group ensures the player repeats recalling the previously not recalled education materials (predicted as LP0). Third, the Exploration group includes the new recalling tasks into the mission (unexplored education materials). And, the fourth group (namely: Recall) consists of the education material that the player already knew it prior to the game session (classified as LP2). Nevertheless, the first three groups are the priority in the process. If there are situations when those groups do not have sufficient members to be included, then GMO adds the Recall group’s member(s) into the mission. According to the size of each group, we choose stratified sampling (STS) without replacement. In essence, the game stage groups become the strata and no replacement of a sample ensures a unique selection of learning tasks in a mission. The proportion of game stages of each group is proportional to the population’s size. With such a grouping method, the SRL application is found in the Recognition and Recall groups. Ideally, the inclusion of Recognition group expects a player to improve his/her memorisation to LP2 category as soon as the current mission has completed. Then, the conditional inclusion of Recall group occurs when there is a deficit in at least one of the priority groups. Such a condition is potentially met after a number of missions that the memory level is assumed to have decayed to a level triggering rehearsal of the corresponding unit of knowledge.

5.2.3.1 GMO - Group Proportion Fitting

Our strategy to optimise the education materials sampling is called group proportion fitting strategy. It is a method to set the proportion of groups in a mission based on 1) the priority of groups (PGF) and 2) the most populated group (MPF).

Both fitting strategy initialises by retrieving the mission length \(m\), the learning units \(E\), an empty list \(LG\) that summarise the learning tasks groups and a \(m\)-long learning tasks slots \(LT\) for the mission. For each item in \(LG\), we create a list of learning units \(E^g\) that have \(LP\) value equals \(g\) (denoted as \(GE\)), total learning units in the group \(GS\) and the proportion of the group members \(GP\) initially 0). Then, we sort the list of grouped \(E\) inside \(GE\) based on their repetitions (ascending).

Subsequently, the group sampling prioritise to Recognition, Retry, Explore and
Recall groups to ensure the player learn each unit of knowledge at least once. Hence, we count the sum of GS of these groups \((totalGS)\). Then, we iterate within these priority groups to measure \(GP = m \times GS/totalGS\). In this iterations, we also normalise the value of \(GP\) to not exceed the group’s size \((GS)\). We repeat the same iteration to ensure the total proportion of groups \((GP)\) is equal to \(m\) in the fitting strategy. If the total group proportions \((totalGP)\) population fails to fit the mission length, a variable \(d\) counts the absolute difference between the \(m\) and \(totalGP\).

The first strategy (PGF in Fig. C.2 in the Appendix) adds any deficits based on the group priority as follows: 1) Recognition, 2) Retry, 3) Exploration and 4) Recall. Any additional proportion are prioritised to the first three groups as long as the proportion \((GP)\) is less than its size \((GS)\). In a priority group we can add the \(GP\) to the deficit \(d\) if there are sufficient members in the group. Otherwise, we increase \(GP\) iteratively in this group. There is another condition to adding \(GP\) of the next group if all the learning units in the higher priority group are assigned to the mission \((GP == GS)\). On the other hand, if an excess occurs, the order of removal starts from the stage groups: Recall, Exploration, Retry then Recognition. We prioritise the removal of \(d\) learning tasks on the least-prioritised group’s proportion \(GP\) if the \(GP >= d\). Otherwise, iterative removal of \(GP\) from the current group or the higher priority group. Once the \(totalGP == m\), we sample the top \(GP\) learning units in \(GE\) within each group to provide the learning tasks \((LT)\) of the mission. We set these additional or removal priorities based on our belief that a priority is important to a condition that helps the player accomplish the goals efficiently.

On the other hand, the most populated fitting strategy finds a group with the most members (MPF details in Fig. C.3 in the Appendix). From this group, it removes any excess until the \(GP\) proportion fits the total mission. If there is a shortage, we added \(d\) proportion to the Recall group’s \(GP\), if it has a sufficient size. Otherwise, we add the most populated group’s \(GP\) with \(d\). Then, we store the learning tasks of the mission in the vector \(LT\) by sampling the top \(GP\) learning units in \(GE\).

### 5.2.3.2 GMO - Sequencing Learning Tasks

Once a mission has fitted proportion of learning tasks, we need to form a particular order of game stages that can optimise the overall learning in the current mission. The sampled learning tasks in the mission have no typical sequence initially. Meanwhile, the order of game stages in a mission depends on the preference of the player which is driven by his/her typical learning style and gaming skills [14]. For instance,
a player may prefer subsequent game stages that consist of significantly different recalling tasks, or, another player may prefer the game stages in which the education materials are similar to some extent. However, in our case, since the education materials of the mission were sampled by groups, there is a small chance that a particular preference of education materials exists. In fact, we rather exploit the positive experiences of the past or latest played game stages in which we expect to motivate the player to do his/her best in the following Retry and Exploration game stages. To accommodate that, we allocated game stages in which the learning performances were either recognised (LP1) or recalled (LP2) in the early stages of the mission. This also ensures the application of spaced repetition learning. We can associate playing game stages in a mission to learning steps that start with the ones previously learned or the ones that will be learned optimally. Initially, we allocated the Recognition group in the early game stages and we expect they can affect the outcomes of the remaining two groups: the failed game stages and the unplayed ones. If necessary, the Recall group could also be added to the other three groups. Then, a particular sequence of game stages should be controlled to improve the experiences.

Such typical sequencing problem commonly exist in entertainment games, such as Pac-Man [121, 122] and Backgammon [123], in which Reinforcement Learning [124] algorithms were applied to train agents playing such games to reach the goal optimally. In the serious game field, Bellotti also applied reinforcement learning (RL) to personalise the tasks sequence [14]. Basically, a reinforcement learning algorithm has a lookup table that maps states and their value for each possible action. Yet, the multi-dimensional features of the education materials and their unknown relationships in the mission make such a table-based RL approach ineffective for online learning. Therefore, an experience generalization [88, 89] was introduced as an alternative to the look-up table approach. Hence, we chose Q-learning reinforcement learning with a neural network as the function approximation ($Q(s,a)$) to personalise the sequence of game stages.

According to Sutton [124], the requirements of applying reinforcement learning are the descriptions of states, actions and rewards of the problem space. In the case of a series of learning tasks, the origins of states are the statistics of learning tasks’ attributes ($EI$) and their distances. The first half of the states consists the statistics of $EI$ including the average, median, minimum and maximum values of each $EI$. These state details remain unchanged even the sequence of learning tasks alters in a mission.
Meanwhile, the second half of the states is the statistic concerning the distances between adjacent learning tasks. To understand this concept, we need to picture a series of \( m \) learning tasks (\( E^0, E^1, \ldots, E^{m-1} \)). Each learning task contains \( n - 1 \) attributes called \( EI \). We measure the absolute distance (\( dEI^e_i \)) as \( |EI^e_i - EI^{e+1}_i| \). Variable \( e \) denotes the \( e^{th} \) learning task and \( i \) refers to the \( i^{th} \) attribute of the learning task. Each vector of distances of the \( i^{th} \) attribute (\( dEI_i \)) consists of \( dEI^0_i, dEI^1_i, \ldots, dEI^{m-2}_i \). Then, from each vector of \( dEI \) we measure the statistics including the mean \( \mu EI \) (see Eq. 5.1), median \( medEI \) (i.e. the point in the centre of vector of distances), minimum \( minEI \) and maximum \( maxEI \) of distances of each \( EI \).

\[
\mu EI_i = \frac{\sum_{e=0}^{m-2} |EI^e_i - EI^{e+1}_i|}{m} \quad (5.1)
\]

If the order of the learning tasks changes, the values of \( \mu EI, medEI, minEI \) and \( maxEI \) will alter accordingly. These values indicate the preference of learning steps from the perspective of the education materials’ attributes. In our case study there are eight state details for each attribute of the learning tasks. In total, the dimension of the states \( s \) is equal to 48 as a result of eight (state details) multiplied by six \( EI \) attributes.

Meanwhile, the action \( (a) \) acts as the sequencing basis of the remaining game stages in the mission. In our case study, GMO sorts the game stages based on an \( EI \) and the sorting order: ascending or descending. In total, there are twelve (12) actions (given that total six \( EI \) multiplied by the two sorting options) available. To accommodate the RL agent’s learning, these actions are enumerated as integer values from 0 to 11. For instance, the first action (index 0) sorts the learning tasks based on atom-1-number in ascending order. Meanwhile, the last action (index 11) bases the learning tasks sequence to the total-character-symbol-2 in descending order. Each chosen action \( (a) \) creates a new state \( s' \) for the current learning episode (i.e. current mission).

Then, at each end of a game stage, the GMO-sequencer learns whether the pair of latest action and previous state is rewarded based on the difference of the LP of the previous and current game stage. We denote the LP of the previous game stage as \( LP_{t-1} \) and the current game stage as \( LP_t \). In order to formulate the reward, we enumerate LP values as 5, 10, -10, 0 for the LP2, LP1, LP0 and the unplayed ones, respectively, for the former game stage. Simultaneously, we enumerate the LP values after playing the current game stage as 11, 20 and -40 for LP2, LP1 and LP0, respectively. Thus, the immediate reward \( r(s,a) \) is the addition between the enumerated \( LP_{t-1} \) and \( LP_t \) (see
5.2. PROPOSED ADAPTATION TECHNIQUE

Given these specifications of states, actions and reward, the neural network of the RL agent estimates the Q-value. We use Pybrain’s reinforcement learning module called Neuro-fitted Q-learning (NFQ) as the RL agent. NFQ consists of input, hidden and output layers. In our case, the input and the hidden layers have nodes as many as $s$ size multiplied by total $a$. Meanwhile, the output layer has one node that spits the estimated Q-value. RL agent learns by exploring the possible states and actions iteratively. In our case, one game stage is an iteration for the RL agents to learn. At each end of a game stage, the NFQ inputs 48-long state details $s_{t-1}$ of the previous game stage and the selected action $a$ that produce the current sequence. Then, the LP difference between the previous and current game stage sets the reward value ($r(s_{t-1}, a)$) of the current learning tasks sequence. The Q-value will be computed based on this reward (see Eq. (2.21) for details). Normally, each input fed into the RL agent will update the networks accordingly; specifically, the weights ($\theta$) that connect neurons. However, NFQ starts learning given a sufficient number of samples (a sample contains a set of $s_{t-1}$, $a$ and $r(s_{t-1}, a)$). In this case, 25 percent of the samples will be the test data. This means the RL agent needs at least four samples of the game stage to update its neurons. In the subsequent game stage, the RL agent retrieves a vector of Q-values predicted by the NFQ given the previous state $s_{t-1}$. Each Q-value is an estimated value of a specific action by means of state $s_{t-1}$. Then, the RL selects an action of the highest Q-value. And the training phase repeats once a game stage finishes.

In addition, the sequence optimisation is performed using different configuration of the RL agent. The first is the agent’s learning target that defines an event enabling the agent updates its beliefs. There are a couple of strategies: 1) Stage-by-stage reinforcement learning (SbS-RL) and 2) Mission-by-mission reinforcement learning (MbM-RL). SbS-RL punishes or rewards the sequence whenever a stage has been played;
then, the order of the remaining stages in the mission is changed accordingly (see Fig. C.4 in Appendix). MbM-RL learns the sequence of a mission after all the stages have been played; then, the next mission’s stages sequence is optimised accordingly (see Fig. C.5 in Appendix). In the second configuration is the agent’s exploration rates. The range of this parameter is between 0.0 to 1.0. The lower this value, the agent’s action is more restricted to what it already knew. In contrast, the higher the exploration rate, the more various actions the agent is trying in the next game stage. Yet, we use 0.1 and 0.3 exploration rates as the common rates allowing an adequate exploitation-exploration actions by the agent.

5.2.4 Experience-driven Content (EdC)

The main functionality of EdC is searching a GC for each game stage set up on a mission from the GMO. The input of EdC is the education material’s id (EID), the predicted Fun and LP, game content specifications from the latest game stage and average game score. EdC searches the game content from the corresponding EID which must consider the uniqueness of game content when the same education material must be repeated to prevent boredom growing rapidly. In fact, we need to consider the search space in a PCG-driven game content is often too large that a heuristic search is infeasible or the timely process during a game session. This must be avoided to maintain the player’s perseverance with the game. We propose an algorithm that applies a clustering method to solve this problem. The flowchart in Fig. 5.3 applies conditional procedures to define the type of distance for searching a suitable GC in the selected pool using clustering analysis. Such a condition is set by the identity of the education material, the predicted LP and Fun, latest game score, the parameters of the last game content and the played difficulty level. Given the mapping of the SEG content, i.e. learning item and game content, searching the GC is filtered by the EID of the education material and the difficulty category that is driven by the latest Fun and LP.

First of all, EdC declares a variable G CType which can have a string representing the GC search method (i.e. Default, Similar, Different). Then, it retrieves the EID of the education material of the game stage set by the GMO. This EID is important for the initial search in the content module of the SEG. Subsequently, it identifies the difficulty category of the selected education material. If the predicted FUN is positive, it means that the PlayedGC and PlayedLevel are appropriate for the player. However, an increasing gaming ability may occur here indicated by the increasing average score of the player. So, if the average score is high (i.e. surpasses a particular threshold),
Algorithm 10 Experience-driven Content

1: **Purpose:** Optimising the search of game content.

2: **Input:**
   - Selected learning task id, $EID$;
   - Latest prediction of $Fun$ (binary) and $LP$ (0,1,2);
   - The latest $score$;
   - The latest played game content, $PlayedGC$;
   - The latest played difficulty level, $PlayedLevel$;

3: Set variable $GCTYPE$ which indicates the type of GCsearch;

4: if $Fun$ is TRUE? then
   5: if $score$ is high? then
      6: if $PlayedLevel < \text{High}$ then
         7: Increase difficulty to one level higher;
         8: SET $GCTYPE = \text{"Similar"}$;
      9: else
         10: SET $GCTYPE = \text{"Different"}$;
      11: end if
   12: else
      13: SET $GCTYPE = \text{"Similar"}$;
   14: end if
   15: else
      16: if $LP > 0$ then
         17: Select $NewLevel$ based on the latest $score$;
         18: if $PlayedLevel == \text{NewLevel}$ then
            19: SET $GCTYPE = \text{"Different"}$;
         20: else
            21: SET $GCTYPE = \text{"Default"}$;
         22: end if
         23: else
            24: if $PlayedLevel > \text{Low}$ then
               25: Reduce difficulty to one level lower;
               26: SET $GCTYPE = \text{"Similar"}$;
            27: else
               28: SET $GCTYPE = \text{"Different"}$;
            29: end if
         30: end if
   31: end if
   32: SelectGC($GCTYPE$);
then EdC increase the difficulty level if the PlayedLevel was lower than the *High* level. Subsequently, EdC must keep the GC characteristics for the following game stages (i.e. repeated or new) by selecting another unplayed GC, which is closest or *similar* to the played one. For this purpose, EdC sets \( GCType = "Similar" \). We argue that by maintaining the game content characteristics when the above conditions are met, it can maintain the enjoyment. And, should a game stage repetition be necessary for the next mission, the growth in boredom (negative FUN) can be slowed down, given the fact that the new GC is similar but it has little variation from the played one. Another case is when the PlayedLevel was maximal, then, to prevent boredom growing the GC selection falls into the most different GC from the same difficulty category.
Otherwise, if the FUN was negative, then, the decision is driven by the predicted LP. If FUN was negative and Learning was positive, assign a new difficulty level based on the latest score. Yet, if there is no change of the difficulty level, then the selection goes to the most different unplayed GC to prioritise FUN improvement. Or, if a different difficulty level occurs, then the default (unplayed) GC is preferable. In the case when the player failed to recall the objective in the latest game stage (learning prediction is LP0), the algorithm assigned a similar game content from the lower difficulty level if the latest difficulty was above Low. We expect that by setting a lower difficulty can offer more chances to succeed the next game stage. If the latest difficulty level was already Low, then, it is impossible to select a lower difficulty level. Alternatively, we can select the most different GC to prevent a negative affective occurring.

The technical problem in selecting the GC in a difficulty category is, occasionally, the abundant GC candidates. Therefore, we propose a clustering analysis in EdC prior to the game sessions. EdC short-lists the unplayed GC candidates as clusters’ centroids. Then, in a real-time game session, EdC measures the distance of the PlayedGC versus centroids to determine the best GC based on the criteria as follows: (1) similar GC is the nearest centroid, (2) different GC is the farthest centroid, and (3) default GC is the centroid of the unplayed candidates. Fig.5.4 shows how the clustering analysis is useful in EdC and Eq. (5.3) to calculate the centroid of a cluster, where $x_m$ denotes the centroid of the $m^{th}$ cluster, with $x_i$ being the $i^{th}$ GC in the pool and $n$ being the number of GC candidates.

$$x_m = \frac{\sum^n_{i=1} x_i}{n}$$  \hspace{1cm} (5.3)

### 5.3 Evaluation

Given the three modules in the proposed adaptation method, the EP has been thoroughly evaluated during the development of the non-intrusive assessment. In this Chapter, we will evaluate the adaptation using a simulation approach that involves the GMO. Meanwhile, the EP module is simulated by the virtual player and the EdC is temporarily not used for simulation efficiency. The following subsections explain the simulation design, the synthetic player description, the simulation result and the summary of experiments.
Figure 5.4: Clustering Analysis in EdC

5.3.1 Adaptation Simulation Design

First of all, we assume that a player is offered a new item to be recognised in his/her short-term memory in the first session of a game stage. Therefore, we set the mission to contain six game stages (we denote this as $s = 6$) as the middle value within the suggestions in [117] and [118]. Then, we design the goal of the simulation as to identify the effect of the adaptation on a player’s SEG memorisation accomplishment. To achieve that, we need to run $e = 12$ experiments involving a typical player and the SEG in $i = 100$ iterations each (see Fig.5.5 for the simulation design and Table.5.1 for the types of experiment). From each experiment’s iteration, we record the outcomes – which consist of the accumulated rewards of each mission ($R_m$) – and we count the total missions ($TM$) until the player has accomplished all goals (i.e. all learning goals are classified as LP2).

We formulate the accumulated reward of mission $j$ ($R_j$) as the sum of the immediate rewards ($\sum_{t=0}^{s-1} Reward_t$) of the game stages (see Eq.5.2). In a mission, we reward
the subsequent game stages from the achievements (LP scores) in the previous and current game stages. For this purpose, the values of LP are enumerated (see Section 5.2.3). The enumeration of LP scores in the previous and current game stages are different. In such ways, the RL agent can reward or punish the subsequent game stages properly according to their LP scores. Given that Reward$_t$ is the sum of LP$_{t-1}$ and LP$_t$; thus, $R_j = (0 + LP_0) + (LP'_0 + LP_1) + ... + (LP'_{s-2} + LP_{s-1})$. The notations with prime indicating that the enumerated value differs from the previous sequence, although the LP category is the same (see Eq.(5.2)). This enumeration difference occurs due to the LP position at the current game stage. Thus, $R_j$ is equal to the sum of LP$_{t-1} + LP_t$, where $t \in [1, ..., m - 1]$, plus the LP score of the first game stage enumerated as LP$_0$ (see Eq.(5.4)). Each iteration stores a list of these $R_j$ and we need them to graphically picture the progression of the player within each mission.

Once an iteration has completed, i.e. indicated by the LP2 category in all education materials, we can determine the total mission ($M$). Provided by $M$ from all iterations, we measure the Mean, Min, Max, Median and Standard Deviation of the experiment. Additionally, we also capture the graphical representations of rewards progress in each mission from four iterations of an experiment. From these graphs, we analyse the performance of the configuration set in the experiment.

$$R_j = LP_0 + \sum_{i=1}^{m-1} (LP'_{i-1} + LP_i)$$ (5.4)

In general, there are three benchmark simulations: 1) Adaptation simulation without GMO, 2) Adaptation Simulation using GMO without RL agent and 3) Adaption simulation using GMO and RL agent. Table.5.1 lists the details of the experiments. In the first benchmark, a virtual player plays the game stages which are generated without the GMO principles, we call this No-GMO-*. This benchmark consists of two experiments. First, we generate random learning goals into a mission using simple random sampling without replacement technique No-GMO-RANDOM. Second, the player plays a mission wherein a mission contains $m$ learning goals with the original order stored in the database. In the second benchmark, the scenario involves the GMO without reinforcement learning optimising the learning goals’ sequence, we identify this as GMO-NoRL-*. This second benchmark selects education materials for a mission based on their LP groups (see 5.2.3 for more details). Two experiments will be held wherein the learning goals are random (GMO-NoRL-Random) and structured (GMO-NoRL-Raw), respectively.
CHAPTER 5. ADAPTATION IN SERIOUS EDUCATIONAL GAME

Experiment outcomes:
- Mean of M₀...M₉₉
- Min of M₀...M₉₉
- Max of M₀...M₉₉
- Median of M₀...M₉₉
- Standard Deviation of M₀...M₉₉
- Graphical Representation of four (4) RI

Simulation

Experiment_0 ------ Experiment_e

Iteration_i; i = {0 ... 99}

Mission_j; j = {0 ... TM}

Stage₀ ------ Stage₁ ------ Stage₅

Reward₀ = 0 + LP₀
Reward₁ = LP₀ + LP₁
Reward₅ = LP₄ + LP₅

Figure 5.5: Hierarchy of Simulation Experiments

The last benchmark fully applies GMO principles in which an RL agent is optimising the sequence of recalling tasks in every mission, we call this benchmark GMO-RL. Within this benchmark, we will run eight experiments with different methods inside the GMO that include group fitting strategy, agent’s learning target and agent’s exploration rate.

5.3.2 The Virtual Player

Once we have the experimental design clarified, in this subsection we define the synthetic player that will thoroughly “play” those game stages in the experiments. We use a synthetic player because it has consistent responses with respect to various inputs it is receiving from the experiments and controllable simulated memory.

A simulated experiment requires a player to recall all the learning tasks completely. Then, the same learning tasks must be recalled completely again by the same player in the following simulated experiments. Once all experiments are completed, we make a comparison of the learning gain between experiments. As a requirement, the player must be of the same level of knowledge (memory) prior to each experiment. It is not feasible to get such kinds of human players whose level of knowledge are the same.
5.3. EVALUATION

Table 5.1: Simulated Adaptation Experiments

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Learning Goals Generation</th>
<th>Fitting Strategy</th>
<th>RL Agent’s Target</th>
<th>Agent’s ε − greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-GMO-Random</td>
<td>Random</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>No-GMO-Raw</td>
<td>Raw</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>GMO-NoRL-Random</td>
<td>Random</td>
<td>MPF</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>GMO-NoRL-Raw</td>
<td>Raw</td>
<td>MPF</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>GMO-RL</td>
<td>Controlled</td>
<td>PGF</td>
<td>SbS</td>
<td>0.1</td>
</tr>
<tr>
<td>GMO-RL</td>
<td>Controlled</td>
<td>PGF</td>
<td>SbS</td>
<td>0.3</td>
</tr>
<tr>
<td>GMO-RL</td>
<td>Controlled</td>
<td>PGF</td>
<td>MbM</td>
<td>0.1</td>
</tr>
<tr>
<td>GMO-RL</td>
<td>Controlled</td>
<td>PGF</td>
<td>MbM</td>
<td>0.3</td>
</tr>
<tr>
<td>GMO-RL</td>
<td>Controlled</td>
<td>MPF</td>
<td>SbS</td>
<td>0.1</td>
</tr>
<tr>
<td>GMO-RL</td>
<td>Controlled</td>
<td>MPF</td>
<td>MbM</td>
<td>0.3</td>
</tr>
<tr>
<td>GMO-RL</td>
<td>Controlled</td>
<td>MPF</td>
<td>MbM</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Alternatively, if a human player is forced to do the experiments, it is also not feasible to reset his/her memory prior to each experiment. Because, in our case, a human player’s memory tended to be affected by repeated treatments even though they are separated by time and scenario [125, 126, 127]. The fact is that even one treatment, e.g. a pre-test, can act as a score gain for a student assessment [128]. Since the same issue also existed in [14]; as a solution, a synthetic player is a reliable and consistent experiment’s target because we can actually reset its memory whenever necessary.

For simulation purposes, we created a virtual player that mimics a typical human player who takes into account the similarity of the subsequent education materials in a mission. The virtual player can receive the treatments provided by the sequence of recalling tasks, generated by the GMO, in order to update its memory of the education materials. Inside the virtual player, we simulated the memory as a list of pairs between education material’s id and the latest XP prediction. We could have simulated the memory degrade through time using the Ebbinghaus’ Forgetting Curve [129] if a simulated experiment’s iteration had taken days to complete. However, we assumed that the virtual player simulates the SEG game missions continuously in an iteration.

By of an example, if a game stage lasts about 100 seconds (including the transition time between game stages) and a mission contains six game stages, then, the mission’s time consumption lasts for about 10 minutes. Assuming an iteration consists of 200 missions to complete an iteration, that is equal to around 34 hours of time consumption. Based on that, a memory of an education material decays exponentially until 25% of
memory strength, in one condition; that is, the learning material is not rehearsed within 33 hours. Yet, in fact, there is a 6% chance an education material will be repeated in a mission of the simulation’s experiment (i.e. a mission with six learning tasks randomly sampled without replacement from the 100 education materials). Thus, in 16 missions, the chance that education materials will be repeated is about 96%, or almost definitely sampled. So, the time consumption of 16 missions is 2.7 hours and the corresponding memory will decay only around 11% (leaving the memory retention 89% strong). Based on the assumed calculations that the memory of an education material, it ensures that within 16 missions the education material will be repeated again. Thus, we made the synthetic player’s memory does not decay overtime during the simulation as for the SRL application. Instead, the SRL is partly applied by conditional inclusion of the **Recall** group.

Internally, the virtual player processes the input to update its **simulated memory** based on some procedures to simulate a learning process. Here, we define specific conditions that make the virtual player expresses the learning (LP) and Enjoyment (Fun) experiences given the played game stages. The conditions in this regard consists of the properties of the latest and the current education materials, as well as the corresponding XV that they are categorised (i.e. Unplayed, XV1, XV2, ..., XV6). As a result, the virtual player updates the XV category with respect to the current recalling task. In addition, the virtual player is configurable to make it behaves differently via the following parameters: **the education material’s attribute ID to focus** ($at$) and **the threshold distance between the attribute ID between the subsequent stages that positively affect the latest achievement** ($da$).

The value of $at$ refers to the player’s preference of a learning goal’s property during the game session. For instance, a player is more progressive when memorising a sequence of learning goals which have similar visual symbols. And, the virtual player’s $da$ being a threshold maximum distance defining the XP it will produce. If the difference between subsequent learning goals’ $at$ values is smaller than $da$, then the expected outcome of the latest game stage is closer to the highest XP value. In contrast, the greater the difference between subsequent $at$ values to the $da$, the expected outcome becomes closer to smallest XP value ($XP = 1$). Our experiments use a synthetic player configured using parameter $at = 2$ (i.e. attribute: the atomic number of the first atom) and $da = 5$ (i.e. maximum distance of $at$ between different education materials).
5.3. EVALUATION

5.3.3 Simulation Result

Based on the simulation design, we ran the experiments in Table 5.1 and recorded the outcomes from the smallest simulation’s unit, i.e. game stage, to the final result of each benchmark. We also illustrate if human players participate in the same experiments. Therefore, we can observe that the virtual player satisfies our expectations that it performs ideally whose a human player is impossible to run these experiments.

In No-GMO-Random, the 100 iterations measured averagely 162 missions for the synthetic player to finally recall all the education materials (categorised as LP2). The standard deviation of the total missions in this experiment was around 24. We captured four experiment iterations to represent the synthetic player’s performances shown in Fig.5.6. The x-axis consists of the subsequent missions during the game sessions, while the y-axis contains the accumulated rewards of each mission ($R_m$). Overall, these graphs show increasing progress of the virtual player’s performances, especially in the first half of the game session. The graphs also show that the virtual player’s alternating performances converge gradually when the missions are approaching the end. This implies that even if the missions are of random learning goals, the virtual player can still learn them completely. The obvious drawback in this simulation is the numerous missions to finish all the learning goals. This is due to the learning performances of the synthetic player increasing through repetitions. Our concern is raised if human players play such game stages randomly. Indeed, the random tasks can produce unpredictable and inconsistent experiences for human players. For instance, the learning tasks between two subsequent game stages have a significant difference of their complexity (i.e. properties). In the first stage, the player recalled the education material relatively easily, then, suddenly, the following game stage asked to recall a complex learning material. Such a significant change of game elements could disengage the human player earlier before he/she completes the numerous SEG missions. Moreover, the memory affects his performances (potentially improvements). We may see such improvements as a decreasing trend of total missions in an experiment. Hence, it becomes difficult to obtain the representative samples.

Meanwhile, in the second experiment (No-GMO-Raw) where the learning goals selection was structured, unfortunately, the simulation looped unlimitedly. We spotted the forever loop occurred in the first mission. The cause was the sequence of the retrieved learning goals (from the database table) failed to meet the virtual player’s preference to progress. As a solution, the database should have stored a specific order of learning goals, that fits the player’s preference, then, the player can progress. In
CHAPTER 5. ADAPTATION IN SERIOUS EDUCATIONAL GAME

Mission Simulation – No GMO, Mission Len: 6

![Graphs showing cumulative rewards per mission without GMO (Random Assignment)](image)

Figure 5.6: Cumulative Rewards per mission without GMO (Random Assignment) (i.e. a Mission = six game stages)

the actual implementation on human players, however, this manual ordering task of the education materials prior to release and set up per human or virtual player will not be feasible. Because, there were too much burden on the developer and/or the expert to observe various human players thoroughly to obtain some necessary information regarding the preferred sequence of learning goals. Alternatively, the positive results may be achieved if the generation and sequence of learning goals were gradually optimised to fit the player’s preference. And this alternative approach is more feasible to achieve in the following experiments. As a comparison, the first benchmark results set the baseline to observe the improvements of the virtual player’s memorisation in the next experiments.

On the other hand, if we employ a human player in this experiment, they will produce different outputs compared to the virtual player. In that, forever loop will not exist. Again, memory effect plays the important role. Therefore, the performance graph may look similar to Fig.5.6 but a fewer total mission.

We recorded the same information in the GMO-NoRL-Random and GMO-NoRL-Raw which is shown by Fig.5.7 & 5.8. From these graphs, it is clear that the virtual player performed inconsistently and there seemed no increasing progress compared to the No-GMO-*. Surprisingly, it only needs around 66 or 67 missions to have finally recalled and repeated (i.e. SRL applied) all the education materials in both types
of sequence. Specifically, if we applied a raw sequence of learning goals within the mission generation, the graph stays with the same player. This confirmed that partially applying the GMO principles accelerated the virtual player’s learning performances. The virtual player recalled all 100 education materials in at least twice as fast compared to the same player who played the games randomly without GMO. The reason behind this speedy learning was the grouping strategy in GMO. These groups ensured all categories of the education materials (i.e. Recognition, Retry, Exploration, Recall) existed within each mission. Moreover, if a group existed in the smallest population, it would guarantee one place in the mission. Based on our knowledge, a human player will act similarly and even improve with respect to his/her memorisation performances. Certainly, a human player has the ability to recognise the task he had completed previously. When he/she recognised that the unsuccessful recall is allowed be repeated, his/her memory on the corresponding material will be improved. Likewise, the human player can prolong his/her memory of the recalled education materials knowing that they are ensured a game stage in the Recognition group’s slot or Recall group in the mission. Meanwhile, the new learning tasks introduced (from the exploration or un-played group) can act as a surprise element in the mission. In the entertainment games, this surprise makes the game more engaging.

In the last benchmark (Mission Simulation - RL GMO), the same virtual player played different settings of the GMO’s Reinforcement Learning. In Section 5.2.3 in this chapter we have explained the process involved for optimising the recalling of education materials in a mission.

The first two experiments set the GMO using the priority groups fitting (PGF) strategy and SbS-RL. Both Fig.5.9 and Fig.5.10 show the accumulated rewards from different exploration rates of the RL. We can see that there were consistent performances of the virtual player shown by some consecutive missions reaching maximum rewards measured by the GMO. Yet, the player also suffered some steep drops of performances, which were shown by cumulative rewards of less than (minus) 400. Fortunately, the virtual player required a lower number of missions (mean: 48 missions) to accomplish all goals compared to the GMO-NoRL-Raw (Fig.5.8) that forces the player to complete the game in 66 missions. In addition, both graphs (regarding different exploration rates) show similar trends of performances except that using a smaller exploration rate can maintain the high performances more consistently.

When a human player plays the SEG wherein such configured GMO optimises the total missions to complete, he/she will likely benefit from the personalised sequence
CHAPTER 5. ADAPTATION IN SERIOUS EDUCATIONAL GAME

Figure 5.7: Cumulative Rewards per mission using GMO without Machine Learning, Random Sequence of Mission (i.e. a Mission = 6 game stages)

Figure 5.8: Cumulative Rewards per mission using GMO without Machine Learning, Raw Sequence of Mission (i.e. a Mission = 6 game stages)
5.3. EVALUATION

Figure 5.9: Mission Simulation - RL GMO (PGF and SbS-RL), exploration rate: 0.1

Figure 5.10: Mission Simulation - RL GMO (PGF and SbS-RL), exploration rate: 0.3
of the learning tasks. Based on our experience as a gamer, the player recognises such similarities between the subsequent learning tasks, although not descriptively noticing them. Thus, with the low exploration rates, the RL agent exploits the immediate sign of this recognition observable from the reward in each stage (SbS-RL). As a result, the player can feel more confident to complete the game stage in which the learning task is relatively the same as the latest one. Concurrently, significant drops in the performance of the human player can potentially take place in a few missions due to the negative reward produced from each game stage. Such drops, once or twice after many missions, could be due to the newly introduced learning tasks. Alternatively, it could be caused by the subsequent game stages containing learning tasks in which the distances between them exceeded the player’s preference. Since the SbS-RL agent updated its neural network at each end of a mission, such drops were immediately recovered in the coming missions, which will not frustrate the player for too long.

The next couple of experiments configured the GMO using the priority groups fitting (PGF) strategy and MbM-RL. Both Fig.5.11 and Fig.5.12 show the accumulated rewards from different exploration rates of the RL. From those graphs, the total missions required by the player were more than the former experiment. We can see that
there were inconsistent performances of the GMO shown by the ripples in the cumulative rewards of the missions. Fortunately, these missions with negative rewards were not as low as in the previous experiment. Yet, more frequent negative rewards existed and only a few above zero were elicited by the player.

Our concern for the human player from the GMO-NoRL-Raw and GMO-NoRL-Random appeared again in this experiment. This is due to the MbM-RL agent not being frequent enough optimising the sequence of the learning tasks. Hence, the human player may become frustrated and potentially withdraw him/herself earlier from completing the SEG.

We ran the next two experiments by configuring the GMO using the most populated group fitting strategy (MPF) and SbS-RL (see Fig. 5.13 and Fig. 5.14). In general, this GMO set-up caused the virtual player to produce a very similar trend of performances when the GMO was configured using PGF and SbS-RL. Especially, the early missions often caused very low performances of the virtual player. However, these experiment’s graphs show a gradual increase and some converging performances when the missions were getting closer to the end. Such a trend was similar with the No-GMO-* experiments, but had improved in total missions to complete the SEG game sessions by the virtual player. Applying this type of GMO for a human player is a better choice in that
CHAPTER 5. ADAPTATION IN SERIOUS EDUCATIONAL GAME

Figure 5.13: Mission Simulation - RL GMO (MPF and SbS-RL), exploration rate: 0.1

Figure 5.14: Mission Simulation - RL GMO (MPF and SbS-RL), exploration rate: 0.3
Table 5.2: Missions Totals of GMO Performances (each experiment runs in 100 iterations)

<table>
<thead>
<tr>
<th>Experiments</th>
<th>Mean</th>
<th>Min</th>
<th>Max</th>
<th>Median</th>
<th>Stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-GMO-Random (Slow)</td>
<td>162</td>
<td>116</td>
<td>256</td>
<td>160</td>
<td>24.905</td>
</tr>
<tr>
<td>No-GMO-Raw</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>Top (Fastest)</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>Ideal</td>
<td>34</td>
<td>34</td>
<td>34</td>
<td>34</td>
<td>0</td>
</tr>
<tr>
<td>GMO-NoRL-Random</td>
<td>67</td>
<td>65</td>
<td>70</td>
<td>68</td>
<td>1.303</td>
</tr>
<tr>
<td>GMO-NoRL-Raw</td>
<td>66</td>
<td>66</td>
<td>66</td>
<td>66</td>
<td>0</td>
</tr>
<tr>
<td>PGF, SbS-RL, 0.1</td>
<td>47</td>
<td>44</td>
<td>51</td>
<td>47</td>
<td>1.575</td>
</tr>
<tr>
<td>PGF, SbS-RL, 0.3</td>
<td>48</td>
<td>45</td>
<td>53</td>
<td>48</td>
<td>1.822</td>
</tr>
<tr>
<td>PGF, MbM-RL, 0.1</td>
<td>75</td>
<td>64</td>
<td>84</td>
<td>76</td>
<td>4.332</td>
</tr>
<tr>
<td>PGF, MbM-RL, 0.3</td>
<td>75</td>
<td>64</td>
<td>88</td>
<td>76</td>
<td>4.914</td>
</tr>
<tr>
<td>MPF, SbS-RL, 0.1</td>
<td>52</td>
<td>47</td>
<td>58</td>
<td>52</td>
<td>2.404</td>
</tr>
<tr>
<td>MPF, SbS-RL, 0.3</td>
<td>52</td>
<td>47</td>
<td>59</td>
<td>52</td>
<td>2.806</td>
</tr>
<tr>
<td>MPF, MbM-RL, 0.1</td>
<td>89</td>
<td>76</td>
<td>100</td>
<td>88</td>
<td>5.288</td>
</tr>
<tr>
<td>MPF, MbM-RL, 0.3</td>
<td>88</td>
<td>76</td>
<td>100</td>
<td>88</td>
<td>5.528</td>
</tr>
</tbody>
</table>

it potentially gives positive experiences to him/her, even more desirable than applying the GMO using PGF and SbS-RL. Because, the lowest outcomes appeared in the early missions and the next mission’s drops were gradually decreasing.

In the last two experiments, we configured the GMO using the most populated group fitting strategy and MbM-RL. We took four samples in each exploration rate. Overall, we can see that most of the early missions mark low rewards below zero. Yet, after some points in the game sessions, the performances stayed between zero and positives until the end. Unfortunately, the GMO using this configuration forces the virtual player to play more missions compared to the baseline GMO. We are thus concerned about an early disengagement by a human player when experiencing such oscillating outcomes and the lengthy missions to be completed.

### 5.3.4 Summary

In this subsection, we summarise the simulation experiments (see Table 5.2). The table’s content is based on the experiment’s 100 iterations, which include the Mean, Min, Max, Median and Standard Deviation. The SEG adaptation that generates a number of random learning goals in each mission became the **slow baseline benchmark** with the **average** total missions as 162 and a **standard deviation** of around 24. On the other hand, we estimated the **fast baseline benchmark** given the mission length =
CHAPTER 5. ADAPTATION IN SERIOUS EDUCATIONAL GAME

Figure 5.15: Mission Simulation - RL GMO (MPF and MbM-RL), exploration rate: 0.1

Figure 5.16: Mission Simulation - RL GMO (MPF and MbM-RL), exploration rate: 0.3
5.3. EVALUATION

Figure 5.17: Comparison of 'TotalMission' values

6 and the 100 learning goals. So, the fast baseline benchmark was equal to 100/6 = 16.67 or around 17 missions. This top benchmark is achievable (but rather very rarely) if the player can perform maximally and consistently in all the game stages (i.e. EP model predicts LP2 and FUN in the first session of each game stage). Instead, the more ideal performance is when a player was initially not knowing all the learning materials. Then, the first instance of each game stage make him/her recognised the education materials (categorised as LP1). In the second session of the corresponding game stage he/she manages to recall it sufficiently as LP2 category. It means, the ideal player’s performance requires two attempts for each learning goal which is equals to twice the top benchmark. Then, we compared the experiments with these baselines to identify the effectiveness and efficacy of our adaptation approach. To simplify such measurements, we normalise them by dividing the smallest baseline value with the resulting experiment values (see Fig.5.17). In that graph, we sort the normalised values in an ascending order.

In Fig.5.17 we can see that all the experiments were above the slow baseline. One of the adaptation simulation results were close enough to the ideal one – 0.72 of the ideal baseline, we can assume that the adaptation approach was working. The
stage-by-stage (SbS) reinforcement learning generated a better series of stages that allowed the synthetic player to perform more progressively compared to the mission-by-mission (MbM) RL. Moreover, the learning performances of the virtual player became more efficient when the GMO combined the SbS agent with the Priority Group Fitting (PGF) strategy. It produced results above 0.7 of the ideal one. Although, with slightly lower results when GMO applied the SbS agent and MPF strategy (0.65 of the ideal baseline). Nevertheless, there exist some frequent drops of performances, especially in early missions. Although the MbM agent needed additional missions to provide good game stages, it maintained a narrow difference between high and low performances. We are currently not concerned with this difference of rewards between PGF and MPF fitting strategies because they depend on the RL agent’s learning target. On one hand, the MbM-RL accumulates its rewards when a mission ends and reinforcement is performed after four missions have completed. This value of four was the minimum number of samples the RL agent can use for reinforcing its knowledge. On the other hand, the SbS-RL accumulates its reward in each post-game stage, reinforces its knowledge every end of a mission and also reinforces its neural network more often. As a result, SbS-RL agent updates its knowledge \(m\) times more than MbM (\(m\) refers to the mission length). Thus, it was not surprising that the GMO that activated the SbS-RL agent outperforms the MbM-RL agent. Meanwhile, very close performances exist between the GMOs that set the RL agent’s exploration rate 0.1 and 0.3. Surprisingly, the GMO application without RL agent surpassed the GMO with MbM at 0.5 versus 0.42 of the ideal result, respectively.

Our conclusion is based on that ideal situation where a virtual player was consistent with its behaviours, given the immense iterations of the experiments. Adapting the SEG using GMO that is equipped with SbS and a low exploration rate is expected to make the actual player’s learning more efficient with respect to the total missions to accomplish all the learning goals. Even applying GMO without the RL agent is considered to helping players to learn efficiently at around half of the ideal case. In the future, we would like to experiment using different synthetic players whose properties are entirely different from the current one. In such a way, we can further clarify the significance and claims regarding our adaptation approach in the SEG. We understand that the highest results found in the GMO-SbS are far from the ideal expectation. Thus, there is considerable scope for improvements. Especially, in improving the procedures in the GMO, which include the mission length optimisation and the optimised sampling strategy.
Chapter 6

Conclusion

This thesis has introduced a novel approach in developing an adaptive serious educational game. The approach was inspired by the existing techniques from education perspectives, serious games and entertainment games research fields. We will discuss the stages of our approach including the limitations and shortcomings of the research undertaken. Then, we elaborate some potential future works inspired by the findings in our research. Lastly, we finalise our research with a concluding remark.

6.1 Discussion

The research has been split into three stages for developing modules of an adaptive serious educational game. We will discuss them in the following subsections.

6.1.1 Development Framework

In Chapter 2 we have elaborated existing approaches and frameworks to develop serious educational games or game-based learning.

Therefore, our initial stage of this research highlights the importance of an SEG development framework undertaken to be efficient and effective. Instead of designing the serious game based on the characteristics of the knowledge to be transferred, we combine the knowledge space with a game content space from an existing entertainment game. And, each content space is independent of each other. We have thus presented a novel approach in our framework to develop the SEG content module.

Our framework presents an annotation and categorisation procedures for each content space. In the knowledge space, a relevant and reliable resource, and the inherent
properties of the knowledge representation are crucial to make each procedure semi-automatic by applying a rule-based approach. In the case study, only the developer was in charge developing the rules and ensuring the resulting output of each procedure was as expected. Meanwhile, our framework currently processes procedurally generated game content because the provided controllable attributes and values give us two advantages. In one, the developer is able to determine the attribute threshold that disparate the difficulty of the game content difficulty. In the second, an unsupervised machine learning can categorise the similarity automatically. Theoretically, our framework minimises the experts' involvement and the process requires a much shorter time consumption. Hence, our development framework should be more efficient than the existing approaches for the same procedure.

Once both content spaces have attributes values and are appropriately categorised, the mapping assigns an education material with a unique group of game content from each difficulty category. This mapping ensures no one game content group is assigned to two or more education materials. At this stage, the developer can follow their knowledge and assumptions specifying the mapping method based on the characteristics of the education materials and the game content clusters. The mapping is the determining task for the success of the SEG product. In the case study, the developers created the appropriate mapping method efficiently based on the inherent specifications of each content space. This is an alternative method where the developer is lacking collaboration with a relevant expert. Using the framework we elaborated in Chapter 3, we successfully developed SEG content module wherein parameter values control the generation of a game stage of the produced SEG named Chem Dungeon. A user study of human players playing the Chem Dungeon has subsequently shown reasonable results, in that it promotes an effective learning and enjoyment for the player [1].

It is worth stating that our current case study was inspired by Chem Fight (the original serious game) and it is subject to limitation. Chem Fight was designed purely for SEG, and therefore many of its game mechanics were designed by taking the properties of atoms and compounds into account. Nevertheless, Chem Dungeon emphasises a PCG-based SEG that tolerates the recall form of learning. To our knowledge, fact-based education materials (e.g., mathematics, chemistry, and physics) are the most applicable knowledge spaces under our development framework. Because reliable resources are available from schools and their attribute values have been globally verified. Moreover, they are often learned by a recalling or memorising technique that allows mapping into the game content space that, to some extent, is less complicated.
6.1. DISCUSSION

We believe that through repetitive tasks, which naturally exist in the entertainment game, the memorization of such forms of knowledge are promoted. Hence, it opens up opportunities to adjust the method for different forms of learning specified by the knowledge.

6.1.2 Non-intrusive Assessment

The core goal of this research stage is demonstrating the feasibility of assessing the player’s learning and enjoyment without interrupting the game session. The contribution from this research is the dataset for this purpose, analysis of the collected dataset and the assessment module (a.k.a Experiences Prediction) that confidently predicts the desired experiences.

We have administered a survey of human players playing the Chem Dungeon SEG. The survey recorded each player’s demographic, game data, pre- and post-game exam and the self-reported enjoyment. The players who took part in the survey were of a wide range of gaming experiences. However, just about more than half of the participants were novice players who play a two-dimensional game at least once a week, while the remaining players were somewhat familiar with such a game genre. And, only two expert players involved in the survey produced a very reliable dataset.

The survey produced a dataset for the learning performance (LP) and another dataset for the enjoyment. The inputs recorded were high-level game data such as total successful chemical bond and accurate shots. While the low-level game data from mouse and keyboards were not recorded.

The dataset with the binary-labelled Enjoyment Report has 22 input variables. From 540 re-sampled enjoyment dataset, the Random Forest (RF) predicted Enjoyment with scores above 0.8 in all the performance measures. Meanwhile, the three-class Learning Performance dataset has 30 input variables. From 607 pre-processed samples, the AUC of all LP scored averagely 0.95 and F-scores around 0.85 of the RF, predicting the LP. Such performance measures showed that we have the predictive models to non-intrusively assess the learning and enjoyment of a player confidently during a game session. The LP prediction, on one hand, was very accurate. It is due to the fact that the recall form of learning was the simplest learning method applicable in the game sessions, even in daily life. Thus, the gaming data between classes of LP were observably distinctive for the RF. On the other hand, the enjoyment prediction was not as accurate as the LP prediction since its performances were below 0.9 scores.
It could be put down to the selection of input features and the inconsistency of the players expressing their enjoyment. Such unreliability could be significant to the predictive model’s performance.

The one drawback of our non-intrusive assessment is that we currently apply it for assessing the low-level learning (i.e. recall) of a player on a single and independent education material. If relationships exist between education materials, we need to assess the player in the higher level to summarise his/her knowledge. Hence, the upcoming learning goal is based on this updated knowledge of the player where prerequisites take control.

In addition, such prediction performances within the 90 seconds of the time limit of a game session could be another contribution to this research. Meanwhile, previous research preferred the completion of a serious game scenario to finally perform an assessment instead of a time-limited assessment. We are sure that this could be an interesting research possibility for serious game research, for instance, how to predict the learning of a player in a more constraint time. Likewise, serious games can take the benefit from such research, especially for reinforcing the knowledge memory in a rapid decision-making, such as in a battle situations, an emergency medical treatment or a for natural disaster response procedures.

### 6.1.3 Adaptation

Given the mixed content spaces and the predicted experiences, we have elaborated an appropriate adaptation technique in Chapter 5 and developed an adaptation module that can work together with the other modules to create an adaptive serious educational game. In general, the adaptation works hierarchically from the mission’s adaptation to game content selection. In the top-level adaptation, the Game Mission Optimiser (GMO) optimises a sequence of learning goals (i.e. education materials) in a mission. Then, within each learning goal of the mission, the Experience-driven Content (EdC) searches game content based on the predicted experiences and the characteristics of the played game content.

We simulated the SEG on a virtual player to measure its learning improvements. As a baseline benchmark, the virtual player played the SEG without adaptation module. The same player then played the SEG with the GMO applied in the module. From both experiments, we found that the virtual player recalled and reinforced its memorization of 100 chemical compounds two times faster when using the adaptive SEG, compared to when it played the non-adaptive SEG. Unfortunately, the virtual player’s recalling
speed was far from the ideal case (i.e. the player recalled and reinforced a learning goal in one attempt). The maximum accomplishment reached by the virtual player in the adaptive environment was around 0.35 of the ideal case. In our experiment, we also found that stage-by-stage adaptation (SbS) provided a better impact for the player’s learning rather than the mission-by-mission adaptation (MbM). Overall, the simulated experiments implied that adaptive SEG encouraged the virtual player’s learning to be more efficient, proving the null hypothesis of our research was rejected.

We understand that our adaptive serious educational game is limited for the recalling purposes of a series of independent education materials. Our approach underlines the correlation between independent education materials with respect to their inherent properties. And, adaptively generating them based on these properties has a positive implication for the player’s learning. Furthermore, our adaptive SEG can be beneficial for many courses that require players to recall abundant facts, such as vocabulary, history and other basic science facts. Especially, for some users who may get bored easily when they have to acquire knowledge presented in its native form, such as text or audio. In addition, although the target players of our research were limited to adults, we can apply our adaptive SEG for younger target players as well. Because, during most of their growth period, they learn through recall such as learning basic numbers, times, the alphabet, object names and many more to name. And, applying our adaptive SEG for younger players will make their learning more enjoyable.

The first shortcoming of our adaptation method is driven by the fact that we simulated the adaptive SEG on a virtual player, wherein our observation and analysis become very restricted. In fact, there is significantly more to observe when applying the adaptive SEG on human players. The next shortcoming results from the fact that our adaptation specifies a static length of a mission throughout the game sessions. On one hand, it is computationally less expensive for the reinforcement learning agent to adapt the game stages. On the other hand, the adaptation is not maximal given that some players may learn better in a smaller size of the mission or vice versa.

6.1.4 Application of Machine Learning in Other Game Elements

Machine learning attracts a lot of attention in terms of automation of processes in the development and in-game sessions of serious games apart from the above modules in our research. An agent can control Non-player Character (NPC) behaviours either to help or to oppose the player. An opposing NPC acts with respect to the level generation of an educational game. Meanwhile, a helping one acts on its own to improve
the player’s performance within the game. A rule-based agent only works for limited situations identified by the helping NPC. Hence, the more complex the serious games, the process to create such rules becomes more arduous and it is only feasible up to a certain point. Therefore, a reinforcement learning algorithm can “crawl” those triggering situations. By this approach, the helping NPC’s knowledge is improving with the more game sessions it is recognising.

Narrative serious games also open for machine learning applications. The Interactive Storytelling Architecture for Training (ISAT) is an example [6]. ISAT involves human trainer to author the training scenario. Instead of leaving the trainer authoring the training alone, an agent can accompany him/her by providing recommendations of training scenarios. This agent must be trained prior to game sessions such as using deep learning algorithm regarding its communication ability using natural language. Similarly, a narrative storytelling game for bullying education may apply machine learning approach in their process [21]. For instance, instead of manually collecting scenarios from students, one can apply an active learning approach by providing a proper (unscripted) scenario generator and let one or some relevant experts to be the oracle in assigning responses to a scenario.

Another potential machine learning application is predicting cheaters in competitive games. One can notice a cheater from the unusual event of a game session. For instance, a change of total coins collected doubled in the subsequent events while the related actions were unchanged. This anomaly is detectable via a prediction model that handles subsequent data. One can train a neural network algorithm from samples of game action data. We can obtain this data from a survey where participants are given responsibility to either play honestly or via cheating.

There are open challenges to develop a multi-player educational game, such as educational esports. Players will have different roles to get involved in this type of game. For instance, an esports to learn basic arithmetics consists of players whose roles are positive number collector, negative number collector, operator collector, etc. In general, there are two groups of statistics which carry abundant information for machine learning to learn from. The first statistics originate from each player considering their roles, while the second one is the match data considering the competing teams and the results. At the small-scale application, a machine learning (such as naive bayes) can be used to recommend the suitable roles for a player based on his/her past game data. Here, we re-train the naive bayes each time a match completes. The label of a player input data is the match result (i.e. win or lost). Then, it outputs the recommended roles
6.2 Future Work

Research in SEG development that is attempting to mix independent content spaces is relatively new. A few reliable approaches are proposed for developing serious games for students with special needs [38], non-invasive assessment methods and relevant adaptive approaches [19, 22, 33]. These indicate there are open challenges open for research in such field.

The most notable shortcoming is the recall form of knowledge suitable for our approach. Other research proposed various methods when a different type of learning is required, such as serious games for social learning [21, 130] are categorised in the understanding cognitive domain [131]. When a more complex cognitive domain is applied into our approach, it could be interesting research to adjust the method in each development stage. For instance, the knowledge and game content spaces may need a semantic attribution and annotation. At that point, the mapping of both spaces follows the semantic attributes.

A higher-level of learning technique such as application [131] is trickier to be applied under our development framework, albeit recalling is preferable. There are three crucial steps to follow such an implementation. In the first, the education materials must be broken down into the smallest chunks or units [118]. These smallest elements of knowledge can be learned by a player via recalling strategy. Then, the second step is modelling the relationships between the chunks of knowledge as accurate as possible based on the relevant resource [100]. Finally, the strategy to deliver these chunks of knowledge in the game must be based on the modelled relationships. Because it plays an important role in allowing them to be appropriately learned by a player.

Above paragraph emphasized that any type of learning is applicable under our development framework in one condition, converting the target knowledge to the units of knowledge that can be learned through rehearsal/recalling. In addition, any genre of game content spaces such as 3-dimensional first person shooter or platformer are also suitable under our development framework. Consequently, the attribution of both content spaces should be adjusted accordingly.

Our development framework requires the developer to infuse his/her abstraction allowing the mapping of the separate spaces. Alternatively, an active learning technique
can be applied for the mapping procedure, especially when the knowledge and game content spaces are very large. The developer acts as the oracle annotating the 'good' or 'bad' mapping of some pairs between knowledge units and game content. So, instead of burdening the developers determining the mapping method, which sounds infeasible, this active learning can be more efficient and effective inspired by the approaches in [28, 29].

Our non-intrusive assessment module is accurate enough for predicting a unit of knowledge material and the enjoyment. However, correlated knowledge is currently not under our consideration. In the future, we would like to model the correlated knowledge based on the learning prediction of each unit of knowledge. Previous research demonstrated the modelling of a player’s knowledge wherein a learned material has implications for or is the prerequisite to other materials [19, 71]. We can adjust our assessment module by adding a correlated learning model of the players on top of the predicted learning of units of knowledge. We can also use the syllabus to draw the correlation between learning units [100]. The adaptation method therefore needs adjustment as well. Instead of solely using the properties-driven relationships between series of learning goals, we also involve the correlated learning model of the player to drive the content adaptation.

In the GMO simulation, we partly applied the SRL because we did not estimate the memory decay for it. The one reason was the simulation ran continuously to measure the player’s learning efficiency via the total missions required to recall all the knowledge units. In the future, we would like to simulate the whole adaptive SEG that is considering details of factors relevant to which a human player will experience. First, the simulation time space will be scaled down to allow simulated long-term sessions. For instance, one minute in the simulation represents one hour in real time. Second, we add the functionality of the virtual player in the followings. Its memory will be represented by a list of triplets composed of education material id (EID), updated XP and the memory strength. This memory strength is updated through time based on the Ebbinghaus’ Forgetting Curve [129]. There is also new attributes in the virtual player that include pausing probability (per hour unit) and the length of pause event to simulate the events other than playing SEG such as sleeping, eating, relaxing, reading. Thus, we create a simulation in which the conditions are as close as the actual sessions. Optionally, we can also add a new attribute with respect of the distraction the virtual player is prone to degrade its memorised knowledge significantly. Third,
we refine the rule of education materials sampling that is considering the estimated decay of a recalled education material. The rule is basically some thresholds of memory strength that indicate the priority of repetition. Say, we have two threshold values: 0.6 and 0.4. When a particular memory strength is less than the later threshold value, the corresponding learning material has the highest priority to be repeated.

Lastly, we would like to inspect the actual effect of our adaptive SEG versus the non-adaptive SEG on two groups of human players. One of the purposes is the same as the simulated experiments, that is, identifying the efficiency of learning of the players playing the SEG. In addition, we also want to examine their long-term memorization as one of the implications of a SEG that applies a spaced repetition learning method.
Bibliography


Appendix A

Input features for Learning Prediction

This appendix includes a table that describes the independent variables used for predicting the Learning Performance category. This list acts as the reference for Chapter 3 and Chapter 4 regarding the independent variables used.

This table consists of the name for each feature which is the given name in the dataset, a short description of the feature, type of value and the range of value.

Note:
* indicates the feature is only applicable before the first successful bond created.
** indicates the feature is only applicable after the first successful bond created.
<table>
<thead>
<tr>
<th>Input Feature</th>
<th>Description</th>
<th>Value Type</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>enemyType</td>
<td>Type of enemy movements towards the avatar.</td>
<td>Ordinal</td>
<td>Random, Simple, Smart</td>
</tr>
<tr>
<td>numberOfEnemy</td>
<td>Total enemies in the game.</td>
<td>Discreet Integer</td>
<td>1, 2, 3, 4, 5</td>
</tr>
<tr>
<td>numberOfBullets</td>
<td>Total Bullets collected by the avatar.</td>
<td>Discreet Integer</td>
<td>1, 2, 3, 4, 5</td>
</tr>
<tr>
<td>numberOfPaths</td>
<td>Total paths in the maze.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>numberOfCorners</td>
<td>Total corners in the maze.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>numberOfCrosses</td>
<td>Total intersections in the maze.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>numberOfDeadEnd</td>
<td>Total tiles defined as a dead-end path.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>complexityMeasure</td>
<td>A measure of the maze complexity.</td>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>totalNavigationOK</td>
<td>**Total successful moves of the avatar.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>shotHitUnbondingAtom</td>
<td>**Total accurate shots to the wrong atom.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>Input Feature</td>
<td>Description</td>
<td>Value Type</td>
<td>Value Range</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>shotHitBondingAtom</td>
<td>Total accurate shots to the correct atom.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>shotHitEnemy</td>
<td>Total accurate shots to the enemy.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>battleOK</td>
<td>Total successful battles: avatar vs a 'weak' enemy.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>totalNavigationFailed</td>
<td>Total failed moves e.g., hit a wall.”</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>shotMiss</td>
<td>Total inaccurate shots i.e. wall shots.”</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>battleFail</td>
<td>Total failed battle: avatar vs a 'strong' enemy.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>totalLivesLost</td>
<td>Total lives lost during a game session.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>bulletsTaken</td>
<td>Total bullets collected by the avatar.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>potionsTaken</td>
<td>Total potions collected by the avatar.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>XP</td>
<td>An experience measure = (actions^+ - actions^-)</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>input Feature</td>
<td>Description</td>
<td>Value Type</td>
<td>Value Range</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>21 result</td>
<td>The final result.</td>
<td>Ordinal</td>
<td>0 = {defeat, time up}, 1 = {victory}</td>
</tr>
<tr>
<td>22 bondOK</td>
<td>Total correct atoms collected that forms the desired chemical compound.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>23 bondReadTimeAvg</td>
<td>Average seconds reading pop-up information about the successful bond.</td>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>24 failedBond</td>
<td>*Total wrong atoms collected that do not form the desired chemical bond.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>25 p1-totalNavigationOK</td>
<td>*Total successful moves of the avatar.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>26 p1-shotHitUnbondingAtom</td>
<td>*Total accurate shots at the wrong atom.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>27 p1-shotHitBondingAtom</td>
<td>*Total accurate shots at the correct atom.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>28 p1-battleOK</td>
<td>*Total successful battles: avatar vs a 'weak' enemy.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>Input Feature</td>
<td>Description</td>
<td>Value Type</td>
<td>Value Range</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>p1-EbTime</td>
<td>*Total seconds reading pop-up information about the first successful bond.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p1-failedBond</td>
<td>*Total wrong atoms collected that do not form the desired chemical bond.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B

Game Data features for Fun Prediction

This appendix includes a table that describes the game data features used for predicting the Learning Performance category. This list act as the reference for Chapter 3 and Chapter 4 regarding the independent variables used.

This table consists of the name for each feature which is the given name in the dataset, a short description of the feature, type of value and the range of value.
<table>
<thead>
<tr>
<th>Game Data Feature</th>
<th>Description</th>
<th>Value Type</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 enemyType</td>
<td>Type of enemy movements towards the avatar.</td>
<td>Ordinal</td>
<td>Random, Simple, Smart</td>
</tr>
<tr>
<td>2 numberOfEnemy</td>
<td>Total enemies.</td>
<td>Discreet Integer</td>
<td>1, 2, 3, 4, 5</td>
</tr>
<tr>
<td>3 numberOfBullets</td>
<td>Total bullets collected by the avatar.</td>
<td>Discreet Integer</td>
<td>1, 2, 3, 4, 5</td>
</tr>
<tr>
<td>4 numberOfPaths</td>
<td>Total paths in the maze.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>5 numberOfCorners</td>
<td>Total corners in the maze.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>6 numberOfCrosses</td>
<td>Total intersections in the maze.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>7 numberOfDeadEnd</td>
<td>Total tiles defined as a dead-end path.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>8 complexityMeasure</td>
<td>A measure of how complex a maze is.</td>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>9 totalNavigationOK</td>
<td>Total successful moves of the avatar.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>Game Data Feature</td>
<td>Description</td>
<td>Value Type</td>
<td>Value Range</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>10  shotHit</td>
<td>Total accurate shots at the wrong atom, correct atom or an enemy.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>11  battleOK</td>
<td>Total successful battles: avatar vs a ‘weak’ enemy.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>12  totalNavigationFailed</td>
<td>Total failed moves e.g., hit a wall.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>13  shotMiss</td>
<td>Total inaccurate shots i.e. wall shots.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>14  battleFail</td>
<td>Total failed battle: avatar vs a ‘strong’ enemy.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>15  totalLivesLost</td>
<td>Total lives lost during a game session.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>16  bulletsTaken</td>
<td>Total bullets collected by the avatar.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>17  potionsTaken</td>
<td>Total potion collected by the avatar.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>18  XP</td>
<td>An experience measure = ((\text{actions}^+ - \text{actions}^-))</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>Game Data Feature</td>
<td>Description</td>
<td>Value Type</td>
<td>Value Range</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>19 result</td>
<td>The end result of a game session.</td>
<td>Ordinal</td>
<td>0= {defeat or time up}, 1= {victory}</td>
</tr>
<tr>
<td>20 bondOK</td>
<td>Total correct atoms collected that forms the desired chemical compound.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
<tr>
<td>21 bondReadTimeAvg</td>
<td>Average seconds reading pop-up information about the successful bond.</td>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>22 failedBond</td>
<td>Total wrong atom collected that do not form the desired chemical compound.</td>
<td>Unsigned Integer</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C

Flow Diagrams in Game Mission Optimiser
Figure C.1: Procedures inside GMO
Figure C.2: Priority Group Fitting strategy
Figure C.3: Most Populated group Fitting strategy
Figure C.4: Stage-by-Stage Sequencing Agent
Figure C.5: Mission-by-Mission Sequencing Agent

START

Initialise the sequencing agent;
Set the agent's exploration rate;
R = 0;

Agent selects an action (a_k <- a);
SORT LT based on the decoded a_k;
Update sequence states (s_k);

Update mission's reward
r(s,a) = R;
Send r(s,a) to agent;

Agent learns the sequence;

END

START

For i=0,...,m-1

Search Game Content via EdC

Selected game content (Gi)

Generate game stage (LT[i] & Gi);
Player plays the game stage;
Playlog generated;

Predict LP via playlog (EP module)

Update reward
r_i = LP_i-1 + LP_i;

Accumulate rewards
R = R + r_i