



UNIVERSITÀ DI PARMA

UNIVERSITA' DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
“TECNOLOGIE DELL'INFORMAZIONE”

CICLO 34

Towards Intelligent Serious Games:

Integrating Deep Knowledge Tracing and Transformer-based Recommendation

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Francesco Zanichelli

Dottorando: *Baha MG. Thabet*

Anni 2018/2019 – 2020/2021

Alla mia famiglia

To my FATHER who passed away last year while waiting for my PhD. degree... May God have mercy upon your soul

Contents

List of Figures	0
List of Tables	0
Abstract	0
Introduction	1
Literature Review	7
2.1 Serious Games Data Analytics	7
2.2 Serious Games frameworks	11
2.3 Knowledge Tracing (KT).....	13
2.2.1 Probabilistic Knowledge Tracing (PKT)	14
2.2.2 Deep Knowledge Tracing (DKT)	14
2.2.3 Neural Networks Models	15
2.4 Transformer-based Models.....	17
Intelligent Serious Games Model	21
3.1 Model Overview	21
3.2 iGDA A Conceptual Serious Games framework	23
3.2.1 iGEM	27
3.2.2 Dynamics.....	28
3.2.3 Achievements.....	28
3.3 Learning Analytics (LA)	29
3.3.1 LA Player’s Model.....	29
3.3.2 Learning Analytics Environments	31
3.3.3 DKT-Each Sequence Length Approach	32
3.3.4 Hybrid-Deep Knowledge Tracing (RNN-CNN).....	33
3.3.5 DKT-Missing Sequence Padding Method.....	35
3.4 Transformer-based Recommender	37
3.4.1 Spaced Repetition and Flashcards Approach	38
3.4.2 Recommendation and Filtering Context	39
3.4.3 Transformer-based Recommender Architecture	40
3.4.4 Transformer-based Recommendation Algorithm	41

3.4.5 Transformer-based Flashcards Generation Framework	42
3.5 xAPI Tracker	44
3.5.1 Experience APIs	45
Experiments and Results	46
4.1 iGDA Serious Games Framework	47
4.1.1 C++ Code Challenge Learning Outcomes Matrix	49
4.1.2 C++ Code Challenges Online Quiz.....	51
4.1.3 xAPI Integration and Learning Records Store	52
4.1.4 Results.....	53
4.1.5 Summary	58
4.2 Hybrid-Deep Knowledge Tracing.....	59
4.2.1 Dataset 1 Preprocessing for Training	59
4.2.2 Dataset: Simulated-5 (Baseline)	60
4.2.3 Model Configuration	61
4.2.4 Design of Experiments	61
4.2.5 Results.....	62
4.2.6 Summary	67
4.3 Transformer-based Paragraph Generation	68
4.3.1 Dataset 2: Programming Skills Guide	68
4.3.2 Design of Experiments	69
4.3.3 Results.....	70
4.3.4 Summary	74
4.4 Transformer-based Answer Generation.....	75
4.4.1 Dataset 3: Programming Skills Summaries	75
4.4.2 Design of Experiments	76
4.4.3 Results.....	77
4.4.4 Summary	80
4.5 Transformer-based Questions Generation.....	81
4.5.1 Dataset 4: C++ Questions/Answers	81

4.5.2 Design of Experiments	81
4.5.3 Results.....	83
4.5.4 Summary	86
4.6 Flashcards Generation Summary	87
Conclusions and Future Work	89
5.1 Conclusions	89
5.2 Future Work.....	91
5.2.1 Integration of The Intelligent Model	92
Bibliography	0

List of Figures

Figure 1. The Intelligent Serious Games (ISG) Components.....	22
Figure 2. Learning Based-Achievements Model.....	24
Figure 3. Inputs-Process-Outcomes by Garris	25
Figure 4. iGDA Framework combines three prior models	26
Figure 5. iGDA Framework	27
Figure 6. Conceptual Model Process of Learning Analytics	30
Figure 7. Learning Analytics and SG Environment.....	31
Figure 8. Deep Knowledge Tracing-based Hybrid Model	34
Figure 9. Missing Sequence Values Prediction Problem	36
Figure 10. Transformer-based Recommender	38
Figure 11. Transformer-based Recommender System Architecture	40
Figure 12. Transformer-Based Recommendation Algorithm.....	41
Figure 13. Fully Automated Flashcards Generation Framework.....	43
Figure 14. xAPI Tracker	45
Figure 15. C++ Code Challenge Serious Game	48
Figure 16. Code Challenge	49
Figure 17. C++ Online Quiz for Dataset	51
Figure 18. C++ Code Challenge Results (Skills Proficiency Distribution)	56
Figure 19. xAPI Captured Statements.....	58
Figure 20. AUC prediction performance range for all 22 sequence lengths.....	64
Figure 21. Prediction performance trend for all sequence lengths for each model	66
Figure 22. Prediction performance trend for all sequence Lengths for GRU – Hybrid GRU-CNN.....	66
Figure 23. Average Cosine Semantic Similarity Scores and Summary Ratio	78
Figure 24. Science Questions Generation Performance Trend	86
Figure 25. Proposed Activity Diagram for the C++ Code Challenge	93
Figure 26. Mock-up of Prediction and Recommendations in Action.....	93

List of Tables

Table 1. DKT-Missing Sequence Padding (MSP) Algorithm.....	36
Table 2. C++ Code Challenge Learning Outcomes Matrix	50
Table 3. xAPI Statement	52
Table 4. Example of Instructional Element and Game Elements Mapping	53
Table 5. Example of Instructional Mechanics and Game Mechanics Mapping	54
Table 6. Questions' difficulty and satisfaction.....	55
Table 7. Responders' Self-evaluated C++ Proficiency Level	55
Table 8. C++ Code Challenge Dataset.....	60
Table 9. Prediction at Each Sequence Length Input/Output.....	62
Table 10. AUC/ACC Prediction Performance (Average for all Sequence Lengths).....	63
Table 11. AUC Prediction's Performance on Each Sequence Length for C++CCH Dataset	65
Table 12. Sequence 11 prediction using 5 missing sequence padding method.....	67
Table 13. Tags Structure.....	68
Table 14. Tags Structure Example	69
Table 15. Testing sets: Prompt structure and Output	70
Table 16. Models Performance	71
Table 17. Comparison between the generated and the reference Texts	72
Table 18. Generated code samples with prefix tags.....	73
Table 19. N-gram Performance Results for The Flashcard Sample	74
Table 20. Average Performance of Generated Answers	77
Table 21. Examples of Generated Answers	79
Table 22. Performance of C++ Questions Generation.....	83
Table 23. Examples of Generated Questions.....	84
Table 24. Performance of Science Questions Generation.....	85
Table 25. Examples of Flashcards Generation.....	88

Abstract

Combining Deep Knowledge Tracing (DKT) and Transformer-based recommendation with serious games can establish an intelligent model for modeling players' knowledge state over missions and auto generating help text. This model can help players to look one or more steps ahead and predict the performance of the next missions in gameplay. Afterwards and if needed, the model enables the generation of proactive recommendations as flashcards for players to be able to complete the next mission successfully. In this research, we introduce a novel Intelligent Serious Games model (ISG) based on integrating the state-of-the-art DKT method and a fine-tuned Transformer-based recommendation component to improve players' programming skills for C++, one of the most common programming languages used in first-year computer science and engineering bachelor programs.

We propose novel hybrid prediction models for DKT, a novel Transformer-based Recommender architecture and a novel Transformer-based framework tailored to three different generation tasks. The latter aims to generate flashcards in the form of supporting paragraphs, questions, and answers. Alongside with fine-tuning GPT-2, GPT-Neo, BART and T5 models to four new programming skills datasets. Flashcards are the main tool used in the Spaced Repetition memorization method, yet there are not always available for many topics due to the high efforts required to create them whereas the Transformer-based models come to simplify the process. Our findings revealed the effectiveness of integrating the Deep Knowledge Tracing (DKT) method and the Transformer-based recommender with serious games. The results revealed that the fine-tuned Transformer-based framework models are capable to generate coherent C++ paragraphs, questions and answers inspected semantically with code examples in a fully automated process and using a single input string. Also, the proposed hybrid prediction models with a multi-layer learning approach for DKT achieved the best prediction performance among the other models. Whereas assessing the proposed DKT-Missing Sequence Padding (MSP) recursive method demonstrated its effectiveness in predicting more steps ahead with missing values in the sequences. Also, the novel approach in evaluating the DKT method based on each sequence within a fixed length enabled us to trace and investigate each knowledge state.

Keywords: intelligent serious games, deep knowledge tracing DKT, LSTM, GRU, CNN, hybrid prediction, questions and answer generation, Transformer, GPT-2, GPT-Neo, BART, T5.

CHAPTER 1

Introduction

Serious Games (SGs) are games designed to raise awareness or develop some skills in several domains. In particular, they combine learning with entertainment components to increase the motivation level and improve learning outcomes of players [1]. In the literature several data analytics approaches have been adopted in serious games. SGs data analytics aim to utilize players' interactions in improving design, learning outcomes, players' experience and decision making. However, a recent systematic literature review in [2] revealed some future directions to improve SGs research area with data analytics. The authors suggest focusing more on players as the main intended recipients; secondly, to establish an adaptive gaming experience according to players' needs and feedback; thirdly, to apply neural networks algorithms to analyze data; fourthly, to adopt a standard data format for capturing gameplay interactions to be sharable as open access.

To contribute into these future directions, it is worth to mention that Intelligent Tutoring Systems (ITS) traditionally offer adaptive learner-oriented environment. An ITS utilizes artificial intelligence [3], [4] to improve learning performance. The intelligent model in the ITS is an adaptive environment represented by knowledge tracing [5]–[10] and offers customized recommendations and feedback to comply with the learners' knowledge level. Similarly, our main objective in this research focuses on introducing an intelligent model for SGs. In SGs the intelligent model requires continuous estimation for the proficiency level of players during the gameplay to provide recommendations accordingly.

Recently, deep learning has demonstrated its potential over several domains, including Natural Language Processing (NLP) and knowledge tracing. Deep Knowledge Tracing

(DKT) [5] based on Recurrent Neural Networks (RNNs) has demonstrated excellent results for capturing hidden and long dependencies. DKT is an approach for modeling knowledge states of learners during practice to predict the future learning performance.

In practice DKT in a serious game can help players to look one or more steps ahead to predict the result of the next missions in gameplay. Accordingly, fine-tuned Transformer-based models [11] can auto generate flashcards as proactive recommendations for players to complete the next mission successfully with high confidence.

The emergence of the Transformer-based models [11] such as GPT-2 (Generative Pre-Training) [12], BERT (Bidirectional Encoder Representations from Transformers) [13], BART (Bidirectional and Auto-Regressive Transformers) [14] and T5 (Text-to-Text Transfer Transformer) [15] caused a significant breakthrough in several NLP tasks. These models are already pre-trained on corpus of vast number of web text and have the capability of additional fine-tuning to particular knowledge domains. Yet, Transformer-based NLP tasks such as text generation, questions/answers generation and summarization have proven to be very efficient to produce coherent text. These tasks can be deployed in different contexts to generate flashcards recommendations and feedback for players and learners.

Flashcards method is one of the widely spread application on the spaced repetition technique. Spaced repetition [16] is an evidence-based learning technique aims at reviewing the learning material at systematic intervals to form a long-term memory and improve information retrieval [16], [17]. Flashcards method is to label those with well-known concepts as less frequent review, while difficult or forgotten concepts are labeled with more frequent review to be shown frequently in spaced intervals.

This study introduces an intelligent model for learning programming skills in a serious game. As a matter of fact, learning programming skills can be challenging, especially for

university freshmen with no previous programming experience. In general, one of the main goals of teaching basic programming courses is to let students develop problem-solving abilities. Several studies suggested that traditional methods to teach computer programming are not appropriate for students who have difficulties in learning programming skills, whereas gaming, gamification and other related approaches can be problem-solving oriented and can help to focus on the concepts, hide the programming complexity and provide instant feedback [18]–[22].

In this research, we introduce an Intelligent Serious Games model (ISG) [23] based on combining serious games conceptual framework with the intelligent model of DKT [5] and a Transformer [11] based recommender for learning programming skills.

The main contributions of this study can be summarized as follows:

1. we introduce **a novel Intelligent Serious Games model** based on combining a novel serious games conceptual framework with the state-of-the-art DKT method and a novel Transformer-based Recommender architecture. The main purpose of the model is to look for one or more steps ahead during the gameplay and predict the result of the next missions to decide about the generation and proposal to players of guide flashcards as proactive recommendation.
2. we introduce **a novel SG conceptual framework (iGDA)** which proposed to align the learning model with the game model in the early stage of design, along with artificial intelligence techniques to provide an effective and complete learning experience.
3. we introduce **a novel Transformer-based Recommender architecture** which acts on the basis of the DKT prediction result to generate flashcard recommendations by fine-tuned Transformer-based models and perform spaced repetition filtering.

4. we introduce and evaluate **a novel Transformer-based framework to generate flashcards in a fully automated process**. The new framework combines at least three state-of-the-art transformer-based models fine-tuned on three new C++ datasets and tailored for specific generation tasks as follows:
 - A. we **fine-tune and evaluate** the state-of-the-art **GPT-2 and GPT-Neo models** on a new dataset **for generating paragraphs** in the field of programming skills, and we evaluated two key factors coherence and meaning of the generated texts:
 - we investigate the influence of the annotation tags' structure and associating the prompt text with prefix tags on the generated texts.
 - we investigate the performance trend along different contiguous sequence of N-gram overlap between the generated and the reference texts.
 - we evaluate and compare the performance of the generated paragraphs among GPT-2 and GPT-Neo models.
 - B. we **fine-tune and evaluate** the state-of-the-art **BART and T5 models** on a new dataset **for generating answers** in the field of programming skills, and we compared the performance of the two models:
 - we examine the capability and the performance of the models trained on different datasets to generate C++ answers.
 - we investigate the influence of using different generator decoding methods on the performance of the generated answers.
 - we compare difference evaluation metrics of the generated answers with respect to the summarization ratio.
 - C. we **fine-tune and evaluate** the state-of-the-art **T5 model** on a new dataset **for generating questions** in the field of programming skills.

- we investigate the performance of the model to generate C++ questions using a model trained on a science dataset; a C++ dataset; a mixed dataset with science and C++.
 - we examine the capability of the model trained on a subset of the topics to generate questions for the rest of the topics.
 - we investigate the influence of the dataset size on the performance of the generated C++ questions.
5. We propose and evaluate **a novel DKT-based hybrid prediction models with a multi-layer training approach** and investigate the influence of combining the state-of-the-art: Long Short-Term Memory (LSTM) neural network; Bidirectional LSTM (biLSTM); and the Gated Recurrent Unit (GRU) neural network; for sequential dependencies with Convolutional Neural Network (CNN) for hidden features extraction on the prediction performance.
- we introduce and evaluate **a novel method based on recursive algorithm** called **DKT-Missing Sequence Padding (MSP)** to predict more than one step ahead with possible of missing values in the sequence.
 - we assess **a novel approach in evaluating** the state-of-the-art **DKT method** based on each sequence within a fixed-length sequences of submissions to trace each knowledge state and apply the DKT method in real learning situations.
 - we investigate the impact of real sequence dependency and ordered concepts from basic to advance on the prediction performance using the real C++CCH dataset with the simulated dataset.
6. we introduce **4 new datasets**

- C++ exercise responses dataset called C++ Code Challenge (**C++CCH**) for learning programming skills. This dataset is used in the DKT method.
- A C++ textual dataset called C++ Code Guide (**C++CG**) to fine-tune GPT-2/Neo models to generate programming skills paragraphs.
- A C++ summaries dataset called C++ Summaries (**C++SUMM**) to fine-tune BART and T5 models to generate answers from given paragraphs.
- A C++ questions-answers dataset called C++ Questions/Answers (**C++QA**) to fine-tune T5 model to generate questions from given answers.

The rest of the thesis structure consists of Chapter 2 describing the state of the art, Chapter 3 introducing the ISG model and its components, Chapter 4 in which we describe and discuss the experiments and results, followed by Chapter 5 for conclusion and future work.

CHAPTER 2

Literature Review

In this Chapter we will review several aspects in serious games and deep learning. Firstly, we will discuss the SGs data analytics approaches that adopted in the literature to address the possible contributions in this study; secondly we will discuss several state-of-the-art serious games frameworks to clearly identify the SGs components and the possible integration of a knowledge tracing component with flashcard recommendations; thirdly, we will discuss the different knowledge tracing methods and we will focus on the state-of-the-art deep knowledge tracing based on RNNs and the other neural networks models; fourthly, we will highlight on some NLP tasks and review the state-of-the-art text generation models in order to introduce a model capable to auto generate flashcard recommendations for players.

2.1 Serious Games Data Analytics

Serious games are designed to improve objectives of learning and training with entertainment components [1], [2]. According to [24], [25] gameplay produces a large number of interactions which can be tracked and analyzed to extract patterns and useful information. Several SGs data analytics approaches have been adopted to evaluate and improve several aspects in serious games such as assessment, in-game behavior, game design, student profile, framework, and others. However, this research focuses on utilizing players' interactions for introducing an intelligent model of knowledge tracing to predict the future performance of players and to generate flashcard recommendations.

Therefore, we identified five aspects in SGs data analytics to focus on in this section as follows:

1. target stakeholders and the intended recipients;
2. purpose of the data analytics;
3. algorithms used in the data analytics;
4. assessing methods;
5. real-time/offline analytics.

To review the first three, Alonso-Fernández et al., [2] in 2019 have conducted an important systematic literature review on the applications of SGs data analytics. The final sample was total of 87 studies, and the authors presented significant facts and directions for the future work in this field.

Regarding the target stakeholders, the results shown that only in 7% of the studies the main intended recipients were students, while the majority considered as target designers and developers (35%), researchers (33%), and teachers (22.50%) respectively. These results uncover that, most of the previous studies have focused to help in decision making to improve game design, learning, impact of serious games, and to understand the gameplay interactions. Despite the final outcomes of these studies are intended to benefit students but focusing on them as the main stakeholder is crucial especially during the gameplay to enhance playing experience and learning achievements. On the other hand, for the purpose of the studies, the results show that, 36.8% of the studies have used data analytics for assessments followed by 31% for the in-game behaviors. These results aligned with the previous ones, in the fact of improving decision making by analyzing the impact of different assessment methods on the students' performance as well as the students' behavior and experience.

Referring to the most used algorithms and techniques, the study shows that linear models and regression are the most used methods for supervised approaches (24.50%), while correlations and clustering are used in the unsupervised methods (32.40%) while other visualization techniques were also used (35.30%). From this result, we conclude that the previous studies focused on traditional data analytics models and techniques. In fact, data availability restricts the fast advances in the serious games research area, due to the lack of open datasets as indicated in [2], thus requiring extra efforts in developing serious games to make further testing with new analytical tools and techniques. The authors in [2] concluded with a set of recommendations, some of them aligned with our research, such as:

- 1- apply more complex algorithms such as neural networks;
- 2- adopt a standard format for capturing players' interactions such as Experience Application Programming Interface (xAPI) [24];
- 3- encourage authors to share datasets as open access;
- 4- integration of assessment in the early phase of game design;
- 5- focus on student profiles, feedback, and adaptive learning experience.

Regarding the remaining aspects concerning assessment methods and where data analytics take place, two main approaches have been followed in the literature with non-real-time data analytics. In the first approach, which we call “the significant change method” the authors in [27]–[33] tried to capture and analyze the significant change in the knowledge state of the players before playing and/or during and after finishing the game. In this approach they applied the so-called pre-test and post-test assessments, while [32], [33] mixed it with some qualitative methods such as observations and interviews. The pre-test and post-test methodology is widely common in assessing learning, however, despite its effectiveness, it neglects that in serious games huge records of players'

interactions can be captured and analyzed instantly to extract useful information for players during the gameplay. The other approach in [25], [34]–[38] is a mixed methodology which aims to combine and analyze the pre-test and post-test with the captured interactions to support assessing learning and provide other information related to the players' characteristics and behavior, as well as to game design.

In conclusion, several data analytics approaches have been applied for different purposes. Most of the studies considered to use data analytics for assessments and in-game behavior. The main goal of the assessments was to assess the players' learning performance. However, in-game behavior the main goal was to discover the players' settings, characteristics and to improve the game design process. Moreover, a recent systematic literature review suggested some future directions for serious games research area such as: to target players as the main intended recipients (the scope of our research is player-oriented); to integrate assessments in the early phase of game design (we will discuss this in section 2.2); to apply more advanced techniques for data analytics such as neural networks (we will describe it in sections 2.3); to follow a standard data format such as xAPIs for capturing and sharing interactions; to focus more on feedback and adaptive learning experience (we will discuss in section 2.4 for text generation models).

To the best of our knowledge in the literature there are no serious games integrated with adaptive environment for knowledge tracing and recommendations for players. In the next section, we discuss different SGs frameworks to identify the SGs components from different point of views to address the possibility of integrating the learning model in the early stage of design with knowledge tracing and recommendation components.

2.2 Serious Games frameworks

Generally, several models and frameworks have been introduced for games. The most widely formal approach according to [39] is the Mechanics, Dynamics, and Aesthetics (MDA) [40] framework. The MDA game framework aims at linking all the game components to each other to facilitate a smooth transition from the code to content and play experience and go back again. In the MDA, the authors proposed the framework in three abstract components: Mechanics, Dynamics, and Aesthetics. Mechanics describe the main components of the game including rules, actions, control mechanisms. Dynamics are the gameplay interactions, player's inputs and considered as the run-time behavior of the mechanics. Aesthetics is the player's feeling, experience, and result during/after playing and interacting in the game. However, while the MDA framework focuses on the entertainment games [41], it has limitations and weaknesses [39] such as focusing too much on the mechanics and ignoring other design aspects. Moreover, the MDA is not suitable for gamified content and experience-oriented games.

Other scientists have tried to overcome the MDA limitations by introducing new extensions. In 2008 Winn [41] introduced an expansion for the MDA as Design, Play and Experience (DPE). The DPE provides a formal approach to design SGs and involves a language to discuss design, a methodology to analyze a design, and finally a process to design a SG. Winn replaces Mechanics with Design, Dynamics with Play, and Aesthetics with Experience and translates them into four separated layers as Learning, Storytelling, Gameplay and User experience. Furthermore, the DPE framework is grounded on the technology layer which has a major influence on the design process as an enabler or a limiter.

Robson et al in [42] have introduced Mechanics, Dynamics, and Emotions (MDE) as a new framework to focus on the gamified content, experience, and gamification design in

general. In MDE the authors classified mechanics into three components: Setup which is the game objects, Rule which is the goals and constrains, and the last is the Progression to describe the diverse types of instruments for the experience while it is happening. Dynamics are related to the players' behavior during the gameplay, and emotions are the player's state of mind and a product of how players follow the mechanics to generate dynamics.

Another extension has been introduced by Walk in [39], Design, Dynamics, and Experience (DDE). In the DDE, the authors placed the mechanics as a part of the design component to focus more on the design aspects and process such as blueprints to provide a conceptual design and interface to give more specifications about graphics, sounds and narratives. A recent model also introduced by Pendleton in [43] a Game Design Matrix (GDM). In GDM the authors paired the MDA framework with the Bloom's Taxonomy to provide a serious game design process. The core of this design process is to select and map the output dynamics with the Bloom's levels, after identifying the learning objectives and the subject matter. The GDM provides a step-by-step method to design serious games, with more details/constrains to map Bloom's levels to game dynamics.

To conclude, the MDA is a generic game framework, but it focuses on entertainment games, and it is not suitable for serious games as it lacks any learning model as well as in the other expansions such as the MDE and the DDE. Despite the other MDA expansions such as DPE and GDM are proposed for serious games, but they provide a design process rather than a framework. For instance, the GDM is more oriented to the learning model, and it introduces more specifications for mapping Bloom's levels to the game dynamics in a step-by-step process. Also, the DPE is a formal approach to design SGs by splitting the SGs into four logical layers. DPE also linking the user experience

only with the user interface ignoring the playing and learning experience. While learning experience is one of the most important outcomes, the educational effectiveness of the game increases according to the integration level of the game components with the educational elements.

Our approach in this study is to identify a generic framework for SGs rather than a detailed design process. As there are several design methodologies for software, gaming, learning and instructional design which can be adopted to comply with the serious game's needs. Also, a generic SGs framework enables us to align the outcomes-based learning model with the game. For instance, the game components can be divided into game elements and learning elements (including learning objectives and material), whereas the integration between them can be implemented during the design process. Also, dynamics can be identified as game interactions and learning activities, whereas the output of the SGs can be identified as game achievements and learning achievements.

In the next section, we discuss how we can track the knowledge state of players (which is the learning achievements) in real-time to predict their future learning performance (in the learning activities) using neural networks.

2.3 Knowledge Tracing (KT)

Modeling learners' knowledge state while practicing is a crucial task to improve learning achievements in the educational environments. Knowledge tracing according to [5]–[10] is an essential component in any intelligent tutoring system (ITS) to predict the students' performance. To model this problem, there are two main approaches in the literature, one based on probabilistic methods, and the other based on deep learning.

2.2.1 Probabilistic Knowledge Tracing (PKT)

Corbett and Anderson [6] have introduced the Bayesian Knowledge Tracing (BKT) method based on the Bayesian probability and the Hidden Markov Model (HMM). The original BKT method has been followed by several extensions such as [7], [8] and became the most popular classical probabilistic knowledge tracing method. The authors in BKT defined knowledge tracing as monitoring the student's changing knowledge state during practice to estimate with a high probability that each rule or concept is in the learned state. Also, they assumed that for each concept in the learning model there are two knowledge states: learned state or unlearned state. The transition occurs from the unlearned state to learned state through reading or applying some practice. Generally, the main task in knowledge tracing is to estimate the probability that a student will provide a correct answer in the concept c_{n+1} given that all the previous answers from c_1 to c_n , where n is equal to the questions' sequence length (number of questions).

2.2.2 Deep Knowledge Tracing (DKT)

According to [5], [9] the BKT method and its successor extensions are suffering from the difficulty of capturing the hidden dependencies between the sequence of concepts. However, Piech et al [5] have introduced a recent approach called deep knowledge tracing based on RNNs. The novel approach aims at modeling learners' knowledge state over time to predict the future learning performance for learners. The DKT method has been applied in several studies [5], [9], [10], [44]–[46] and outperformed previous PKT methods.

Despite the success of the DKT method, there are some limitations in terms of using inputs with non- fixed and unordered sequence lengths (timesteps) with neural network models [40]. Most of the previous studies used a non-fixed sequence length of students' submissions with undetermined sequence submission order due to the nature of the used

datasets. Unordered submissions with missing values could influence the performance of the model as there are explicit and implicit dependencies between each series of concepts, and next submission is dependent or partially dependent on previous submissions in the time-series problems. Also, they measured an overall prediction performance without assessing the impact of the non-fixed sequence length sizes on the performance of their models and without assessing the performance on each sequence length, which is unrealistic when we apply the model in real learning situation, and we need to trace each knowledge state. Another limitation in the prior studies is due to the absence of evaluating the DKT method on different neural network models other than the RNNs and its variants. However, other authors have focused partially on these limitations. Wang et al. in [44] have evaluated the DKT method on one exercise question to predict the next one. While in 2022, Hooshyar et al. [47] have assessed the DKT method on the CNN but they evaluated the model on only three sequence lengths out of 20 fixed sequence lengths. To the best of our knowledge no prior studies having assessed the DKT method using a hybrid neural model and on each sequence length within a fixed and ordered series of submissions, which is crucial to apply the model on real learning situation to trace each knowledge state of players.

In the next sub-section, we discuss different neural network models including RNN, its limitations, RNN variants, CNN in order to introduce DKT with hybrid prediction models.

2.2.3 Neural Networks Models

RNNs have been successfully applied for modeling time series data and sequences. The simple architecture of the RNNs consists of three layers: input layer, hidden state layer and output layer. The RNNs have the form of repeating modules receive a new input with feedback from the recent last input. In fact, inputs and outputs are dependent, the prior inputs influence the current input and output using a short-term memory within the

sequence and so on. During the training process, a backpropagation algorithm propagates backward from the output layer to calculate the error gradients and adjusts the weights up or down to decrease the error. Sometimes the error gradients often get smaller until approach to zero and leave the weights unchanged which cause the so-called vanishing gradient problem. Therefore, the simple RNNs with a short-term memory are not suitable to manage long term dependencies modeling [5], [9].

LSTM was introduced by Hochreiter and Schmidhuber [48] to address the problem of vanishing gradient and to manage long term dependencies. LSTM is a special type of RNNs with special memory cells architecture and forward information flow to manage sequences from past to future. This architecture enables the RNNs to remember inputs over a lengthy period of time. The repeating modules in LSTM have different structure than the RNNs. In LSTM, instead of having a single layer like in RNNs, there are three gates called input, output and forget gate that decide which information to remember or to forget. Another type of the LSTM is biLSTM which has another layer to manage the backward information flow from future to past [49].

GRU was introduced by Cho et al. [50] to manage long term dependencies problems. GRU is similar to LSTM but with simple architecture, faster and more efficient on small datasets [51]. In GRU the repeating modules have two gates only update and reset gates without having separated cell states. The update gate is responsible for determining which previous information needs to be passed along the next state, whereas the reset gate decides which information is needed to forget from the previous state.

CNN is a deep feed-forward artificial neural network and one of the most popular architectures of deep learning for feature extraction, computer vision and classification problems [52], [53]. CNN consists of three layers: input layer, hidden state layer and output layer and has a grid-like topology in processing data. Indeed, CNN models can

manage multiple formats of input data, such as 1D for the time-series data and 2D for the imaginary data.

The next section focuses on some NLP tasks and reviews the state-of-the-art text generation models in order to introduce a model capable to auto generate flashcard recommendations for players.

2.4 Transformer-based Models

Several tasks in the field of NLP are rapidly evolving such as text completion, summarization, question/answer generation. Traditionally deep learning has demonstrated its potential over NLP tasks based RNNs [47]. RNNs have positively influenced the NLP fields with excellent results. However, processing long texts caused some limitations [48] due to the sequential processing that slowing down the propagation speed in addition to the need of using parallel computing. This led to the emergence of the Transformer [11] based encoder-decoder architecture as a significant breakthrough in the NLP tasks.

Different Transformer-based models were introduced for text generation such as BERT [13], OpenAI GPT [56] and its successors GPT 2 [12] and GPT 3 [57] for business or the research version GPT-Neo. These models were pre-trained on a large corpus and have the capability of additional fine-tuning to a specific domain. BERT model is constructed using the encoder module, while GPT-2 built by the decoder part of the Transformer. The GPT-2 task for text generation starts with a predefined input text and target text length. Afterwards, the model starts generating one token at a time according to the context of the previous tokens. Once a new token is generated, the model adds it after the previously generated token sequence, and this sequence becomes the new input for the next token and so on until the generated sequence reaches the target length. In contrast, BERT is a

masked language model, and the main task is to predict a masked token in a given text surrounded by other tokens from both sides. Both models have demonstrated remarkable capabilities and results over the state-of-the-art text generation models.

Several studies in the literature [55], [58]–[61] have fine-tuned BERT and GPT2 on customized datasets in different domains to generate text. The authors in [61], investigated the efficacy of dialogue generation for in role-playing games. Despite those models are considered as robust models and can generate human-like text, other authors highlighted on some limitations related coherence and semantic of the generated text.

However, several studies such as in [55], [59]–[64] have transformed the plain text into semi-structured text by injecting the training dataset with metadata, special tags, and tokens to provide extra level of annotation to describe the text structure while training data and to improve coherence and semantic of the generated text.

Also, other Transformer-based models have been introduced such as BART [14] and T5 [15] as encoder-decoder models to include both text understanding and generation to perform a conditional text generation. These models are sequence to sequence (seq2seq) models which take a sentence (sequence of words) as an input in the encoder part and produce another sentence as an output in the decoder part. In practice seq2seq models are capable to outperform the other models in some NLP tasks such as in summarization [65], question generation and translation [66].

Several studies have investigated the question/answer generation task. The authors in [67], [68] have proposed to generate questions from a given text, while the authors in [69], [70] focused on generating answers from given questions. Recent studies in [71], [72] have investigated and proposed generating questions and answers together. In [71] the authors used the ProphetNet [73] model to generate questions from a given text. They filtered questions by computing the cosine similarity to exclude unanswerable and

irrelevant questions, then they generated answers for the non-excluded questions using BERT model. On the other hand, the authors in [72] have fine-tuned the BERT model to extract keywords from a given text, then they used some syntax analysis tools to extract the complete sentences containing the extracted keywords, while in the last phase they feed the complete sentences into the fine-tuned GPT-2 model to generate questions.

However, a very recent review study in [66] tried to provide a comprehensive understanding of question generation tasks in terms of the input context text, the target answer, and the generated question. The study revealed that the majority of the questions generation studies have attempted to fine-tune the models BERT and GPT which are not appropriate for the questions generation task, since they are designed for language understanding instead of language generating. Accordingly, the authors suggest using seq2seq models which are capable of modeling together context texts with different lengths, answers with different granularities, and questions with several types. Moreover, the authors suggest focusing on questions generation for information seeking and recommendation approaches such as in the conversational search and interacting with users for recommendation which is our approach in the flashcard generations.

Also, Kurdi et al. in [54] have conducted a recent comprehensive systematic review of 93 studies addressing the automatic question generation for educational purposes. The systematic review revealed that the majority of the studies in the literature focus on some areas such as (1) generating questions for the purpose of assessment; (2) generating questions for the language domain; (3) template-based approach was used with pre- and post-processing tasks.

In this research, we will introduce a Transformer-based Flashcard generation in the field of learning programming skills based on the findings of the discussed review studies. The

flashcards will be generated as recommendations and information seeking for players in the form of supporting paragraphs and questions/answers aligned with seq2seq models.

CHAPTER 3

Intelligent Serious Games Model

3.1 Model Overview

Reviewing the literature have revealed some important future directions to improve serious games with data analytics research area such as focusing more on players in adaptive gaming experience and applying neural networks for data analytics. Hence, we introduce in this research a novel Intelligent Serious Games (ISG) [23] model (see Figure 1) as adaptive knowledge tracing and recommendation environment. The proposed model combines iGDA conceptual serious games framework with an intelligent model of deep knowledge tracing and Transformer-based recommender. The main goal of the new ISG is to adapt and adjust learning trajectories of players according to their level of proficiency in gameplay. In particular, the model looks one or more steps ahead during the gameplay and predicts the result of the next missions before they are happening in order to provide players with proactive recommendations to complete the missions successfully. Therefore, the ISG continuously keeps tracking and analyzing the current players' knowledge state to predict the future knowledge state of the players and what will happen in the next mission (success/fail). Based on that, and when the predicted knowledge state of the next mission is (non-acquired/fail), the recommender system will be able to make a proactive action by generating guide flashcards relevant to the next mission. These flashcards aim to form a long-term memorization for players using the spaced repetition technique which we will describe in the next sections.

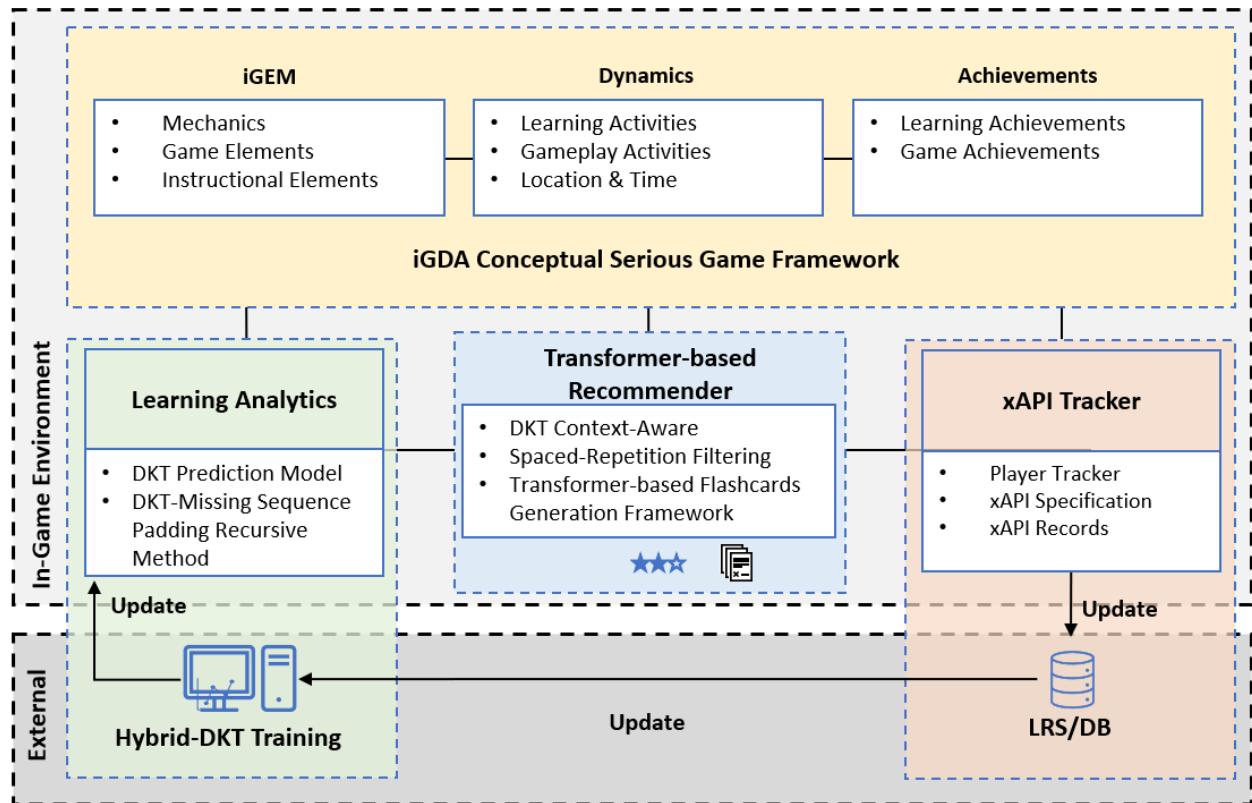


Figure 1. The Intelligent Serious Games (ISG) Components

The ISG model shown in Figure 1 consists of four components as follows:

1. **iGDA Conceptual Serious Game Framework:** a novel re-usable serious game framework, aiming at clearly identifying all game and learning components to be integrated at the early stage of design. In addition, to increase the overlapping between the game elements and mechanics with the instructional content. In fact, the iGDA framework is aligned with the learning-based outcomes approach, and this allows modeling the knowledge state of players with measurable achievements to easy integrate the DKT component and the recommender in real gaming experience.

2. **Learning Analytics:** located in two separated environments: in game environment which is a generic component can hold any analytics approach. In our research it consists of the DKT prediction model with the DKT missing sequence padding method integrated with the serious game and the recommender system; externally, connected on-demand with the Hybrid-DKT training host which is responsible to update the DKT prediction model when needed.
3. **Transformer-based Recommender:** a novel re-usable recommender system acts based on the DKT prediction result, consists of a novel fine-tuned Transformer-based Framework tailored to three different tasks to generate flashcards recommendation, and spaced repetition filtering technique. The recommender system (RS) is Integrated with the serious game and the learning analytics as well. The RS works during the gameplay based on the DKT prediction result to generate personalized guide text for players as flashcard recommendations. The recommender follows the spaced repetition technique to filter and show flashcards to form a long-term memory and reduce the cognitive load.
4. **XAPI Tracker:** this component is based on xAPIs [26] data and communication standard. The main goal of this component is to provide a standard data format with communication protocol for tracking, capturing, storing, and sharing players' interactions to be analyzed in the LA layer (in-game environment) and externally with other databases or learning record stores, to improve data availability and enhance the serious game data analytics research field.

3.2 iGDA A Conceptual Serious Games framework

Serious games refer to games designed for learning with entertainment components and attempt to focus more on the player performance. Therefore, in order to contribute to this

topic, we intend to develop a conceptual framework of serious game that integrates the learning model and the game model all together. The main goals of this framework as follows:

- Define a structure and framework of the interrelationship between learning and gaming models, to allow game designers, instructional designers, and data analytics experts to work jointly in designing a dynamic and integrated environment based on the learning achievements.
- Recognize measurable achievements based on learning outcomes to enable modeling the knowledge state of players and easy integrate the DKT component.
- Identify game dynamics with location and time context aligned with the knowledge state of players where action can take place to integrate the Transformer-based recommender in real gaming experience.

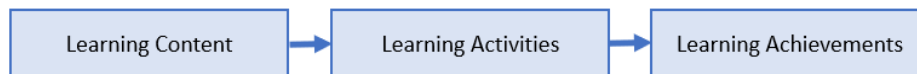


Figure 2. Learning Based-Achievements Model

Learning achievements or achieved learning outcomes are related to the knowledge, skills and attitudes that can be demonstrated by learners after completing a certain course [75], [76]. Also learning achievements can be assessed using qualitative indicators or quantitatively using measurable performance such as in formal exams. From the educational point of view, the instructional material must be designed in a way to achieve the intended learning outcomes as shown in Figure 2. According to Hayes [77], any game designed for educational purposes must be closely related to the educational goals, and he adds that the greater the degree of the overlap between the learning objectives and the game elements, the greater the educational effectiveness of the game.

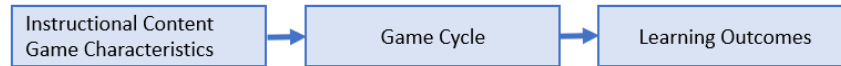


Figure 3. Inputs-Process-Outcomes by Garris

Outcome-based learning is an important approach to improve learning effectiveness and enhance learner achievement. Garris [33] has proposed a model of instructional games consistent with modern educational theories, which is called the Input-Process-Outcome (Figure 3). This perspective presents the learning cycle as “Process” whose results are in the form of learning outcomes as “Outcome” while the “Input” is instructional content with game characteristics. This approach is a learning-oriented model and satisfies the learning needs which we will adopt in our study, but it neglects different gaming aspects such as mechanics, game elements and game achievements.

According to [40], [42], [79] mechanics are related to the game rules, settings, actions, behaviors, boundaries, and control mechanisms that define the player interactions during the gameplay. Although [39], [41] found that mechanics are not everything in the game whereas several aspects should be described such as game content, storytelling, user interface and so on. Regardless of the context of the game Ferro [80] in 2021 sees that there is no specific and consistent definition of the game mechanics, so they divided the game into two parts, game elements and game mechanics (GEM). In GEM, the game mechanics include set of verbs and rules, while the game elements mean the game parts which represents the player, quests, points, timer story and others. Therefore, we concluded that GEM approach is more realistic to include various aspects of the serious games.

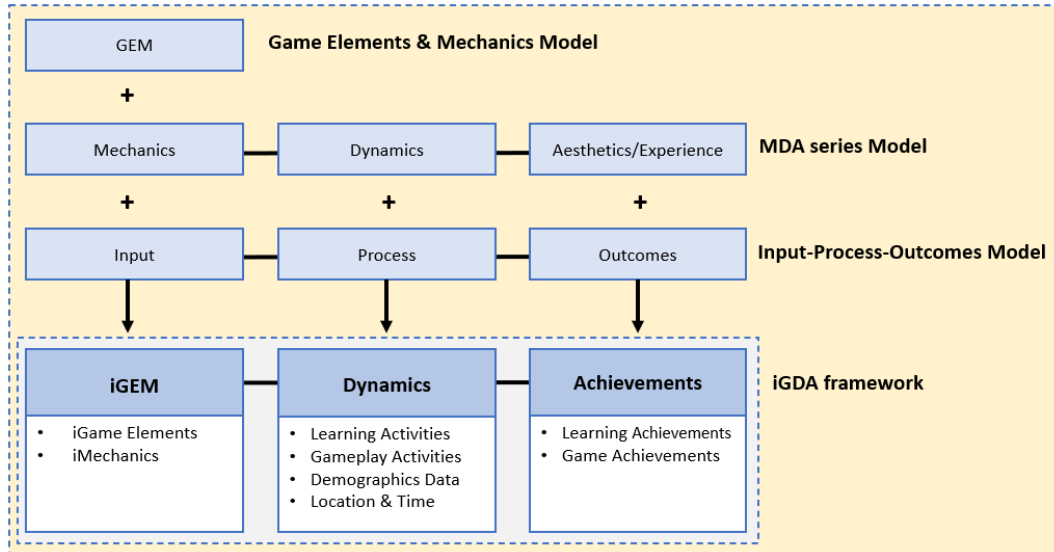


Figure 4. iGDA Framework combines three prior models

At this end we can define Mechanics as: a set of rules and actions that control the interaction between the player and the game elements during the gameplay. Whereas game elements are the characteristic parts of the game including avatars, story, levels, points, quests, timers, and other items in the game environment. While instructional elements are those related to learning objectives, learning materials, and learning activities.

In this research, we combined some components of the state-of-the-art models MDA [40], GEM [80] and INPUTS [78] (see Figure 4) with new specifications to introduce the Instructional Game Elements and Mechanics iGEM framework (see Figure 5).



Figure 5. iGDA Framework

3.2.1 iGEM

iGEM integrates the learning model with the game components by introducing two main components:

- **iGame Elements:** consist of instructional elements (iElements) such as concepts, learning materials and learning activities; game elements ((gElements)) such as timers, badges, help, levels, and places.
- **iGame Mechanics:** consist of instructional mechanics (iMechanics) such as measurable educational verbs-based outcomes such as explore, evaluate, learn and answer; game mechanics (gMechanics) such as game rules and verbs as move, accept, select, and open.

This integration increases the overlap between the learning elements/mechanics and the gameplay elements/mechanics by constructing iGEM mapping matrix. The basic idea of the iGEM mapping matrix is to map the instructional element/mechanic with relevant game element/mechanic where available. For instance, in our C++ Code Challenge serious game (described in the next Chapter) there is a mechanic called “move” which enables the player to move around the board visiting the campus buildings one after another in the form of turns. The integrated instructional mechanic (iMechanics) for move

is “explore” which provides Exploration-based learning as an active learning approach by exploring the instructional content in the game while moving.

3.2.2 Dynamics

Dynamics create achievements and represent the run-time behavior and player’s input of the game. Building on this, all activities and interactions are generated during the gameplay are producing the playing outcome as achievements for players. Dynamics structure includes four main components:

- **Learning Activities:** generated from the player’s interaction with the instructional elements in the iGDA. Indeed, learning activities influence the knowledge state of players and produce the learning achievements.
- **Gameplay Activities:** generated from the player’s interaction with the game elements and produce the Game Achievements.
- **Demographics Data:** which is related to the player gender, age range, country, background, and others. This part of the dynamics is crucial when we intend to integrate filtering technique for items customized with the player preferences.
- **Location and time:** related to the current place of players in a certain time with reference to their achievements. Location and time provide a context for the current knowledge state and the game state of players, where actions can be taken from the game such as providing feedback and recommendation for players.

3.2.3 Achievements

The learner's achievement in any educational experience represents the cornerstone of self-knowledge of the educational needs. It also provides stakeholders with a vision and indicators about the success, effectiveness, and weaknesses of this educational experience. Since serious games are designed for educational purposes rather than for

entertainment, we believe that they are intended to achieve specific goals like any educational experience. Therefore, we expect serious games to make a difference in the level of knowledge, skills, and attitudes of players in the form of achieved learning outcomes. Therefore, in our context it is important to shed more light on the term achievements rather than emotions [42] or aesthetics [40], since the main outputs are learning achievements [76], while lesser outcomes could be related to aesthetics and emotions.

Also, in the literature we found that the term “achievements” does not contradict with the outcomes of the entertainment games. Thus, achievements can be badges, scores, icons, unlocked levels that linked with the fulfillment of specific goals in the game. According to [81] and [82] many gaming platforms have started to adopt the term achievements in their games such as Xbox and PlayStation, and they provide a way to track achievements among the different games that the player has played.

Indeed, achievements in the iGDA framework divided into learning achievements and game achievements. The latter represents for instance the finished turns, earned badges, scores, unlocked places and others related to the game elements. Whereas, learning achievements represent the current knowledge state of players at certain time and place in the gameplay with measurable indicators as “acquired knowledge” or “success” in front of each concept or learning activity.

3.3 Learning Analytics (LA)

3.3.1 LA Player’s Model

Data analytics have been applied in serious games for several purposes. Most of the prior studies in SGs have focused on assessing the players’ performance to assist decision makers in improving design and curriculum. In this research, we focus on the player’s

model of the learning analytics. Zanichelli et al., [83] have proposed a conceptual model of learning analytics for serious games at which we integrated our work with (Figure 6).

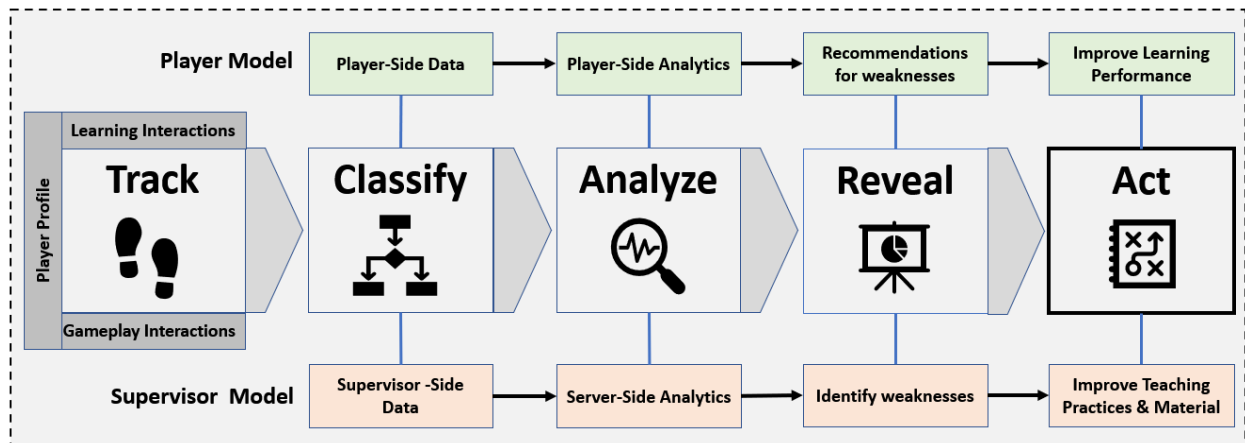


Figure 6. Conceptual Model Process of Learning Analytics

The model consists of five stages aligned with two parallel models, the supervisor model and player model which is our scope. Along the five stages Track, Classify, Analyze, Reveal and Act, the player model gives more specifications on what is happening in the player-side during the gameplay. At this end, the Learning Analytics component we proposed in the ISG is a player-side process starts by capturing players' interactions, performing player-side analytics (predict the next knowledge state), then providing players with suitable recommendations to improve their learning performance.

3.3.2 Learning Analytics Environments

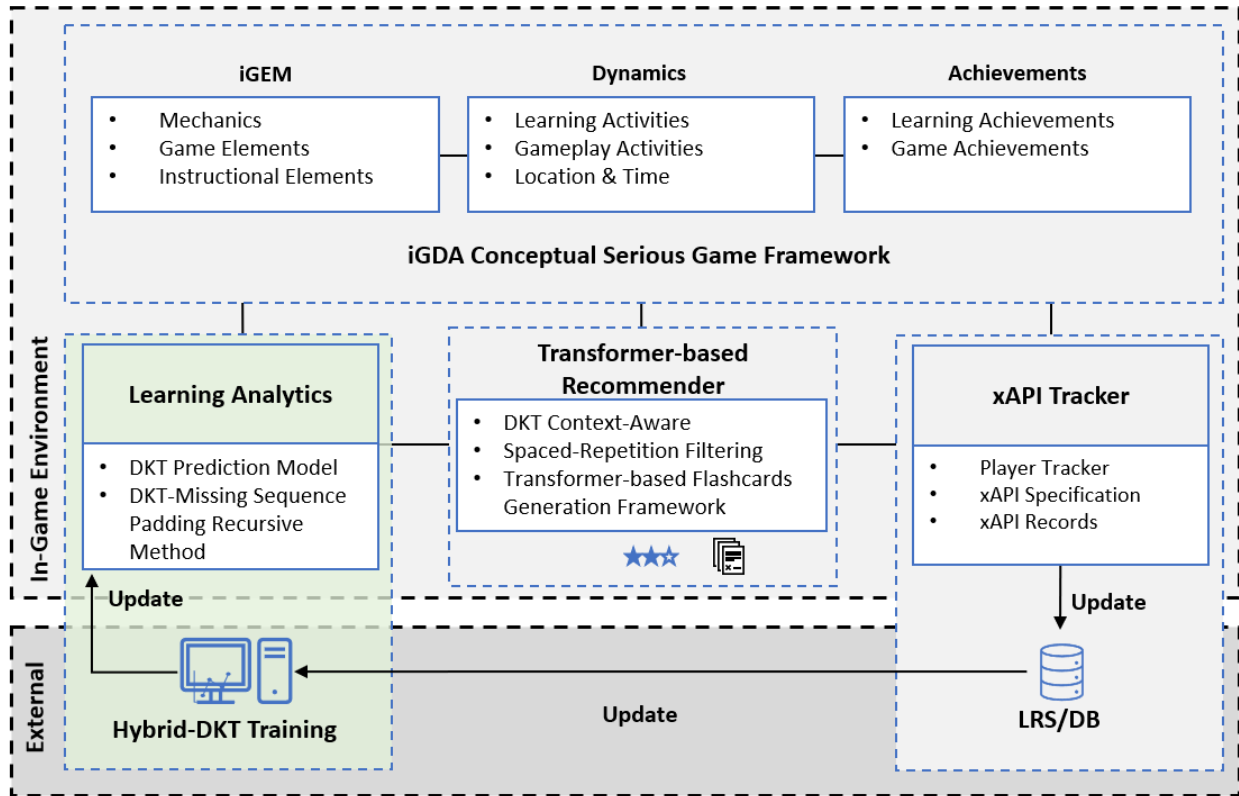


Figure 7. Learning Analytics and SG Environment

The proposed learning analytics component shown in Figure 7 is linked with the xAPI Tracker component to receive the captured interactions in real-time. Also, it is linked with the Transformer-based recommender. Thus, the learning analytics component receives interactions during gameplay, process them using the DKT prediction model and the DKT Missing Sequence Padding algorithm when needed, then it forwards the results to the recommender as a final destination. Indeed, both learning analytics component and the recommender are aligned together and dedicated for players to complete the cycle of “predict then recommend” and so on during the gameplay.

LA external environment: relevant data of players' interactions is forwarded instantaneously during the game to external Learning Records Store (LRS) in the xAPI Tracker component. Consequently, the new interactions forwarded again on-demand (an offline and not automated process) from the LRS to the Hybrid-DKT training host, which is responsible on training/updating the prediction model with the new interactions, then it updates the game engine (in-game environment) with the new updated DKT prediction model.

LA in-game environment: this environment includes the entire serious game mechanics and elements. Players' interactions and achievements occur within the dynamics and achievements components. while the xAPI Tracker component forward the interactions and achievements instantaneously to the DKT prediction model which is also responsible for re-forwarding the results to the recommender system to act accordingly.

3.3.3 DKT-Each Sequence Length Approach

As we mentioned previously in Chapter 2, most of the previous studies in DKT used a non-fixed sequence length of students' submissions with undetermined sequence order. This was as a result of the state-of-the-art DKT task [5] given a series of interactions $x_1, x_2, x_3, \dots, x_t$, predict the next interaction x_{t+1} . Indeed, unordered submissions from basic to advance difficulty could influence the performance of the model as there are explicit and implicit dependencies between each series of concepts, and new knowledge is dependent or partially dependent on previous knowledges. Also, to apply the model in real environment we need to trace each knowledge state of players or in other words each sequence length. Therefore, in our research, we focus on investigating and comparing the performance at each sequence length within a fixed and ordered sequence length. Each sequence length approach in the DKT can be formalized as:

given M ordered game challenges (exercises) $x_1, x_2, x_3, \dots, x_M$, the task is to predict the result of each next game challenge x_{s+1} (sequence) whether it will be completed successfully or not, where S from 1 to $M-1$, and $M > 1$.

- $DS = \{I_1, I_2, I_3, \dots, I_j \dots I_N\}$ a given dataset where I_j is a submission of the j^{th} student.
- $I_j = \{x_{j1}, x_{j2}, x_{j3} \dots x_{js} \dots x_{jM}\}$ a sequence of ordered responses with fixed length M for the j^{th} student.
- $x_{js} = \{Q_{js}, A_{js}\}$ A response entry at sequence S where Q is a question id and A is an answer outcome as wrong/correct where $A \in \{0,1\}$.

N is the number of students, I_j represents a vector of sequence of ordered responses with fixed length $M > 1$, and a response x_{js} at sequence S is performed by the student j . Each response entry x_{js} contains two elements Q_{js} to represent the question id and A_{js} for the answer outcome entry as 0 for wrong answer and 1 for correct answer.

To feed the neural network, each submission $I_j = \{x_{j1}, x_{j2}, x_{j3} \dots x_{js} \dots x_{jM}\}$ should be transformed into vectors for each input and output. $M - 1$ Vectors with length $2M$ to represent the inputs x_s as $\{Q_s, A_s\}^{2M}$, and $M - 1$ vectors with length M to represent the outputs y_{s+1} as $\{A_{s+1}\}^M$ for each entry x_s and y_{s+1} where $s = 1$ to $M - 1$, and $M > 1$.

3.3.4 Hybrid-Deep Knowledge Tracing (RNN-CNN)

RNNs and CNNs are the most popular adopted machine learning models for the time series and image recognition problems. Several prior studies have combined the LSTM and CNN together in other research areas [84]–[87] with excellent performance results. In this study, we propose a hybrid prediction model for the Deep Knowledge Tracing problem (Hybrid-DKT) based on combining the advantages of the RNNs variants and the CNN neural networks. The proposed hybrid network combines two key learning

characteristics together: sequence dependencies learning and feature/pattern extraction. Unlike prior studies in DKT, our work focuses to provide prediction performance on each sequence length within a fixed and ordered series of 22 sequences. Therefore, we seek to improve the performance for each single sequence. We expect from the hybrid combination to improve the performance along all the sequences in qualitative way by level-up the prediction performance ranges as we combined two different learning techniques. This means that, long-term and sequential dependencies will be discovered by the RNNs variants models and passing the results to the CNN model might influence the performance by discovering hidden features and more patterns among the sequences.

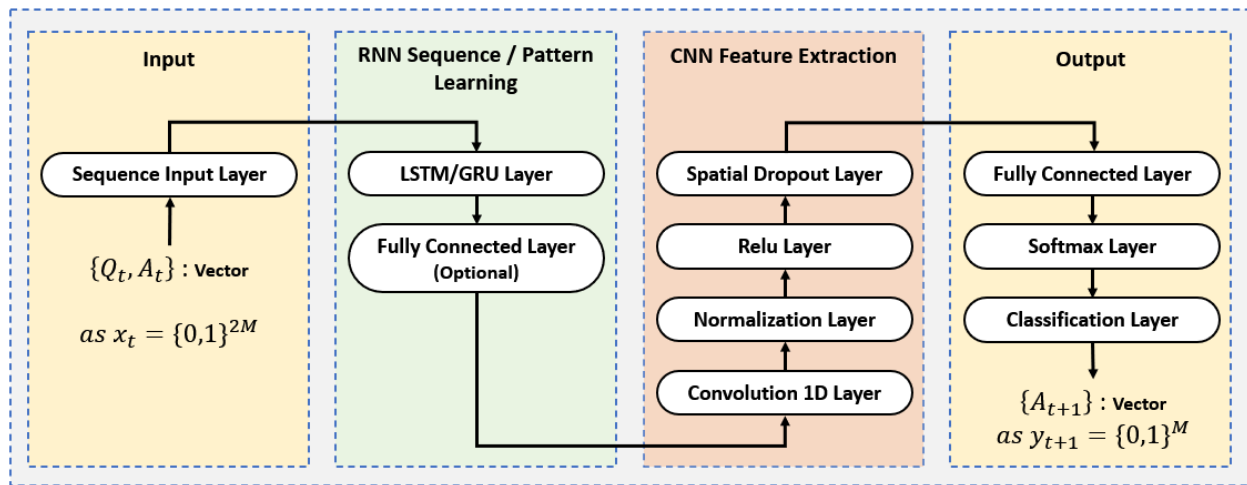


Figure 8. Deep Knowledge Tracing-based Hybrid Model

The Hybrid-DKT [23] model in Figure 8 consists of four layers: input layer, RNN layer followed by CNN layer and output layer. This architecture enables a multi-level learning and feature extraction. Initially the input layer feeds the LSTM/GRU layer to capture the sequence patterns and dependencies from the sequence of input vectors. Accordingly, the LSTM/GRU forward its output to a fully connected layer to adjust weights and bias (as

optional), then it feeds the 1D convolutional layer to learn the low-level features and make more tuning for the output of the LSTM/GRU to produce the feature map. A normalization layer usually is used after learnable layers to normalize a mini batch of data across all channels, to speed up training and reduce the sensitivity to network initialization. The Rectified Linear Unit (ReLU) is an activation function for each input sequence, any value less than zero is set to zero. ReLU usually used to overcome the vanishing gradient problem, allowing models to learn faster and perform better. The dropout layer offers a method to prevent neural networks from overfitting during training. The output layer is the last phase of training by passing the outputs from the CNN layer to a fully connected layer and applying the SoftMax function then the classification layer to infer the number of classes from the output size which 0 or 1 in our case.

3.3.5 DKT-Missing Sequence Padding Method

In flexible learning or playing scenario, there are two different strategies ahead for learners or players. The first one is to move sequentially to learn from the easiest concepts to the difficult ones, while the second is to move randomly or jump-moving to learn the difficult concepts first, and implicitly acquire the simple concepts. In reality, moving randomly raises an issue of missing previous sequences in the DKT prediction model as shown in Figure 9. Usually, we predict the next code challenge x_{s+1} given that all the previous code challenge results. In moving randomly, the player most likely to jump over some unanswered code challenges. For this problem we propose below a solution to overcome the issue of missing values called DKT-Missing Sequence Padding Method [23].

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$P(x_8)$
Ideal DKT Prediction	1	0	1	1	0	0	1	1	?
Missing Sequence Values Problem	1	0	1	1		0			?

Figure 9. Missing Sequence Values Prediction Problem

Generally, the DKT method needs a threshold of at least one value available within a series to predict the second one. In the Code Challenge game (will be discussed in the next chapter), we set this threshold to 4 as players have to answer 4 code challenges to start the game. Therefore, in our case we can predict starting from index 4 as our indexing starts by 0. Table 1 shows the DKT Missing Sequence Padding (MSP) algorithm. For a given code challenge sequence set from 0 to s our goal is to predict the value of the sequence $s + 1$.

Table 1. DKT-Missing Sequence Padding (MSP) Algorithm

Output: DKT_Predict (x_{s+1})	
1	Sequences: [$x_0, x_1, ..x_i, ...x_s$]
2	threshold = 1 // $x_s > A$ threshold should be ≥ 0
3	$first_missing_sequence \leftarrow i$
4	If $first_missing_Sequence > threshold$
5	Recursive_MSP (Sequences, $s + 1$)
6	end
Recursive_MSP (*vector, index)	
7	If $index == first_missing_Sequence$
8	DKT_Predict (vector, index)
9	return
10	end
11	Recursive_MSP (vector, $index - 1$)
12	If $vector[index] == missing_Sequence$
13	DKT_Predict (vector, index)
14	end

1. Initially, at line 2 assign the threshold index ≥ 0 at which the model is able to predict the next value, for instance threshold = 1 which means that the model is capable to predict after the sequence x_1 .
2. Assign the index i at line 3 which is the first missing value in the series and obviously it should be greater than the threshold.
3. The MSP algorithm initiates at line 5.
4. At line 11 the MSP algorithm starts predicting x_{s+1} recursively by decrementing the index and looking back to temporarily predict and fill the missing values only at lines 12 and 13.
5. When the decremented index reaches the first missing value (line 7), predict the missing value at line 8 and terminate the recursive calling at line 9 and exit the program.

3.4 Transformer-based Recommender

In the previous section, we discussed how to estimate the next knowledge state of players, i.e., to know whether they already acquired a specific concept or not. In case it is not, we apply a proactive recommendation to help players to acquire this concept before proceeding to the next mission.

In this section, we propose a novel recommender architecture based on knowledge tracing, spaced repetition filtering and three NLP tasks to auto generate flashcards (Figure 10). Flashcards will be provided for players in the form of recommendations and information seeking. Thus, each flashcard consists of a question to be presented for the player first, an answer to be shown after certain of time, and a hidden supporting paragraph can be viewed on demand.

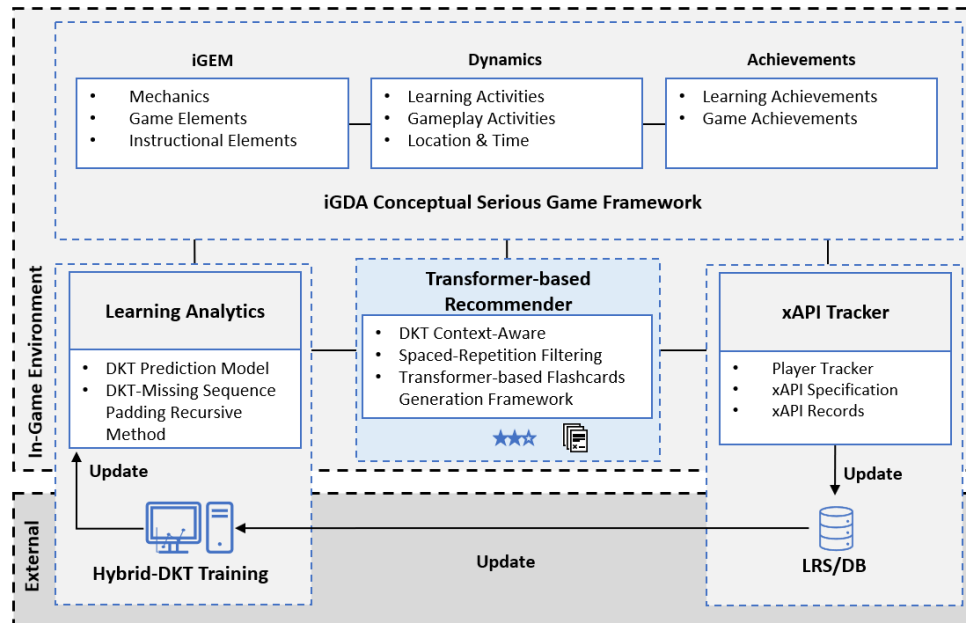


Figure 10. Transformer-based Recommender

3.4.1 Spaced Repetition and Flashcards Approach

Spaced repetition involves one of the important learning and training techniques to form a long-term memory. According to [16], [88], spaced training is an evidence-based learning technique aims at reviewing the learning material at systematic intervals. In addition to that, spaced training is more effective for facts, concepts, skills, and others. However, flashcard method is one of the widely spread application on the spaced repetition technique. The basic idea of using the flashcards is to filter those with well-known concepts with less frequent review, while difficult or forgotten concepts are filtered for more frequent review to be shown frequently in spaced intervals.

In the context of this study, the Transformer-based recommender generates flashcard recommendations and filters them for players according to the spaced repetition technique to show difficult concepts frequently.

3.4.2 Recommendation and Filtering Context

Generally, a recommender system is a personalized technique that can recommend items based on the user's needs [89], [90]. According to Ricci et al., [33] there are three actors in any recommender system namely environments (items to be recommended), users and interactions that were captured between users and items. Traditionally, there are several filtering and recommendation techniques such as: content-based to recommend similar items by filtering those the user liked; collaborative-based which recommend items that are liked by similar users; hybrid-based which combine multiple filtering techniques; and the most recent technique for recommendation is the context-aware [90], [91], which tracks the user's state and consider recommendations at specific situation linked with time, place, and context.

The proposed Transformer-based recommender is context-aware with the DKT prediction result as the reaction time and with the player's attempt to move to a new code challenge as the location. The main task of the recommender is to generate recommendations for a player in the form of flashcards as soon as the DKT prediction result is most probably "Fail" in the next mission. In addition to that, the recommender performs on each reaction a spaced repetition filtering to present all previous flashcards labeled with knowledge state "non-acquired" for a player. Indeed, this recommender reaction and filtering technique enable an evidence-based learning approach for players to form a long-term memory through spaced repetition practice and feedback-driven metacognition [17], [92]. In particular, each flashcard consists of a question and an answer to improve information retrieval of players followed by a supporting paragraph to be provided as feedback-driven for players *"to know what they know and know what they don't know"* [92].

3.4.3 Transformer-based Recommender Architecture

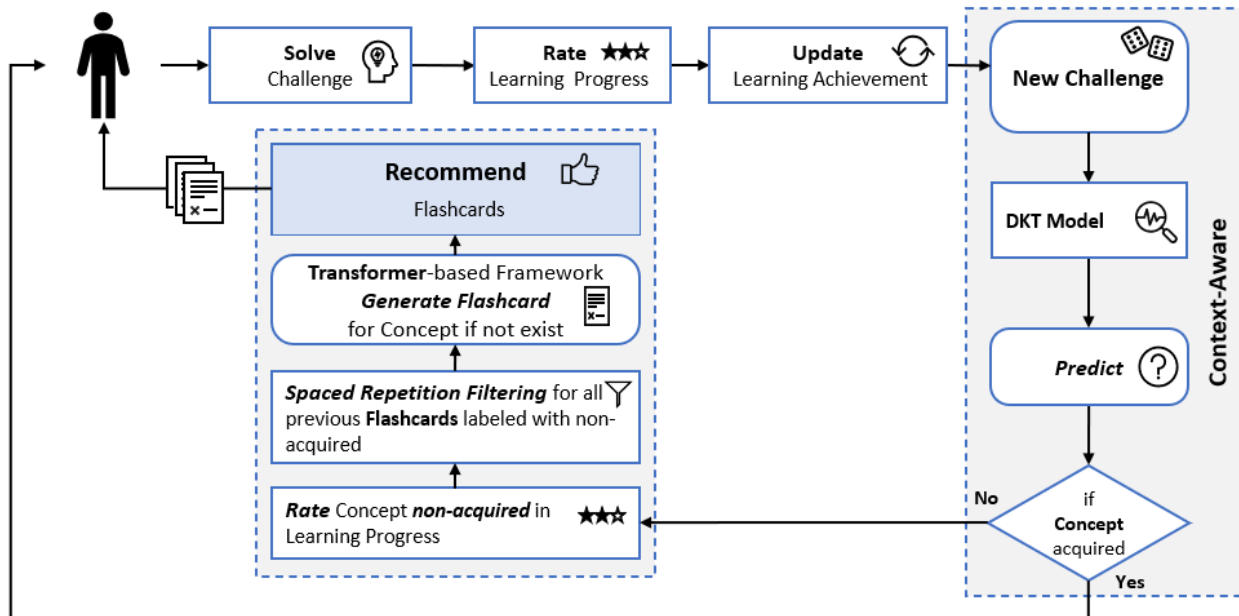


Figure 11. Transformer-based Recommender System Architecture

The architecture in Figure 11 shows the Transformer-based Recommender with the context-aware component.

- **Context-Aware component:** the overall process initiates when the player intends to move to the next new challenge, the DKT predicts the result of the next new challenge, and if the predicted result state is “acquired” the context-aware component allows the player to move and solve the challenge without recommendations. Otherwise, the context-aware component proceeds directly to the recommendation process before letting the player solve the challenge.
- **Transformer-based Recommender:** initiates by rating the predicted concept as non-acquired, applies spaced repetition filtering to retrieve all previous flashcards labeled with non-acquired, generates a flashcard for the predicted concept if it is

not already generated from previous turns, and finally recommends and shows the new flashcard with all filtered ones for a player

3.4.4 Transformer-based Recommendation Algorithm

Output: Recommend_For_Player (Transformer_Flashcard_Generator (c_k))

Context_Aware_Code()

```

1 | Concepts: [  $c_1, c_2, \dots, c_i, \dots, c_t \dots$  ]
2 | Next Challenge  $\leftarrow c_{i+1}$ 
3 | Result  $\leftarrow$  DKT_Predict ( vector,  $c_{i+1}$  )
4 | If Result is non-acquired, then
5 |   | Transformer_Recommender (  $c_{i+1}$  )
6 |   end

```

Transformer_Recommender (c_{i+1})

```

7 | Rate  $c_{i+1} \leftarrow$  non-acquired
8 | foreach  $c_k$  in Concepts where  $k \leq i + 1$  // spaced repetition
9 |   | If  $c_k$  is non-acquired then
10 |    | Recommend_For_Player (Transformer_Flashcard_Generator ( $c_k$ ))
11 |    end
12 |   end

```

Transformer_Flashcard_Generator (C_k)

```

13 | FlashCards_Pool: [  $F_1, F_2, \dots, F_{k-1}, \dots, F_t \dots$  ]
14 | If  $F_k$  is not in FlashCards_Pool
15 |   |  $F_k \leftarrow$  Generate_Flashcard (  $C_k$  )
16 |   end
17 | Return  $F_k$ 

```

Figure 12. Transformer-Based Recommendation Algorithm

Figure 12 demonstrates the recommendation algorithm in three main methods as follows:

1. Context_Aware_Code() method get invoked every time a player intends to move to a new code challenge.

2. At line 3 the DKT model predicts the next challenge c_{i+1} and if the result is non-acquired at line 4, invoke the method `Transformer_Recommender(c_{i+1})` at line 5.
3. Rate the predicted concept as non-acquired in the learning progress at line 7.
4. At lines 8 and 9 perform spaced-repetition filtering by retrieving each previous concept c_k labeled with non-acquired.
5. At line 10 recommend for players a relevant flashcard for the newly predicted concept and each concept c_k labeled with non-acquired by invoking the method `Transformer_Flashcard_Generator(c_k)`.
6. The method `Transformer_Flashcard_Generator(c_k)` at line 14 performs a check if a relevant flashcard F_k is not already generated, then the method generates F_k flashcard at line 15 using the Transformer-based Flashcards Framework that we will introduce in the next section.
7. At line 17 return F_k flashcard for the recommendation method at line 10.
8. Repeat line 8 (point number 4 above) for each concept c_k labeled with non-acquired.

3.4.5 Transformer-based Flashcards Generation Framework

With reference to the recent findings and future directions that we discussed in section 2.5; we introduce in this section a novel Transformer-based framework to generate flashcards in a fully automated process. The new framework combines at least three transformer-based models which are fine-tuned and tailored to three different NLP tasks of flashcards generation process as shown in Figure 13.

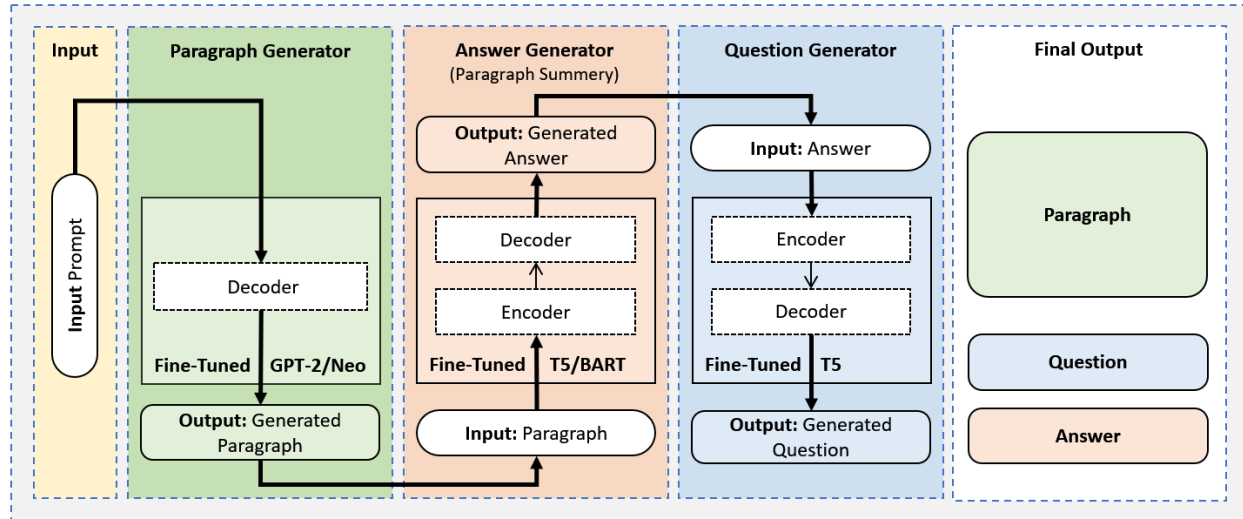


Figure 13. Fully Automated Flashcards Generation Framework

The Flashcards Generation Framework consists of three task generators as follows:

- The Paragraph Generator task receives an input string to generate a relevant paragraph using a fine-tuned GPT-2/Neo decoder, the output of this task is used as an input for the next generator task.
- The Answer Generator task receives a paragraph as an input to generate a relevant answer using the BART/T5 seq2seq models that fine-tuned on paragraph summarization task.
- The Question Generator task receives an answer as an input to generate a relevant question using the T5 seq2seq model fine-tuned on the task of questions generation.

This generic framework can be used in any context for generating paragraphs, answers, and questions. Moreover, the framework needs three different datasets relevant to the discipline to be fine-tuned on the generation tasks. In our research we introduced three C++ programming skills datasets to fine-tune the models on the context of C++ programming skills. Also, for implementing this framework, we recommend evaluating it

using quantitative metrics to measure the N-gram overlaps between the generated and the reference text, and to measure the cosine semantic similarity. We believe this is crucial to maintain coherence text and correct semantically as we will discuss this in chapter 4 in more details.

3.4.5.1 Input Strings for Flashcards Generation

To generate a flashcard for a particular concept the Flashcards Generation Framework needs an input string such as the title of the target concept in order to initiate task-1 for paragraph generation. For this purpose, we can use the learning outcomes matrix which we developed as part of this research, and it will be discussed in the next chapter. This kind of matrix is identifying the code challenges (exercises) as rows mapped to the concepts as columns. Therefore, each code challenge is mapped to a main concept and to one or more secondary concepts when available. Accordingly, when predicting any next code challenge, we can identify all main and secondary concepts that belong to that code challenge. In this case we retrieve the concept title/titles to be used as inputs to generate one or more relevant flashcards.

3.5 xAPI Tracker

Generally, the user profile is a collection of settings and information associated with the user. The Player Tracker (PT) in ISG is responsible for tracking, storing, and sharing the player interactions and achievements during/after the gameplay. In reality, PT provides a unique identity for the player among the other players related to demographic data, performed activities, location, and time.

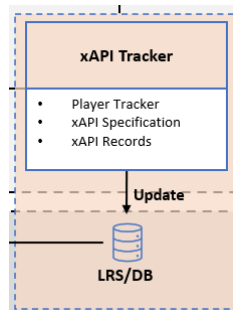


Figure 14. xAPI Tracker

In order to track, store and share the player interactions and achievements, it is crucial to select a standard maintains a data model with a communication model. This standard enables interoperability level with external databases, learning record stores (LRS), learning management systems, and with any recommendation system that can be integrated. In addition, the information should be shared with other researchers for further work in serious games.

3.5.1 Experience APIs

xAPI [26] is a community-driven specifications for learning technology to allow collecting data about a user experience online and offline. xAPI was known before as Tin Can API and have introduced by the Advanced Distributed Learning (ADL), who also introduced the SCORM standard. In xAPI there are two key elements to define data and communication models, for data the xAPI captures the user experience in the form of statements contains actor, verb, and object to be stored and shared to any LRS according to a unique specification that defines the communication method. The main advantages of using the xAPI are the wide adoption and the flexibility of using a wide range of vocabulary and extensions to be adaptable with any environment. In our work, we integrated the xAPI as the main data format and communication component to provide abstraction and wide accessibility for the captured gameplay interactions.

CHAPTER 4

Experiments and Results

In the previous Chapter, we introduced and described the intelligent Serious Games (ISG) model. As described before, the ISG consists of 4 components, the iGDA conceptual framework, Learning Analytics with DKT, the Transformer-based Recommender and the xAPI Tracker. Our approach in this research is to assess and evaluate the efficacy and the feasibility of the ISG components. However, a complete implementation to address the effectiveness of the ISG on the players' attainments will be conducted in future work.

In this work we assessed and evaluated the ISG components through the following:

- we developed the C++ Code Challenge game according to the iGDA serious games conceptual framework, and accordingly we provided the iGEM mapping to identify and describe the overlap between the gaming components and the instructional components. We also developed the C++ Code Challenge learning outcomes matrix to cover 26 code challenge mapped to 17 concepts. Afterwards we transformed the code challenges into online quiz to validate them on real students and to create a training dataset for the DKT prediction model. Finally, we integrated and evaluated the xAPI Tracker component on real students and the ability of sharing the gameplay interactions into two different external data sources.
- we evaluated the DKT method using three novel hybrid models, three state-of-the-art single RNNs variants, and CNN single model on two datasets. The Hybrid-DKT models obtained better prediction performance over the state-of-the-art DKT method with RNNs single variants LSTM, biLSTM and GRU on both datasets.

Also, we evaluated the effectiveness of the DKT Missing Sequence Padding proposed method, the results revealed the ability to recursively predict more than one step ahead with filling the missing values. The metrics that used in the DKT evaluation are the area under the curve AUC and the prediction accuracy ACC.

- we evaluated a novel Transformer-based framework to generate flashcards in a fully automated process through combining three NLP generation tasks: Task1 to generate C++ paragraphs; Task2 to extract and generate answers from the generated paragraphs in Task1; Task3 to generate questions for the generated answers in Task2. For evaluation we considered three common metrics in the text generation problems: SacreBLEU [95] and ROUGE [97] to measure the surface similarity on the words level and SBERT Sentence Transformers [79] to measure the cosine semantic similarity between the reference and generated sentences. The results revealed that the proposed framework is capable of generating coherent and semantically correct flashcards through three experiment sets as follows:
 1. we fine-tuned and evaluated the state-of-the-art GPT-2 and GPT-Neo models on a new C++ textual dataset for generating C++ guide paragraphs.
 2. we fine-tuned and evaluated the state-of-the-art BART and T5 models on a new C++ summarization dataset for generating C++ answers from given paragraphs.
 3. we fine-tuned and evaluated the state-of-the-art T5 model on a new C++ question/answer dataset for generating C++ questions from given answers.

4.1 iGDA Serious Games Framework

The Code Challenge serious game, (Figure 15) developed based on the iGDA model as a board-adventure game using UNITY 2D and simulates the campus environment to

teach the basic skills of computer programming in C++, one of the most important and used programming languages¹. The game consists of 26 challenges, places where players can acquire quick timed-skills, library, badges, concept flashcards, and achievement progress bar which is an evidence-based to track players' success in the game challenges.



Figure 15. C++ Code Challenge Serious Game

Moving on to the player role in the game, the overall goal for the player is to walk through 26 code challenges (see Figure 16) to acquire the programming skills in the form of learning achievements. Initially, players have to answer four code challenges before they can enter the board. Then, the player role is to roll the dice and choose one of the next possible places in the board with some criteria to move and try to overcome the rest of 22 code challenges. Indeed, there are two different strategies a head for players, the first one is to move sequentially to learn from the easiest concepts to the difficult ones, while the second one is to move randomly or jump-moving to learn the difficult concepts first, and implicitly acquire the simple concepts.

¹ <https://spectrum.ieee.org/top-programming-languages/>

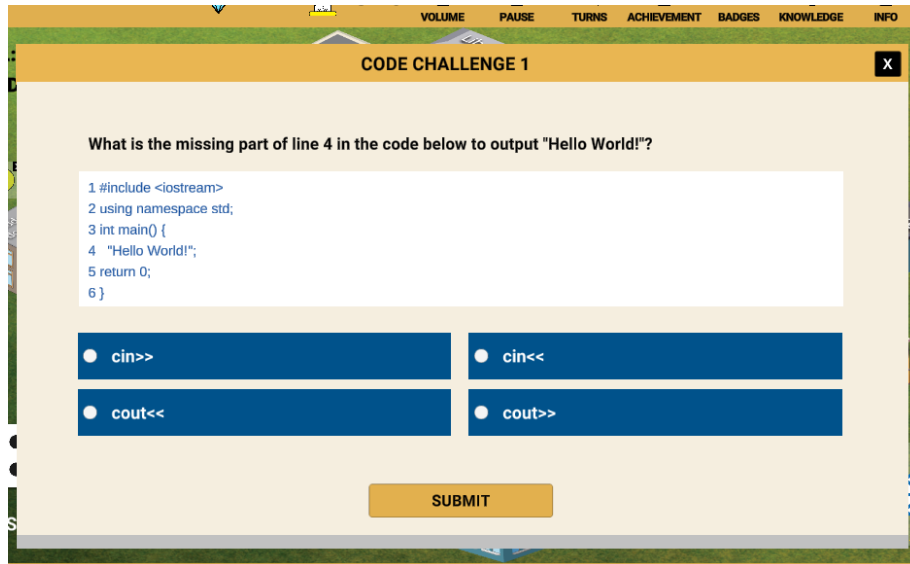


Figure 16. Code Challenge

4.1.1 C++ Code Challenge Learning Outcomes Matrix

We identified 17 concepts of C++ programming using C++ Primer Plus book [93] as a reference, selecting the common basic skills and excluding advanced skills such as those related to objects, classes, and object-oriented programming. Consequently, we developed the learning outcomes matrix by setting-up the 17 identified concepts as columns, followed by creating 26 code challenges as rows mapped to the concepts. The result is each code challenge covers one main concept with one or more secondary concepts (denoted by *M* for main concepts and *S* for secondary concepts) as shown in Table 2. We considered this type of mapping to make sure we aligned each code challenge with at least one concept and to visualize the possible dependencies between them. Also, identifying main and secondary concepts for each code challenge enable the flashcards generation using one or more concept titles as input strings.

Table 2. C++ Code Challenge Learning Outcomes Matrix

#	Code Challenge Type	Syntax	Variables	IO	Data Types	Strings	Arrays	Arithmetic	Math	Logic	Conditions	Loops	Ref/Ptr	Scope	Functions	Recursion	Overload	Struct
1	Syntax1	M		S														
2	Syntax2	M	S		S													
3	Syntax3	M																
4	Variables1	S	M		S													
5	Variables2	S	M		S													
6	Variables3	S	M		S													
7	Input/Output 1	S	S	M	S													
8	Input/Output 2	S	S	M	S			S										
9	Input/Output 3	S	S	M	S			S										
10	Data Type	S	S		M													
11	String1	S	S	S	S	M												
12	String2	S	S	S	S	M												
13	Arrays	S	S	S	S		M											
14	Arithmetic Operators1			S				M										
15	Arithmetic Operators2			S				M										
16	Math	S	S	S	S				M									
17	Logical Operators 1			S						M								
18	Logical Operators 2	S	S	S	S					M								
19	conditional statements 1	S	S	S	S					S	M			S				
20	conditional statements 2	S	S	S	S						M							
21	Loop statements 1	S	S	S	S			S			S	M						
22	Loop statements 2	S	S	S	S			S			S	M						
23	Loop statements 3	S	S	S	S			S			S	M						
24	References / Pointers 1	S	S	S	S	S							M					
25	References / Pointers 2	S	S	S	S	S							M					
26	Functions 1	S	S	S		S								S	M			
27	Functions & Scope	S	S	S	S			S						M	M			
28	Functions 2	S	S	S	S		S	S		S	S	S			M			
29	Functions & Recursion	S	S	S	S					S	S				M	M		
30	Functions & Overload	S		S										S	M		M	
31	Structure	S	S	S	S		S	S		S		S						M

4.1.2 C++ Code Challenges Online Quiz

To validate the code challenges and create dataset1 (C++ Code Challenge Dataset) to train the DKT prediction model, we transformed the 26 code challenges into online quiz as multiple choices and sort statements questions using Microsoft Forms. For most of the questions, the associated source code produces two outputs, thus there are 2^2 possible answers (Figure 17). After that, we published the quiz with total of 36 questions, 26 of them from our game and 10 extra questions with some demographic data, insights, and opinions without names or contacts. We targeted computer engineering, computer science and information technology students from UNIPR. We also shared the quiz with other colleague professors in Italy and abroad to share it with their students. Our goal was to collect responses from trusted population to have high quality series of submissions.

16. What are the outputs of the following program?

```
1. int main() {  
2. string facilityName= "Hotel";  
3. facilityName[0] = 'M';  
4. cout << facilityName + " " + facilityName[0];  
5. return 0;  
6. }
```

(1 point)

64% of respondents (90 of 140) answered this question correctly.

[More Details](#)

 Insights





 Hotel M	36
 Motel M	90 ✓
 Hotel H	8
 Motel H	6



Figure 17. C++ Online Quiz for Dataset

4.1.3 xAPI Integration and Learning Records Store

The Experience API (xAPI) captures the interactions in the form of statements, each statement contains actor, verb, and objects in JSON format. Accordingly, we identified these specifications within the game environment. As shown Table 3, the actor is the player, the verb represents the type of the action such as attempted new turn, with one or more attached objects. We identified several objects for each statement, one to indicate the turn number and activity type. Other objects to describe the result of that activity such as success or fail, response type, duration for this activity and scores. Also, each statement contains a signature with three fields: context, timestamp, and version. Finally, we integrated the xAPI protocol to forward the interactions into two different third-party sources which are the SCORM cloud and google spreadsheets.

Table 3. xAPI Statement

xAPI Attributes	Vocabularies
Actor	Name: name of the player
Verb	Attempted: to indicate a new turn
Object	Name: to indicate number of the turn
	Description: to describe the position/destination of the player
Result	Object type: Activity
	Success: true/false
	Completion: true/false
	Response: to describe the action result of this activity
	Duration: the duration of doing this activity
Context	Score: the earned score for doing this activity
	Supervised game or not by instructor
Timestamp	Time and date signature
Version	The version of the xAPI

4.1.4 Results

4.1.4.1 iGEM Mapping

The proposed iGDA conceptual framework enabled us to identify the serious game components in the design phase through two levels.

In the first level we described and categorized the components into instructional elements, game elements, instructional mechanics, and game mechanics. In this level, the learning model transformed into instructional elements to include concepts, learning materials, and learning activities, and instructional mechanics to include the measurable educational verbs-based outcomes such as explore, evaluate, learn, answer and so on. Moreover, the game objects such as timers, badges, help, levels, and places are categorized as game elements. Whereas the game mechanics identified as rules and verbs such as roll, move, accept, select, open and so on.

In the second level we mapped the iElements with the gElements (see Table 4) and the iMechanics with the gMechanics (see

Table 5), followed by an integration process when available to increase the overlapping between them.

Table 4. Example of Instructional Element and Game Elements Mapping

gElements	Description	Mapped iElements
Flashcards	Used for recommendations	Brief C++ concepts
Diamonds	Represents the scores	Learning achievement indicator
Feedback	information and reaction for players after finishing a task or a challenge	Qualitative and quantitative feedback about the achievement
Progress bars	To visualize the player progress	Learning Achievements

Campus	There are several buildings that represent possible locations on the board, each represent a unique concept and contains code challenges	Concepts and learning activities
Dormitory	A building on the board that represents the starting point where the main mission and objectives are presented as a dialogue scene, some recommendations for players also provided at this place during the turns	Learning objectives and learning recommendations
Library	A building on the board that represents the studying area in the game, and where players can view categorized learning material	Learning material
Round	When the player reaches the end of the board and start over from the starting point	Learning achievement that conducted per a round
Tip card	Lucky cards that can be viewed to give hints related to learning activities	A hint regarding a concept
Player	An avatar representation for the player, it could be male or female	N/A
Board	An area is divided into locations where the player will move through	N/A
Dices	A way to identify a possible movement range by rolling the dices	N/A
Timer	The countdown timer in the game	Learning achievement that conducted in certain time
Volume	Mute/unmute the sounds	N/A
Turn	Rolling the dices, moving, and facing a challenge	Where a learning strategy can be chosen as step by step or jump learning

Table 5. Example of Instructional Mechanics and Game Mechanics Mapping

gMechanics	Description	iMechanics	Description
Choose	Allow the player to choose a location among the movement range	Create, plan, develop	create a playing strategy. i.e., from easy to difficult (visit the nearest place)
Move	Allow the player to walk around the board	Explore, identify	Explore the instructional goals and content
Select	Allow a player to select an option among many options	Identify, solve, recall	Allow the player to identify the correct answer for a specific question
Generate	Allowing the player to generate a new card using the TIP Generator	Evaluate, rate, classify	Allowing the player to evaluate the generated information

Submit	Allowing the player to submit the answer	Review, compare	Obtaining the answer result whether it is correct or incorrect
Use	Allowing the player to use a tip card in answering the question	N/A	N/A
Roll	Turning over the dices to have a movement range	N/A	N/A
Correct	It is the correct result of answering a code challenge	N/A	N/A
Wrong	It is the incorrect result of answering a code challenge	N/A	N/A

4.1.4.2 Dataset 1: C++ Code Challenge Results

A total of 5394 exercise responses were collected from 174 students, 57% males and 43% females, 60% of them from Palestinian universities, 31% from Italy and 9% of them from other countries. The age of 61% of the responders was less than 23 years. Other results revealed that 43% of responders self-evaluated their proficiency level as basic, 42% as medium and 15% as advanced. The majority of them rated the questions' difficulty as average, around 3 out of 5 (5 meaning most difficult). In general, responders were quite satisfied (3.79 out of 5) with the questionnaire.

Table 6. Questions' difficulty and satisfaction

Question	1	2	3	4	5
Questions' difficulty (5 means very difficult)	8%	22%	38%	23%	9%
Responders' satisfaction (5 means very satisfied)	4%	2%	30%	39%	25%

Table 7. Responders' Self-evaluated C++ Proficiency Level

C++ Proficiency Level	Basic	Medium	Advanced
	43%	42%	15%

4.1.4.3 C++ Proficiency Level Distribution

We obtained the proficiency level by computing the average of all responders’ scores for each code challenge. Figure 18 shows the average proficiency level for all students. The results demonstrated that the students obtained an average score of 59% when answering all the code challenges. These results are quite aligned to the self-evaluated proficiency levels as the majority selected “basic” and “medium,” and also aligned with the questions’ difficulty as the majority of responders rated the questions’ difficulty as 3 out of 5.

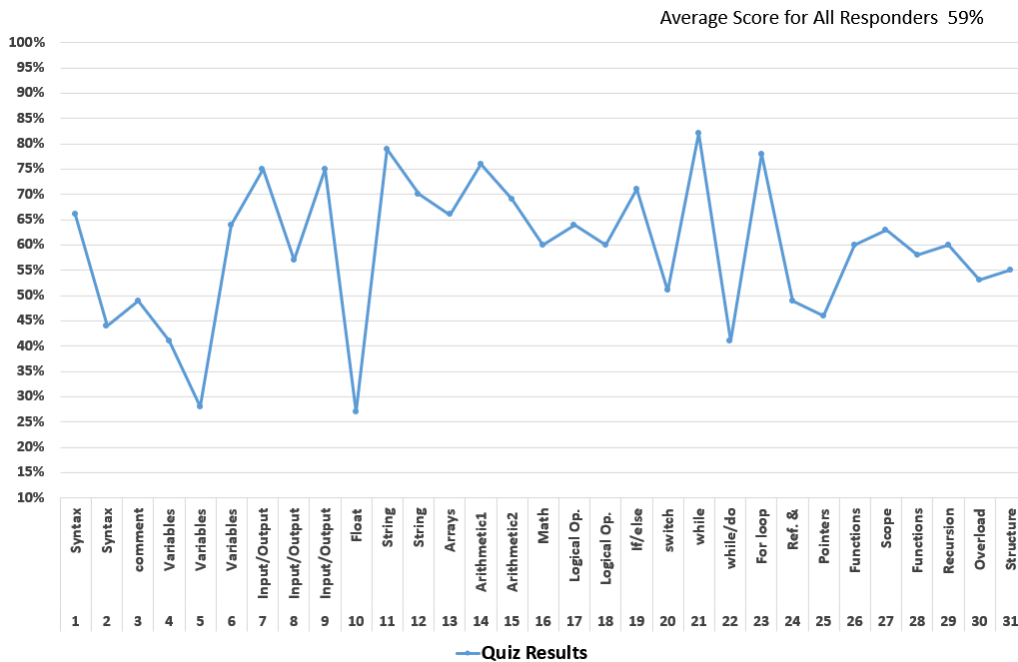


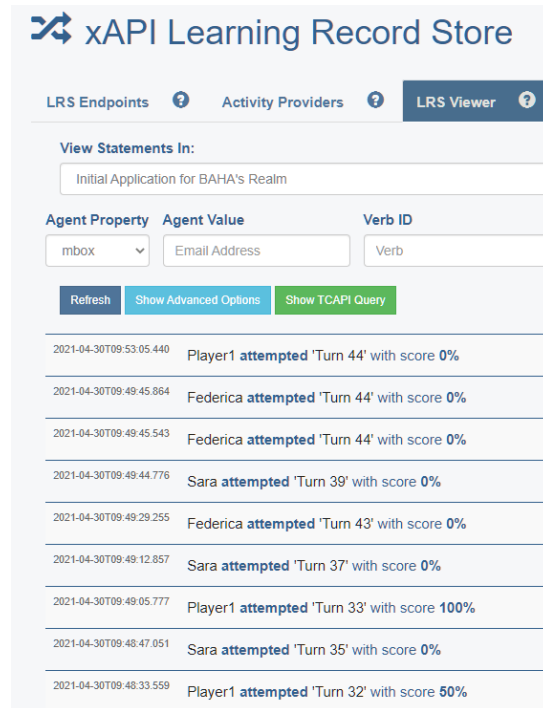
Figure 18. C++ Code Challenge Results (Skills Proficiency Distribution)

The results or the proficiency distribution can be classified into three categories according to the average scores among the 31 sequence answers. First category, most students possess important skills in programming such as dealing with strings, arrays, arithmetic, logic, conditions, and loops with average results between (60% - 80%). However, they

have performed significantly less in floats, switch statements, while/do, references and pointers (20% - 50%). Second category, concepts such as syntax, variables, and input/outputs results shown an oscillating curve from 23% to 73%, and this is due to the majority of these concepts being probably memory dependent and it is easy for the students to memorize them when needed or to correct the error while practicing. Third category, the results revealed that the students showed semi-constant results in dealing with functions, scope, overload, recursion, and structure with results between 50% and 60%, and this may be most probably related on the above results as dealing with functions in programming is a mixed experience of all previous skills and concepts.

4.1.4.4 xAPI Hands-on Workshop

The main purpose of the workshop was to assess and evaluate the practical use of this framework with the xAPI Tracker. Due to some limitations regarding the COVID-19 lockdown, we were unable to distribute and evaluate the game on a large scale. Instead, we have evaluated the game on twenty students. The participants attended the online workshop which have started by introducing the game and the purpose of this workshop. The demographic questionnaire was embedded in the stating part of the game, and it was optional. Then the participants have given 30 minutes to play the game, and the xAPI Tracker was responsible to capture and share all the interactions instantly into two different external data sources the first one was the xAPI Learning Record Store in the SCORM Cloud service and the second was google spreadsheets. The age range was 19 – 29, gender 90% female and 10% male while the country was Italy. Total of 485 turns and interactions are captured in the form of xAPI statements as shown in Figure 19, and each statement contains more specifications regarding the actor, results, verb, and other objects.



The screenshot shows the 'xAPI Learning Record Store' interface. At the top, there are navigation links for 'LRS Endpoints', 'Activity Providers', and 'LRS Viewer'. Below this, a search bar is labeled 'View Statements In:' with the text 'Initial Application for BAHAs Realm'. There are three input fields: 'Agent Property' (set to 'mbox'), 'Agent Value' (set to 'Email Address'), and 'Verb ID' (set to 'Verb'). Below these fields are three buttons: 'Refresh', 'Show Advanced Options', and 'Show TCAPI Query'. The main content area displays a list of statements with columns for timestamp, agent, and verb details.

Timestamp	Statement
2021-04-30T09:53:05.440	Player1 attempted 'Turn 44' with score 0%
2021-04-30T09:49:45.864	Federica attempted 'Turn 44' with score 0%
2021-04-30T09:49:45.543	Federica attempted 'Turn 44' with score 0%
2021-04-30T09:49:44.776	Sara attempted 'Turn 39' with score 0%
2021-04-30T09:49:29.255	Federica attempted 'Turn 43' with score 0%
2021-04-30T09:49:12.857	Sara attempted 'Turn 37' with score 0%
2021-04-30T09:49:05.777	Player1 attempted 'Turn 33' with score 100%
2021-04-30T09:48:47.051	Sara attempted 'Turn 35' with score 0%
2021-04-30T09:48:33.559	Player1 attempted 'Turn 32' with score 50%

Figure 19. xAPI Captured Statements

4.1.5 Summary

In summary, we developed the C++ Code Challenge game according to the iGDA serious games conceptual framework and we introduced the iGEM mapping. The iGEM mapping enabled us to identify iMechanics, gMechanics, iElements and gElements, in addition to increase the overlap between them. We also developed the C++ Code Challenge learning outcomes matrix covering 26 code challenge and mapped to 17 concepts. We also validated the code challenges and created a training dataset for the DKT prediction model. The validation results of the code challenges revealed that the majority of students possess important skills in programming such as dealing with strings, arrays, arithmetic, logic, conditions, and loops. Whereas the students' performance significantly decreased in floats, switch statements, while/do, references and pointers. Also, we integrated and

evaluated the xAPI Tracker component on real students and results revealed the ability of using this integration to share the gameplay interactions into two distinct locations which are SCORM learning record store and google spreadsheet.

4.2 Hybrid-Deep Knowledge Tracing

To assess and compare the Hybrid-DKT models on dataset1 that we introduced in section 4.1 with other public datasets, we found most of the previous studies have evaluated only the overall performance of their DKT models, unlike our own work to evaluate each knowledge state as described in section 3.3.3. They mostly used some public datasets such as ASSISTments' skill builder [94], Algebra 0506 and Statics2011. Those datasets have some limitations such as: each student has a different length of responses; some questions have multiple associated scaffolding questions, therefore there are multiple sequence of responses with sub-sequences; undetermined submission order and repeated responses for each student where a student can keep trying until a proficiency threshold is reached. Differently, our approach (see section 3.3.3) is to evaluate each knowledge state by assessing each response sequence in a fixed-length and ordered responses. Therefore, besides our own dataset1, we also adopted a simulated dataset from the literature. Piech et al. [5] have simulated 5 concepts to cover a fixed-length and ordered sequence of 50 questions and answered by 4000 virtual students, and this dataset was used also to evaluate DKT method in [5], [10].

4.2.1 Dataset 1 Preprocessing for Training

The C++CCH dataset (dataset1) was already scaled and normalized in the form of 0 for false answer and 1 for true. The original dataset consists of 5394 records for 31 questions with 174 unique students. We selected only 4524 responses that belongs to 26 questions ($M = 26$ is the length of the response sequence for each student) of the game as shown

in Table 8. Consequently, we fed all the selected responses into a generation function to generate the inputs matrix in the form of questions and answers $\{Q_s, A_s\}$ as $x_s = \{0,1\}^{2M}$ with output $\{A_{s+1}\}$ as $y_{s+1} = \{0,1\}^M$. Also, we applied the 60% / 20% / 20% split ratio for training, validation, and testing data, respectively.

Table 8. C++ Code Challenge Dataset

Response ID	Player ID	C++ Code Challenge (Q_t)	Response (A_t)
1	1	Q_1	1
2	1	Q_2	0
...	1
26	1	Q_{26}	1
...			
1284	50	Q_{10}	1
...
4523	174	Q_{25}	0
4524	174	Q_{26}	1

4.2.2 Dataset: Simulated-5 (Baseline)

We randomly selected only 174 unique virtual student submissions from Simulated-5 dataset with total of 4524 exercise answers for 26 sequences. The purpose of this selection was to match the size with the C++CCH dataset to evaluate and compare the two datasets on same size. Our objective is to investigate the impact of using a small dataset with fixed-length responses and ordered questions from basic to advance concepts on the prediction performance.

4.2.3 Model Configuration

In our experiments, we applied and evaluated LSTM, biLSTM, GRU, CNN and hybrid RNN-CNN-based models across our dataset C++CCH and Simulated-5 dataset using MATLAB to compare their prediction performance. The same network topologies and settings on all models and datasets were applied. Initially, the proposed Hybrid-DKT topology (see Figure 8) was used to apply the hybrid-RNN-CNN model. For the CNN model we used the same Hybrid-DKT, but we removed the RNN layer and reconnected the input layer directly with the CNN layer. For the LSTM, biLSTM, GRU we removed the CNN layer from the Hybrid-DKT model as well and reconnected the RNN layer with the output layer. Moreover, we adjusted the hidden layers to 98 and batch size to 25 for the RNN single models, and 70 for the CNN and the Hybrid models. We performed 30 training epochs with Adam algorithm [95] to optimize the training. For the CNN layer we adjusted the number of filters to 64, filter size to 5, dropout factor to 0.01 and padding value to “same”.

4.2.4 Design of Experiments

In order to trace the knowledge state of players in real-time during the gameplay and predict the result of the next code challenge along 22 challenges in the game, we trained the model to predict the next code challenge for each sequence length from 5 to 26 as shown below in Table 9. Whereas the first 4 challenges as we indicated in section 3.4.5 the players have to answer them when starting the game.

Table 9. Prediction at Each Sequence Length Input/Output

Sequence Length	Input Sequences (Code Challenges)	Next Code Challenge to be Predicted
5	$\{Q_1, A_1\}, \{Q_2, A_2\}, \{Q_3, A_3\}, \{Q_4, A_4\}$	$\{A_5\}$
6	$\{Q_1, A_1\}, \{Q_2, A_2\}, \{Q_3, A_3\}, \{Q_4, A_4\}, \{Q_5, A_5\}$	$\{A_6\}$
7	$\{Q_1, A_1\}, \{Q_2, A_2\}, \{Q_3, A_3\}, \{Q_4, A_4\}, \{Q_5, A_5\}, \{Q_6, A_6\}$	$\{A_7\}$
...
26	$\{Q_1, A_1\}, \{Q_2, A_2\}, \{Q_3, A_3\}, \dots, \{Q_{25}, A_{25}\}$	$\{A_{26}\}$

In these experiments, we evaluated and compared the prediction performance for each sequence length using the proposed Hybrid-DKT model and the single baseline models LSTM, biLSTM, GRU and CNN in tuple input entry as question and answer $\{Q_t, A_t\}$ on both datasets. Also, we assessed the DKT Missing Sequence Padding method on the trained model.

4.2.5 Results

We evaluated the prediction performance by four metrics: we constructed the confusion matrix and plotted the ROC curve to measure the AUC for each sequence length. We also calculated the prediction accuracy ACC to compare it with the AUC results. Moreover, to simplify the comparison along 22 sequence lengths and 7 prediction models, we calculated the ACC/AUC average of all sequence lengths for each model to make a general comparison indicator between models and datasets. Finally, we considered to compare the AUC prediction performance in four ranges over the sequence over all sequence lengths.

Table 10. AUC/ACC Prediction Performance (**Average** for all Sequence Lengths)

Model	Simulated-5 Dataset		C++CCH Dataset	
	ACC	AUC	ACC	AUC
LSTM	0.752	0.700	0.789	0.786
biLSTM	0.725	0.666	0.786	0.788
GRU	0.743	0.685	0.795	0.794
CNN	0.757	0.703	0.790	0.791
Hybrid (LSTM-CNN)	0.741	0.712	0.786	0.802
Hybrid (biLSTM-CNN)	0.709	0.712	0.784	0.804
Hybrid (GRU-CNN)	0.759	0.716	0.798	0.814

The average performance for all sequence lengths in Table 10 shows that the Hybrid models outperformed the other single models biLSTM, LSTM, GRU and CNN on both datasets (for more details see Table 11). Among the hybrid models, the GRU-CNN model achieved the best prediction performance average with an AUC score of 0.814, higher than the biLSTM-CNN with a 0.804 AUC score and the LSTM-CNN with an AUC score of 0.802, which is expected on our small datasets. For the single models, the GRU model achieved the best average AUC prediction performance with 0.794, followed by CNN and biLSTM with 0.791 and 0.788 respectively, the LSTM model was the last in the list with 0.786 AUC score. The results also revealed that the CNN single model obtained similar performance to the RNNs variants and can be used in the DKT problem. To compare the performance on the dataset level, the C++CCH dataset outperformed the Simulated-5 dataset on all models. This result is expected as the C++CCH dataset contains real submissions and sequence dependency ordered in ascending order from simple concepts to difficult. The best prediction performance for the Simulated-5 dataset was on the Hybrid GRU-CNN with AUC 0.716. Also, the Simulated-5 dataset showed significant differences between ACC and AUC scores on the single models.

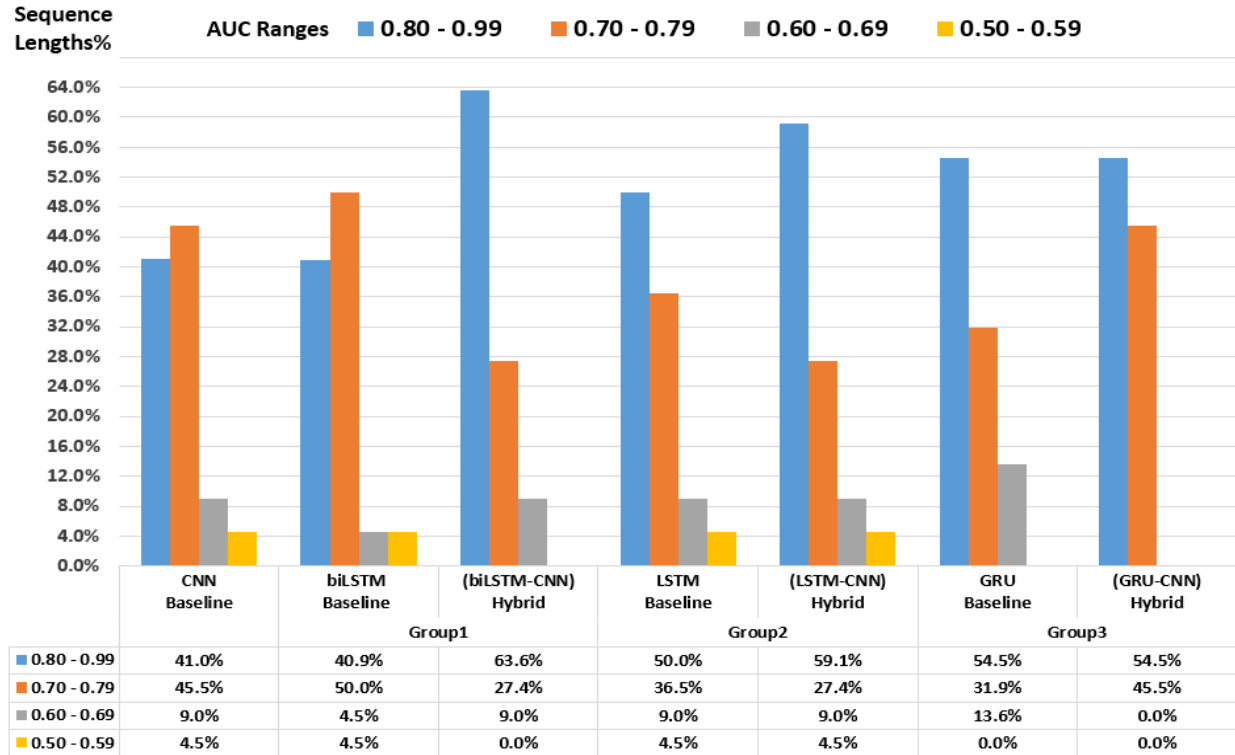


Figure 20. AUC prediction performance range for all 22 sequence lengths

The prediction performance range in Figure 20 shows that along all the 22 sequence lengths the hybrid models achieved the best prediction performance ranges. The combination of the RNNs variants and the CNN network impacts the performance positively with an interesting behavior. Obviously, it leveled-up the overall prediction quality toward increasing the sequence lengths in the AUC interval 0.80 - 0.99 and decreasing those with AUC score less than 0.70. Group 1 shows that 63.6% of the sequence lengths achieved AUC scores between 0.80 and 0.99 in the Hybrid biLSTM-CNN, while the biLSTM baseline percentage was 41%, with significant increase of 23% to the benefit of the Hybrid biLSTM-CNN. Also all the sequence lengths with score less than 0.60 improved and shifted up to the range 0.60 - 0.69. Group 2 shows that around 9% of the sequence lengths shifted up from the AUC range 0.70 - 0.79 to 0.80 - 0.99 in

the Hybrid LSTM-CNN, while the rest ranges below 0.70 remained unchanged. Group 3 shows stability in the range 0.80 - 0.99 with 54.5% for both Hybrid GRU-CNN and GRU baseline, but 13.6% of the sequence lengths shifted from the score range 0.60 - 0.69 to 0.70 - 0.79, so that there are no sequence lengths with AUC performance under the score 0.70 as Figure 22 shows the level-up trend for GRU baseline and Hybrid GRU-CNN.

Table 11. AUC Prediction's Performance on Each Sequence Length for C++CCH Dataset

Sequence Length	biLSTM	LSTM	GRU	CNN	Hybrid biLSTM-CNN	Hybrid LSTM-CNN	Hybrid GRU-CNN
5	0.781	0.875	0.875	0.960	0.938	0.875	0.982
6	0.742	0.742	0.742	0.753	0.717	0.682	0.707
7	0.667	0.633	0.633	0.767	0.733	0.667	0.700
8	0.729	0.750	0.813	0.646	0.667	0.771	0.739
9	0.786	0.775	0.813	0.742	0.852	0.775	0.775
10	0.595	0.500	0.679	0.595	0.821	0.821	0.821
11	0.933	0.933	0.833	0.967	0.933	0.900	0.933
12	0.864	0.808	0.818	0.864	0.808	0.864	0.818
13	0.742	0.703	0.780	0.742	0.709	0.780	0.780
14	0.843	0.843	0.788	0.788	0.843	0.843	0.788
15	0.792	0.792	0.771	0.833	0.833	0.833	0.833
16	0.854	0.958	0.984	0.938	0.896	0.958	0.896
17	0.742	0.742	0.742	0.687	0.742	0.742	0.742
18	0.798	0.843	0.843	0.788	0.843	0.843	0.843
19	0.850	0.850	0.800	0.750	0.800	0.800	0.850
20	0.808	0.808	0.808	0.808	0.808	0.808	0.808
21	0.885	0.923	0.885	0.852	0.885	0.885	0.885
22	0.857	0.821	0.857	0.821	0.821	0.821	0.821
23	0.708	0.750	0.750	0.750	0.750	0.750	0.750
24	0.753	0.697	0.652	0.753	0.697	0.753	0.753
25	0.846	0.846	0.846	0.846	0.775	0.885	0.923
26	0.753	0.707	0.753	0.753	0.808	0.586	0.753

Table 11 and Figure 21 demonstrate the AUC prediction performance trend at each sequence length from 5 to 26 on all models. The results show the impact of the multi-level learning and feature extraction in the proposed Hybrid-DKT model. For example, the

Hybrid GRU-CNN achieved the best prediction performance with top three AUC scores 0.982, 0.933 and 0.923 on the sequence lengths 5, 11, 25 respectively, while the baseline results on the GRU model before applying the hybrid multi-level training were 0.875, 0.833 and 0.846. Similarly, on the same models, the best improvement ratio is 20.9%, 15.5% and 12.2% on the sequence lengths 10, 24 and 5.

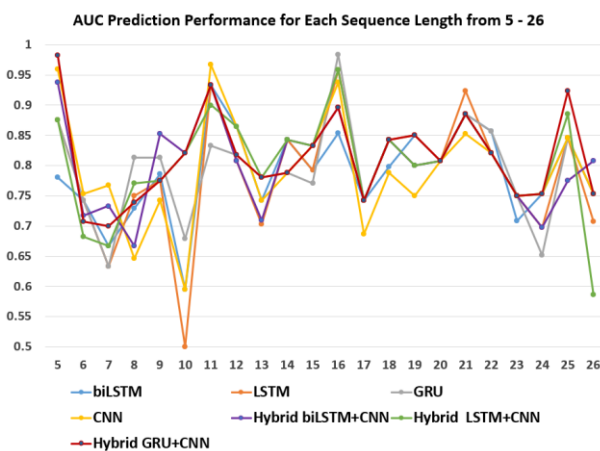


Figure 21. Prediction performance trend for all sequence lengths for each model

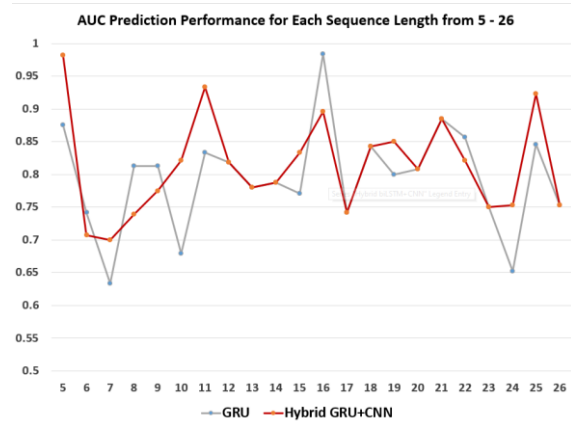


Figure 22. Prediction performance trend for all sequence Lengths for GRU – Hybrid GRU-CNN

For assessing the DKT Missing Sequence Padding (MSP) method across the three hybrid trained models, we recursively predicted the result of the 11th sequence given the results of all available sequences from 1 to 5 and missing results from element 6 to 10. The results in Table 12 revealed the effectiveness of the method with 5 missing values. The Hybrid GRU-CNN achieved the best AUC scores on sequences 7 and 11, while the other sequences are still in acceptable score ranges above 0.75 except sequence 6.

Table 12. Sequence 11 prediction using 5 missing sequence padding method

Sequence Length	Hybrid (biLSTM-CNN) AUC	Hybrid (LSTM-CNN) AUC	Hybrid (GRU-CNN) AUC
6 (missing)	0.673	0.632	0.668
7 (missing)	0.750	0.719	0.813
8 (missing)	0.749	0.792	0.764
9 (missing)	0.786	0.643	0.750
10 (missing)	0.750	0.679	0.786
11	0.752	0.719	0.813

4.2.6 Summary

In summary, we evaluated three hybrid models, three single RNNs variants, and the CNN single model on two datasets. The Hybrid-DKT models obtained better prediction performance over the state-of-the-art DKT method with RNNs single variants LSTM, biLSTM and GRU on both datasets. The GRU-CNN hybrid model achieved the best prediction performance with slightly advantage over the biLSTM-CNN, while the LSTM-CNN was the third on the C++CCH dataset. For the single baseline models on the C++CCH dataset, the GRU obtained the best AUC score followed by the CNN, biLSTM and LSTM. Along all the 22 sequence lengths, the hybrid GRU-CNN obtained the highest improvement ratio with 20.9%, 15.5% and 12.2 over the baseline GRU model for sequence lengths 10, 24 and 5, while the sequence lengths 5, 11, and 25 achieved the highest prediction performance AUC scores 0.982, 0.933 and 0.923 respectively on the GRU-CNN model. Also, the results revealed that the hybrid model have positive impact to level-up the prediction quality toward increasing the sequence lengths in the AUC interval 0.80 - 0.99 by 23%, and decreasing those with AUC score less than AUC 0.70 to 0%. Also, the results revealed the effectiveness of the DKT Missing Sequence Padding

method and the ability to recursively predict far sequences with AUC score 0.813 on 5 missing values.

4.3 Transformer-based Paragraph Generation

4.3.1 Dataset 2: Programming Skills Guide

To fine-tune the Transformer-based paragraph generator on programming skills, we collected data from some free C++ programming skills tutorial websites by a web scraping tool developed in Python. We pre-processed and cleaned the dataset, by reformatting the text and codes in proper statements and paragraphs format, removing unwanted markers, tags, or irrelevant structures.

For paragraphs generation, we investigated and considered three basic formats to be generated. Firstly, we rely on definitions to provide abstract description for a concept. Secondly, explanations are used to describe a concept in more details. Thirdly, source codes give programming code examples related to a concept. Accordingly, we considered for tagging to include the title of the concept with identifying verbs such as define, explain and code as shown in Table 13 and Table 14.

Table 13. Tags Structure

Verb	Tags Structure
Start	< title > [concept title...]
define	< define > definition...
explain	< explain > description...
code	< code > code example...
end code	< endofcode >
end	< endoftitle >

Table 14. Tags Structure Example

Example
<pre> < title > variables < define > variables: A variable definition tells the compiler where and how much storage to create... < explain > variables: A variable definition specifies a data type and contains a list of one or more variables of that type. type must be a valid C++ data type including char... < code > variables: #include <iostream> using namespace std; int main () { // Variable definition: int a, b; int c; float f; < endcode > < endoftitle > </pre>

In the final phase, we transformed the dataset which consists of 2300 text lines into text with tags structure <|title|> as proposed in Table 13 and Table 14 to simplify and guide the model towards certain tags while training, and to generate more relevant and coherent text.

4.3.2 Design of Experiments

We fine-tuned the pre-trained models GPT-2 and GPT-Neo on a new dataset for generating flashcards text in the field of programming skills. We used the GPT-2 model 124M parameters and GPT-Neo 125M parameters, with temperature=0.9 and top_p=0.9. We deployed the experiments in Google Colab environment, a Jupyter-based notebook environment on the cloud.

The paragraphs generation task is formulated as follows: given a pre-defined text as concept title t_1 and target text length M as inputs, generate a paragraph relevant to t_1 as

a sequence of tokens (words/grams) $t_1, t_2, t_s, \dots, t_M$ and not exceeding the given sequence length M . The model generates one gram at a time relevant to the previous tokens' context. Then, the model adds the new generated token after the previously generated token sequence, and the new sequence becomes the new input for the next token and so on until the generated sequence reaches the target length.

We investigated the fine-tuned models on three testing sets with different setups to generate 25 programming skills paragraphs for each testing set. For each paragraph generation we prepared pre-defined text as prompt for the generator and reference text for evaluation, whereas a max sequence length is assigned for each testing set as shown in Table 15. Also, for evaluation, we removed the generated code examples from set 3. We will see in the next section that the evaluation metrics we applied are suitable for the natural language rather than for code syntax.

Table 15. Testing sets: Prompt structure and Output

Testing Set	Prompt Structure	Example	Predicted Output
Set-1	Define tag + title	< define > variables	A definition with max length 50 words.
Set-2	Incomplete Sentence	Global variables are defined outside of...	A complete sentence with max length 24 words
Set-3	Title tag + title	< title > arrays	A paragraph with definition, explanation, and code example with max length 300 words

4.3.3 Results

We considered three metrics: SacreBLEU [96] which is a recent variant of BLEU [97]; ROUGE [98]; and SBERT Sentence Transformers [79]. SacreBLEU and ROUGE are the common metrics in the text generation problem. Both metrics measure the surface

similarity between the sentences and rely on N-gram overlap. Rouge metric focuses on recall, and computes recall, precision and F1 score for the words overlap in 1-gram, 2-gram, and the longest common sequence. In fact, SacreBLEU focuses more on precision score, and computes precision for 1-gram, 2-gram, 3-gram, and 4-gram, in addition to the BLEU Score which is the geometric mean of all four n-gram precision scores. However, SBERT is a recent Transformer-based model introduced to evaluate the semantic similarity. SBERT was adopted in several previous studies such as in [100], [101]. It computes the cosine semantic similarity scores for all possible pairs between two sentence embeddings to measure their semantic similarity.

Table 16. Models Performance

Metrics Model	Testing Set	Sample	SacreBLEU		ROUGE-1			SBERT
			P	BLEU Score	P	R	F1	Semantic Similarity
GPT-2	Set-1	Definitions	0.808	0.760	0.838	0.887	0.843	0.937
	Set-2	Complete Sentences	0.918	0.807	0.933	0.846	0.880	0.930
	Set-3	Paragraphs	0.901	0.750	0.901	0.875	0.876	0.959
GPT-Neo	Set-1	Definitions	0.808	0.769	0.851	0.901	0.847	0.944
	Set-2	Complete Sentences	0.795	0.685	0.800	0.756	0.771	0.860
	Set-3	Paragraphs	0.922	0.849	0.875	0.828	0.843	0.965

The models performance results in Table 16 show that the generators are capable of generating programming skills text with scores higher than 85% for n-gram-based metrics and 90% for the semantic score. The high scores for both N-gram-based metrics and the cosine semantic confirmed that the models-maintained two key factors coherence and meaning of the generated paragraphs (see the generated samples in Table 17). However, limited BLEU scores are obtained on both models, but these results are expected as the BLEU score is a result of computing the geometric mean of all four N-gram precision scores, and the precision decreases when N increases.

Table 17. Comparison between the generated and the reference Texts

Samples	Generated Text	Reference Text
Definition	<i>goto: Transfers control to the labeled statement. Though it is not advised to use goto statement in your C++ to use goto statement</i>	<i>goto: Transfers control to the labeled statement. Though it is not advised to use goto statement in your program...</i>
Complete sentence	<i>A pointer is a variable whose value is the address of another variable.</i>	<i>A pointer is a variable whose value is the address of another variable. Like any variable or constant, you must declare...</i>
Paragraph	<i>const: Constants can be of any of the basic data types and can be divided into Integer Numerals, Floating-Point Numerals, Characters, Strings and Boolean Values. constants are treated just like regular variables except that their values cannot be modified after their definition...</i>	<i>const: Constants refer to fixed values that the program may not alter and they are called literals. Example const int LENGTH = 10; Constants can be of any of the basic data types and can be divided into Integer Numerals, Floating-Point Numerals, Characters, Strings and Boolean Values...</i>

In the testing sets level, the long paragraphs (set-3) achieved the best semantic similarity score higher than 0.95 on both models followed by the short definitions (set-1), and complete sentences (set-2) was the last which were generated without prefix tags. This result confirms two conclusions: firstly, the models achieve better performance when associating prefix tags with the input prompt; secondly, increasing the max-length-size of the generated text has positively influence on the performance, as we assigned it to 300, 50 and 24 for long paragraphs, short definitions, and complete sentences, respectively. Overall, GPT-Neo achieved slightly better performance than GPT-2 in definitions and paragraphs, whereas GPT-2 performed better than GPT-Neo only on the complete sentences with small max-length-size.

Table 18. Generated code samples with prefix tags

Prompt: < code > variable:	Prompt: < define > variable:
<pre> ❏ < code > variable: #include <iostream> using namespace std; int main () { // Variable definition: int a, b; int c; float f; // actual initialization a = 10; b = 20; c = a + b; cout << c << endl ; f = 70.0/3.0; cout << f << endl ; return 0; } < endcode > < endoftitle > </pre>	<pre> ❏ < define > variable: A variable definition tells the compiler < explain > variables: A variable is a variable whose type is < code > variables: #include <iostream> using namespace std; int main () { // Variable definition: int a, b; int c; float f; // actual initialization a = 10; b = 20; c = a + b; } </pre>
Prompt: < code > scope:	Prompt: < code > global variable:
<pre> ❏ < code > scope: #include <iostream> using namespace std; // Global variable declaration: int g = 20; int main () { // Local variable declaration: cout << g; return 0; } output: 10 < endcode > < endoftitle > </pre>	<pre> ❏ < code > global variable: Global variable is defined outside A program can have same name for local and global variables < code > scope: #include <iostream> using namespace std; // Global variable declaration: int g = 20; int main () { // Local variable declaration: int g = 10; cout << g; return 0; } output: 10 < endcode > < endoftitle > </pre>

The generated code samples with prefix tags in Table 18 demonstrate that the models are capable to generate relevant code examples. The annotation tags influenced the generated text to be more relevant and responsive with the prompt. For example, when prompting “<|code|> scope” the model is given the priority to generate relevant code for the keyword “scope” despite there are other details for this keyword such as a definition and explanation. Whereas when prompting “<|code|> global variable” this keyword has only a definition in the dataset. Therefore, the priority is given to define it as the first

relevant text, then the model generated the second relevant text as code tagged with `<|code|>` scope.

Table 19. N-gram Performance Results for The Flashcard Sample

Model	GPT-2				GPT-Neo			
Metrics	SacreBLEU	ROUGE			SacreBLEU	ROUGE		
Contiguous N-gram (word)	Precision	Precision	Recall	F1	Precision	Precision	Recall	F1
1-gram	0.901	0.901	0.875	0.876	0.922	0.875	0.828	0.843
2-gram	0.859	0.866	0.852	0.853	0.902	0.831	0.794	0.805
3-gram	0.844	N/A			0.894	N/A		
4-gram	0.833				0.890			
Longest Common Subsequence (LCS)	N/A	0.878	0.861	0.862	N/A	0.852	0.806	0.822

Table 19 shows the performance trend along different contiguous sequence of N words overlap between the generated text and the reference. Initially, SacreBLEU metric has four n-gram precision scores, whereas ROUGE has two n-gram scores with longest common subsequent score (LCS). Generally, the performance decreases when the n value increases, but the score results demonstrate slightly decreasing which assures the quality of the generated text on different n-grams. For example, in the GPT-Neo model, SacreBLEU metric obtained the best precision among all models and metrics with score 0.922 on 1-gram and the score decreased slightly to 0.890 on 4-gram precision. Also, the model GPT-2 achieved the best F1 score value of 0.876 on 1-gram, and the longest common subsequent slightly decreased the F1 score to 0.862.

4.3.4 Summary

We fine-tuned the GPT-2 and GPT-Neo models with a new dataset to explore the generation of programming skills paragraphs. We evaluated both models by two N-gram-based metrics and a cosine semantic similarity metric. The results revealed that the fine-

tuned models are capable to generate coherent guide text and code examples for programming skills. Both N-gram-based metrics and cosine semantic similarity metrics scores assure the quality and capability of the models to generate coherent text. We also, investigated the influence of associating the prompt text with prefix tags, the results showed that the generated samples with prefix tag achieved better performance. Also, the results demonstrated that the GPT-Neo model achieves better performance on the long text length, whereas GPT-2 performs better on the short text length. Also, the results revealed that the models are capable to detect and recognize the structure of the annotation tags, which is positively influenced the generated text to be more relevant and responsive with the prompt. Furthermore, we investigated the performance trend along different contiguous sequence of N words overlap between the generated and the reference text. The score results demonstrated slightly decreasing in the performance when N value increases, which assures the quality of the generated text on different N-grams.

4.4 Transformer-based Answer Generation

4.4.1 Dataset 3: Programming Skills Summaries

In this part of the experiments our goal is to fine-tune and compare the Transformer-based models T5 and BART to make it capable of summarizing C++ paragraphs to extract and generate answers. Unluckily, we were not able to find any public datasets related to C++ summaries. Instead of that we found datasets for news summaries, scientific papers summaries and some others. Subsequently, we collected C++ questions/answers from different programming skills websites. After cleaning and removing the short answers, we manually provided a summary for each paragraph to form a dataset with 300 C++ paragraphs/answers' summaries covering different topics in C++. Also, in order to train the model to recognize other forms of summarization, we added to the dataset 3700

paragraphs/summaries from the standard dataset `cnn_dailymail` [83] to produce a C++Summary dataset with 4000 pairs of paragraphs/summaries.

4.4.2 Design of Experiments

We fine-tuned the pre-trained models T5 and BART on our new C++ Summary dataset (C++SUMM) to generate answers in the field of programming skills. We used the T5-large and BART-large models, with `max_seq_length`: 128, `train_batch_size`: 8 and 9 epochs. We deployed the experiments in Google Colab environment, and we set the generating parameters as follows: `max_length`: 50, `top_k`: 50, `top_p`: 0.95 and `num_return_sequences` to 1, whereas we applied different values for `num_beams` and `do_sample` as we described below.

Experiments: We conducted in total 18 experiments to generate answers by summarizing the generated paragraphs (C++ guide text) in section 4.3:

- we investigated and compared the performance of the models T5 and BART to generate C++ answers;
- we examined the capability and the performance of the models trained on: (1) both `cnn_dailymail` and C++ summaries together; (2) the C++ summaries only; (3) the `cnn_dailymail` summaries only; to generate C++ answers. For this purpose, we produced 3 different versions of the C++SUMM training dataset DS1, DS2 and DS3;
- we investigated the influence of enabling/disabling the beam search and the greedy search decoding during the answer generation process;
- we compared difference performance metrics of the generated answers with respect to the summarization ratio.

4.4.3 Results

We evaluated each generated answer with respect to a reference answer by the same three metrics used in the previous text generation experiments, which is SacreBLEU [77], ROUGE [79] and SBERT Sentence Transformers [79]. In addition to that, we computed the summary improvement ratio as the number of words between the original paragraph and the answer summary: *Summary Improvement Ratio* (P, S) = $|P-S| / |P|$. We also repeated each experiment 10 times and we computed the average performance for each metric.

Table 20. Average Performance of Generated Answers

Dataset	Model	Search Alg. Enabled	SacreBLEU	ROUGE-1			ROUGE-2			ROUGE-L			SBERT Cosine Semantic Similarity Score	Summary Improvement Ratio %
				P	P	R	F1	P	R	F1	P	R		
DS1: CPP with cnn- dailymail	BART	Sampling	0.550	0.603	0.702	0.624	0.468	0.547	0.484	0.566	0.654	0.584	0.869	44.0
		Beam	0.559	0.613	0.709	0.634	0.478	0.554	0.493	0.576	0.660	0.593	0.873	44.5
		Greedy	0.565	0.604	0.694	0.622	0.467	0.539	0.480	0.567	0.647	0.583	0.869	46.2
	T5	Sampling	0.559	0.595	0.577	0.558	0.415	0.422	0.397	0.553	0.543	0.522	0.832	57.0
		Beam	0.572	0.616	0.595	0.575	0.433	0.439	0.414	0.575	0.563	0.541	0.850	57.4
		Greedy	0.562	0.600	0.587	0.564	0.409	0.427	0.398	0.561	0.556	0.530	0.847	57.2
DS2: CPP	BART	Sampling	0.526	0.571	0.678	0.595	0.421	0.504	0.441	0.536	0.633	0.557	0.856	43.6
		Beam	0.527	0.575	0.681	0.596	0.425	0.507	0.444	0.539	0.634	0.558	0.856	43.3
		Greedy	0.552	0.596	0.695	0.622	0.442	0.522	0.462	0.560	0.646	0.581	0.870	44.8
	T5	Sampling	0.476	0.509	0.517	0.485	0.305	0.314	0.292	0.459	0.466	0.437	0.792	53.4
		Beam	0.473	0.497	0.516	0.482	0.292	0.313	0.287	0.452	0.470	0.439	0.805	53.1
		Greedy	0.464	0.485	0.481	0.456	0.266	0.280	0.257	0.440	0.438	0.415	0.797	56.4
DS3: cnn- dailymail	BART	Sampling	0.392	0.411	0.760	0.515	0.288	0.558	0.365	0.370	0.687	0.464	0.824	16.1
		Beam	0.391	0.410	0.763	0.514	0.286	0.559	0.364	0.367	0.686	0.461	0.827	15.4
		Greedy	0.417	0.437	0.782	0.543	0.319	0.596	0.400	0.401	0.721	0.498	0.837	17.7
	T5	Sampling	0.341	0.353	0.618	0.433	0.212	0.382	0.262	0.302	0.530	0.370	0.773	21.7
		Beam	0.353	0.365	0.641	0.448	0.237	0.426	0.292	0.323	0.565	0.395	0.791	21.1
		Greedy	0.354	0.366	0.642	0.450	0.232	0.417	0.285	0.322	0.562	0.393	0.793	21.0

The average performance of the generated answers in Table 20 shows that the fine-tuned models are capable to summarize paragraphs and generate C++ answers even with a trained dataset consists of only C++ summaries. Whereas mixing the C++ summaries with the cnn-dailymail summaries in the training dataset for the BART model and beam search enabled achieved the best average F1 scores for Rouge-1, Rouge-2, and Rouge-L and cosine semantic similarity score of 0.873 with summarization ratio reached to 44.5%, which is a slight improvement over when enable sampling or greedy search instead of beam on the same model. Also, slightly performance decrease was obtained on the BART model when using the C++ summaries only in the training dataset with scores 0.856 for both beam and sampling, and 0.870 when enabling the greedy search. However, generating answers by training a dataset consists of the cnn-dailymail summaries only showed decreasing in the cosine similarity scores under 0.840 and a significant decreasing in the summarization ratio to less than 22%, which means that the models just re-generating almost the same paragraphs.

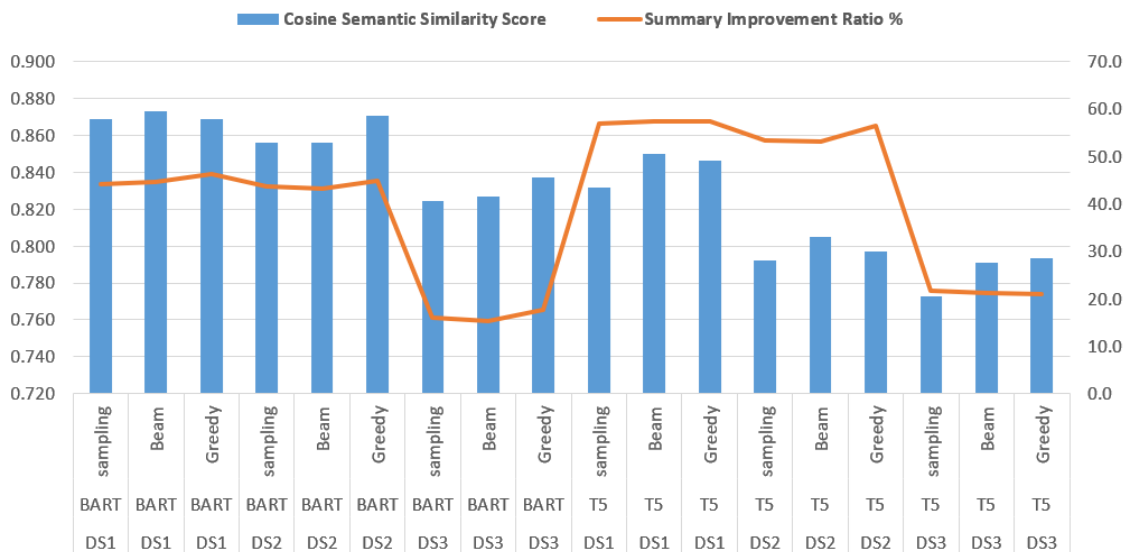


Figure 23. Average Cosine Semantic Similarity Scores and Summary Ratio

In general, the BART model achieved best Rouge and Cosine semantic similarity scores on all datasets, while the T5 achieved the best summarization ratio with 24% increase over the BART model as shown in Figure 23. For the model's generator decoding methods, mostly beam decoding demonstrated better performance scores than the others, whereas greedy search decoding comes next, and random sampling is the last. Despite the greedy search gets the second rank in the quantitative scores, but in reality the greedy search returns the most probable next word, which means it mostly returns same sentences without rephrasing new sentences, as we conducted a qualitative evaluation and the result comes to confirm that as shown in Table 21.

Table 21. Examples of Generated Answers

Generated Answers Beam/Sampling (Summary)	Generated Answers Greedy (Summary)	Reference Answer	Reference Paragraph
<i>A constructor is the class method that initializes an object of that class.</i>	<i>A constructor is a member function that is invoked whenever you create an object; it has the same name as that of the class.</i>	<i>A constructor is a function that is called when an object is created. It is used to initialize the object.</i>	<i>A constructor is defined as a member function that is invoked whenever you create an object; it has the same name as that of the class. There are two types of constructors: Default constructor: This auto-generated constructor does not take any arguments. Parameterized constructor: In this constructor, it can pass arguments.</i>
<i>The break statement is a statement that causes the user to exit an iterative or switch statement.</i>	<i>A break statement is a C or C++ control statement that contains the keyword break and a semicolon (;). It is used to end an iterative or a switch statement</i>	<i>It is a way of exiting a loop (or switch statement) at any point, instead of only at the end, by using the keyword break.</i>	<i>break statement: A C or C++ control statement that contains the keyword break and a semicolon (;). It is used to end an iterative or a switch statement by exiting from it at any point other than the logical end. Control is passed to the first statement after the iteration or switch statement.</i>
<i>void specifies that the function does not return a value.</i>	<i>A void pointer is a pointer which is having no datatype associated with it.</i>	<i>Void keyword specifies that the function does not return a value.</i>	<i>You use the void() return type when you do not want to return any value. void specifies that the function does not return a value. A function with a void return type completes its task and then returns the control to the caller.</i>
<i>Inheritance is the mechanism in which</i>	<i>Inheritance is the mechanism in which you</i>	<i>Inherence is a mechanism in which you can create a</i>	<i>Inheritance is the mechanism in which you can create a new class i.e., child class from the</i>

<i>you can create a new class from the existing class i.e., child class.</i>	<i>can create a new class i.e., child class from the existing class</i>	<i>new class called a child class from the existing parent class</i>	<i>existing class i.e. parent class. This child class is also known as a derived class and the parent class is also called Base class.</i>
--	---	--	--

4.4.4 Summary

We fine-tuned the BART and T5 models with a new C++ Summary dataset (C++SUMM) to summarize and generate answers from the generated paragraphs in section 4.3. We evaluated the model by two N-gram-based metrics, the cosine semantic similarity metric, and we computed the summarization ratio. The results revealed that the fine-tuned models are capable to summarize paragraphs and generate C++ answers with best semantic similarity score 0.873 and best summarization ratio reached to 57.4%. We explored the ability and the performance of the models to generate C++ answers using three different training datasets, the results revealed that combining the C++ summaries with the cnn-dailymail summaries achieved better performance on all metrics than using the C++ summaries alone in the training dataset. Whereas, using a dataset with only the cnn-dailymail summaries downgraded the performance. While using the summarization ratio uncovers the importance of using this metric in the summarization tasks. Despite the models in some cases achieves accepted scores, but in reality, it is regenerating almost the same paragraphs (rows 13-15 in Table 20). Overall, the BART model demonstrated better Rouge and Cosine Semantic scores, whereas the T5 model showed better summarization ratio. Finally, we investigated the influence of using different generator decoding methods, the quantitative metrics scores revealed that the beam decoding obtained the best performance followed by the greedy search decoding. Whereas the qualitative evaluation uncovers that the beam and the sampling decoding methods are better than the greedy as the latter is mostly returns same sentences from a paragraph without rephrasing new sentences. These results emphasize the importance of applying multiple evaluation metrics for the summarization task such as N-gram-based, Semantic-

based and summarization ratio or (compression ratio) as the best performance is a compromise among them. Also, we underline the importance of having a relevant summary dataset so that the model can better recognize the context and achieve better summarization ratio.

4.5 Transformer-based Questions Generation

4.5.1 Dataset 4: C++ Questions/Answers

Our goal was to fine-tune the Transformer-based T5 model to make it capable of generating C++ questions from the answers that generated in the previous section 4.4. Unfortunately, we were not able to find any public datasets related to this topic. The majority of the available question/answer datasets that we found was related to other topics such as Amazon products [102], RACE [103] for English language, Bing search [104], CoQA [105] conversational-style QA dataset, Wikipedia-based Stanford Question Answering Dataset SQuAD [106], and SciQ [107] for science exams (including Physics, Chemistry, Biology, and other disciplines). Therefore, we collected C++ questions/answers from different programming skills websites. After cleaning and removing irrelevant questions, the dataset consists of 500 pairs of questions/answers covering 28 topics in C++ including all the basic programming concepts, OOP, and socket programming. Also, in order to train the model to recognize other forms of questions, we added to the dataset 1500 questions/answers from the datasets RACE, CoQA and SQuAD, and 1000 questions with the supporting answer paragraphs from SciQ science exams dataset, to produce a C++QA dataset with 3000 pairs of question/answer.

4.5.2 Design of Experiments

We fine-tuned the pre-trained model Google T5 on our new C++ Questions/Answers (C++QA) and on SciQ science exams datasets to generate questions in the field of

programming skills and to evaluate the impact of the dataset size. We used the T5-large model, with `max_seq_length`: 128, `train_batch_size`: 8 and 6 epochs. We deployed the experiments in Google Colab environment, and we set the generating parameters as follows: `num_beams`: None, `do_sample`: True, `max_length`: 50, `top_k`: 50, `top_p`: 0.95 and `num_return_sequences` to 1.

Exp. Set-1: We investigated the performance of the model to generate C++ questions for topics not included in the dataset. In real situation, it is difficult to include all topics in a specific domain due to data availability and the needed effort to create many questions/answers for the topic. Also, we needed to examine the capability of the model trained on a subset of the topics to generate questions for the rest of the topics. Therefore, among of 28 C++ topics, we conducted six experiments using 6 different versions of the C++QA training dataset to include 0 topics (i.e., only the SciQ dataset was used), 4 topics, 7 topics, 14 topics, 21 topics and 28 topics. We also created 6 different testing datasets with 60 questions each to evaluate only the ability of the model to generate questions for topics left out from the training dataset (i.e., for 28, 24, 21, 14, 7 and 0 excluded C++ topics).

Exp. Set-2: We investigated the influence of the dataset size on the performance of the generated C++ questions. We evaluated the performance using training datasets with 25%, 50% and 75% of 440 C++ Question/Answers, while the remaining 60 questions of the C++QA dataset were used for testing.

Exp. Set-3: As we were able to have at most only 440 C++ questions available for training, in order to investigate the influence of the dataset size on the performance we had to resort to science questions from the larger SciQ dataset. We started with 440 questions and scaled up to tenfold in ten experiments.

4.5.3 Results

We chose to evaluate each generated question with respect to a reference question by the same three metrics used in the previous text generation experiments, which is SacreBLEU [77], ROUGE [79] and SBERT Sentence Transformers [79]. We also repeated each experiment 10 times and we computed the average performance for each metric as shown in Table 22. The variance is not reported as it was less than 0.001 for all experiments.

Table 22. Performance of C++ Questions Generation

Metrics		SacreBLEU		ROUGE-1				SBERT			
		Precisions		Precisions		Recall		F1		Semantic Similarity (Cosine score)	
Exp.Set-1: Percentage of CPP Topics in the Training DS.		AVG	% Of the best result (row 6)	AVG	% Of the best result (row 6)	AVG	% Of the best result (row 6)	AVG	% Of the best result (row 6)	AVG	% Of the best result (row 6)
1	0% of Topics	0.205	28.1	0.199	26.8	0.430	56.4	0.251	34.0	0.563	64.9
2	15% of Topics	0.503	69.0	0.579	77.9	0.670	87.8	0.595	80.5	0.768	88.5
3	25% of Topics	0.48	65.8	0.582	78.3	0.649	85.1	0.576	77.9	0.789	90.9
4	50% of Topics	0.597	81.9	0.659	88.7	0.648	84.9	0.631	85.4	0.838	96.5
5	75% of Topics	0.702	96.3	0.705	94.9	0.667	87.4	0.666	90.1	0.859	99.0
6	100% of Topics and Questions	0.729	100.0	0.743	100.0	0.763	100.0	0.739	100.0	0.868	100.0
Exp.Set-2: Percentage of the 440 C++ Questions in the Training DS.											
1	75% of Questions	0.757	103.8	0.751	101.1	0.751	98.4	0.734	99.3	0.855	98.5
2	50% of Questions	0.693	95.1	0.729	98.1	0.735	96.3	0.712	96.3	0.837	96.4
3	25% of Questions	0.617	84.6	0.669	90.0	0.672	88.1	0.646	87.4	0.81	93.3

The questions generation performance results in Table 22 show that the fine-tuned model is capable of generating C++ questions even when only 15% of the topics are covered in the training data. The model achieved best semantic similarity cosine score of 0.868 when

all the topics are included in the training dataset, with a slight 1% improvement over when including 75% of the topics (score 0.859). Limited performance decrease was obtained when covering 50% and 25% of the topics with scores 0.838 and 0.789, respectively. However, decreasing under 25% the included topics in the training datasets produced similarity cosine scores of 0.768 and 0.563 for 15% and 0% of the topics, respectively. For the other metrics, results show oscillating scores between 0.5 and 0.7 and this is due to the model rephrasing questions semantically correct but using different words or phrases as shown in Table 23.

Table 23. Examples of Generated Questions

Generated Question	Reference Question
<i>What do you mean by passing by value and passing by reference?</i>	<i>Explain Pass by Value and Pass by Reference</i>
<i>What is the conditional expression?</i>	<i>Define the conditional expression?</i>
<i>What do you call functions that share the same name?</i>	<i>What is function overloading?</i>
<i>What is the keyword enum used for in C++?</i>	<i>Is enum a data type?</i>

For the second experiment set (lower part of Table 22), results evidenced that performance slightly decreases when decreasing the number of C++ questions (all randomly chosen from the available 440 C++QA questions) in the training dataset. For instance, the similarity score decreased to 0.855 when using 75% of the questions, whereas the scores decreased to 0.837 and 0.810 when using 50% and 25% of the available C++QA 440 questions for training, respectively. We examined the effect of increasing the dataset size using the SciQ dataset in which we have enough questions to evaluate up to tenfold.

Table 24. Performance of Science Questions Generation

Metrics		SacreBLEU		ROUGE-1						SBERT	
Exp.Set-3: Percentage of SciQ Science Questions in the Training DS.		Precisions		Precisions		Recall		F1		Semantic Similarity	
		AVG	% Of the best result (row 10)	AVG	% Of the best result (row 10)	AVG	% Of the best result (row 10)	AVG	% Of the best result (row 10)	AVG	% Of the best result (row 10)
1	440 x 1 – 10%	0.609	71.9	0.644	74.7	0.635	74.2	0.621	73.1	0.794	85.9
2	440 x 2 – 20%	0.647	76.4	0.659	76.5	0.659	77.0	0.644	75.8	0.811	87.8
3	440 x 3 – 30%	0.594	70.1	0.64	74.2	0.685	80.0	0.644	75.8	0.814	88.1
4	440 x 4 – 40%	0.75	88.5	0.766	88.9	0.729	85.2	0.734	86.4	0.859	93.0
5	440 x 5 – 50%	0.774	91.4	0.795	92.2	0.811	94.7	0.794	93.4	0.885	95.8
6	440 x 6 – 60%	0.869	102.6	0.866	100.5	0.838	97.9	0.846	99.5	0.907	98.2
7	440 x 7 – 70%	0.776	91.6	0.798	92.6	0.762	89.0	0.768	90.4	0.876	94.8
8	440 x 8 – 80%	0.802	94.7	0.836	97.0	0.828	96.7	0.82	96.5	0.907	98.2
9	440 x 9 – 90%	0.788	93.0	0.813	94.3	0.799	93.3	0.794	93.4	0.886	95.9
10	440 x 10 – 100%	0.847	100.0	0.862	100.0	0.856	100.0	0.85	100.0	0.924	100.0

Table 24 and Figure 24 demonstrate experiment set 3 for the science questions generation. The results confirmed that the model achieves some performance improvements when increasing the number of SciQ questions in the training dataset. For instance, the generated questions at 10X outperformed the others with similarity score 0.924 whereas at 5X and 1X achieved 0.885 and 0.794 similarity scores, respectively. To compare both datasets, results hint that increasing the C++ dataset may improve the performance of the generated questions. However, the science questions needed 4X to 5X of the data to achieve the same 0.868 SBERT semantic similarity score of C++ questions. This is expected due to fact the C++ topics are limited in number, while the science question dataset is much more general as it covers chemistry, biology, physics, and other disciplines, thus it needs more training questions to learn from.

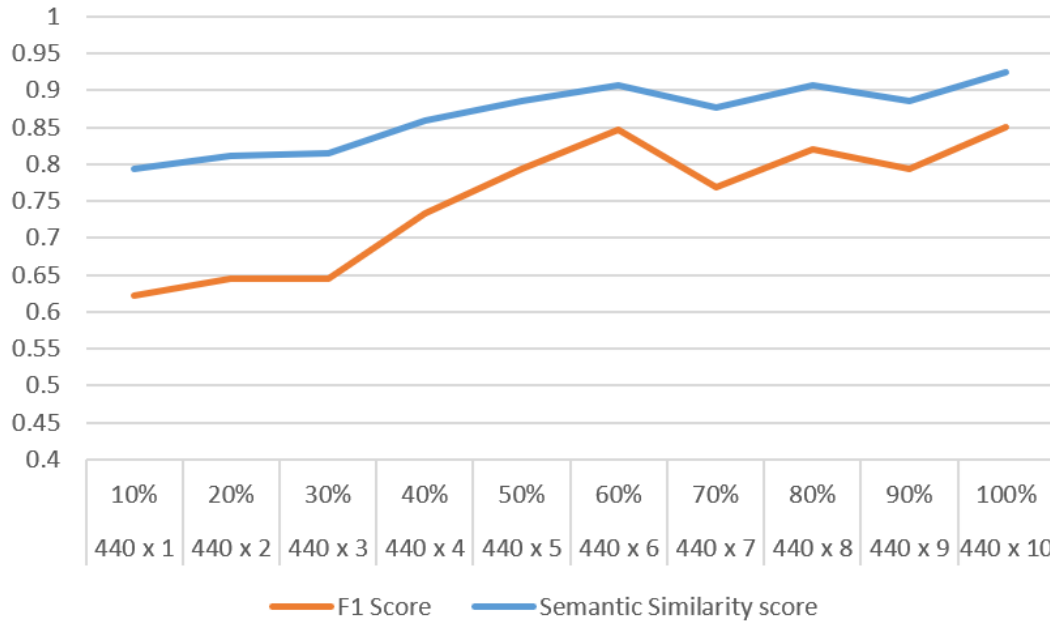


Figure 24. Science Questions Generation Performance Trend

4.5.4 Summary

We fine-tuned the T5 model with a new dataset to generate C++ questions for the flashcards. We evaluated the model by two N-gram-based metrics and a cosine semantic similarity metric. The results revealed that the fine-tuned model is capable of generating coherent questions with a semantic similarity score around 0.87. Also, we investigated the performance of the model to generate C++ questions for topics not included in the training dataset, the results revealed that the model is able to generate coherent questions with including only 15% of the topics in the training data. Furthermore, we investigated the influence of the dataset size on the performance of the generated C++ questions and on the science questions dataset, the results for both datasets show that increasing the number of relevant questions in the training data can improve the performance of the generated questions. Finally, we remark we also explored the ability

of generating C++ questions using a training dataset with only science questions (row 1 in Table 22). Results revealed that the model can generate only up to 50% coherent questions, while the other 50% is incorrect/irrelevant. These results underline the importance of having a relevant Q&A dataset so that the model can recognize the context and the terms' dependencies for a certain subject.

4.6 Flashcards Generation Summary

In order to generate flashcards, we introduced a Transformer-based framework to generate flashcards recommendation for players through three different transformer-based generators (see Figure 13 in sec. 3.4.5). Through three experiment sets in the previous sections (4.3 to generate paragraphs, 4.4 to generate answers from the generated paragraphs and 4.5 to generate questions from the generated answers) we examined and successfully evaluated the Transformer-based Flashcards Generation Framework.

Table 25. Examples of Flashcards Generation

Input Prompt	Generated Paragraph (Model=GPT-2, temperature=0.7, top_p=0.9)	Generated Answer (Model=BART, num_beams=3, do_sample=True, top_p=0.95)	Generated Question (Model=T5, do_sample=True, top_p=0.95)
</title> scope	<i>Scope refers to the area of code where a variable can be accessed, while lifetime refers to the duration of time for which a variable exists. In older programming languages, scope and lifetime are usually the same, but in newer languages like C++ they have different meanings, it is possible for a variable to be out of scope but still alive, which is exploited in the creation of classes.</i>	<i>Scope and lifetime have different meanings. Scope is where a variable can be accessed, but lifetime is how long a variable exists.</i>	<i>What is the difference between variable scope and lifetime?</i>
</title> conversion	<i>Type conversion is changing an entity of one data type into another. This is done to take advantage of certain features of type hierarchies, or to enable operations that were not previously possible. Type conversion can lead to loss of data if it is not done carefully.</i>	<i>Type conversion is changing one data type into another. This can lead to loss of data if it is not done carefully.</i>	<i>What is the type conversion?</i>
</Define> conditional statement	<i>A conditional is a statement that tells the computer to control the flow of a program based on whether a certain condition is met. The most common conditional is the if-else statement, but there are also conditional expressions and switch-case statements.</i>	<i>A conditional is a statement that allows you to control the flow of your program based on whether a condition is true or false.</i>	<i>What is a conditional statement?</i>

Table 25 shows some examples of generating flashcards related to certain topics in C++ by prompting a prefix </title>/</define> followed by keywords, then the model generates a paragraph related to the input keyword, followed by summarizing the paragraph to extract a short answer and finally generating a relevant question.

CHAPTER 5

Conclusions and Future Work

5.1 Conclusions

In this work, we presented a novel Intelligent Serious Games model. This study revealed the effectiveness of combining the Deep Knowledge Tracing (DKT) method and the Transformer-based Recommender as an intelligent model for serious games and for games in general. This research highlighted that the proposed DKT-based hybrid prediction models with the Missing Sequence Padding Recursive (MSP) method are capable to look one or more steps ahead during the gameplay to predict the result of the next missions. Also, results showed that the proposed Transformer-based Recommender enables a generation of proactive coherent-flashcard recommendations so that the player can successfully complete the next mission with high confidence.

We introduced a novel SG conceptual framework (iGDA), which enabled us to identify learning and gaming characteristics to apply and align the intelligent model that we proposed with the player's learning model.

In the intelligent model, we introduced a novel approach in evaluating the DKT method based on each sequence length, within a fixed and ordered series of submissions. This approach is crucial to apply the model on real situation to trace the knowledge state of players/students at each mission. Furthermore, we proposed and evaluated a novel DKT-based hybrid prediction models, and we investigated the influence of combining the LSTM, biLSTM and GRU with CNNs as a multi-layer learning approach. The results revealed the effectiveness of this approach, the models LSTM, biLSTM and GRU are

efficient for long-term and sequential dependencies, but this might not necessarily be useful for all sequence lengths. For this reason, the CNN model as a second layer is efficient for feature extraction and can enhance the performance with discovering hidden patterns and features. Also, the study uncovered the fact of the CNN as a single model can be used in the DKT problem or in the time-series problem in general as it achieved performance close to the RNNs single variants.

Also, we introduced and evaluated a novel method for DKT-Missing Sequence Padding with recursive algorithm. The results revealed the effectiveness of this method to predict far sequences on series with missing values. Therefore, the model can look more steps ahead during the gameplay.

For the second part of the intelligent model, we introduced a novel Transformer-based Recommender System architecture to generate recommendations in programming skills. We also introduced and evaluated a novel Transformer-based framework to generate flashcards in a fully automated process. Using three new programming skills datasets, we fine-tuned GPT-2, GPT-Neo, BART and T5 models to three different generation tasks to generate paragraphs, questions, and answers. We also considered three metrics to evaluate the N-gram overlaps and the semantic similarity to assure the quality and capability of the models. The results revealed that the fine-tuned models are capable to generate coherent paragraphs, questions, answers, and code examples for programming skills.

For the paragraph generating task, the results showed that the generated samples with prefix tag achieved better performance. The results also revealed that the models are capable to detect and recognize the structure of the annotation tags, which positively influenced the generated text to be more relevant and responsive with the prompt. Also, the investigation along different contiguous sequence of n words overlap between the

generated and the reference text demonstrated slightly decreasing in the performance when N value increases, which assures the quality of the generated text on different n-grams.

For evaluating the BART and T5 models to generate answers the results revealed that combining the C++ summaries with the cnn-dailymail summaries achieved better performance on all metrics, also the BART model demonstrated better Rouge and Cosine Semantic scores, whereas the T5 model showed better summarization ratio. Results of evaluating the summarization task emphasize the importance of applying multiple evaluation metrics with summarization ratio as the best performance is a compromise among them.

Generating questions results showed the capability of the model trained on a subset of the topics to generate questions for the rest of the topics as the model needs to capture the context of the knowledge domain. Whereas increasing the number of relevant questions in the training data can improve the performance of the generated questions. However, limited performance can be obtained in the absence of relevant questions in the training dataset.

5.2 Future Work

Overall, we introduced an Intelligent Serious Games (ISG) model, and we focused on assessing and evaluating the efficacy and the feasibility of the ISG components and in particular the intelligent model. Further work will address the influence and the effectiveness of the ISG on the players' attainments by implementing a complete ISG-based game and testing on real players.

The Intelligent model of combining the DKT-based hybrid prediction models and the Transformer-based Recommender is a generic model and can be replicated and

extended in different disciplines and can be integrated with any online learning environment. Also, applying the intelligent model on the entertainment games can positively impact the gaming industry. Other future opportunities to extend this work by generalizing the proposed Transformer-Based Recommender System and expanding the datasets. In addition to extend the Transformer-based framework tailored tasks to include auto short answers' correcting and scoring and generate multiple questions/answers from a certain paragraph.

5.2.1 Integration of The Intelligent Model

The proposed activity diagram in Figure 25 shows how the DKT prediction model and the recommender system impact the gameplay, learning progress, learning achievement and the playing behavior in general. Predicting the next knowledge state of the player occurs just before solving the challenge. Consequently, the recommender acts directly if the prediction result is "failed" to change the playing flow. It generates flashcard recommendations for players before letting them solve the challenge. Also, after solving the challenge, a backward flow of interactions updates learning achievements and predicted achievements with success/fail. The learning progress will be updated considering each (acquired/non-acquired) concept. Figure 26 shows a mock-up screen example for changing of the playing flow.

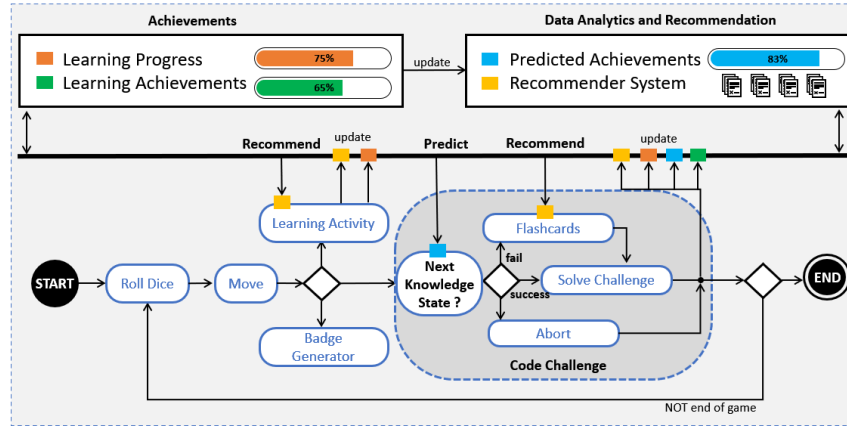


Figure 25. Proposed Activity Diagram for the C++ Code Challenge

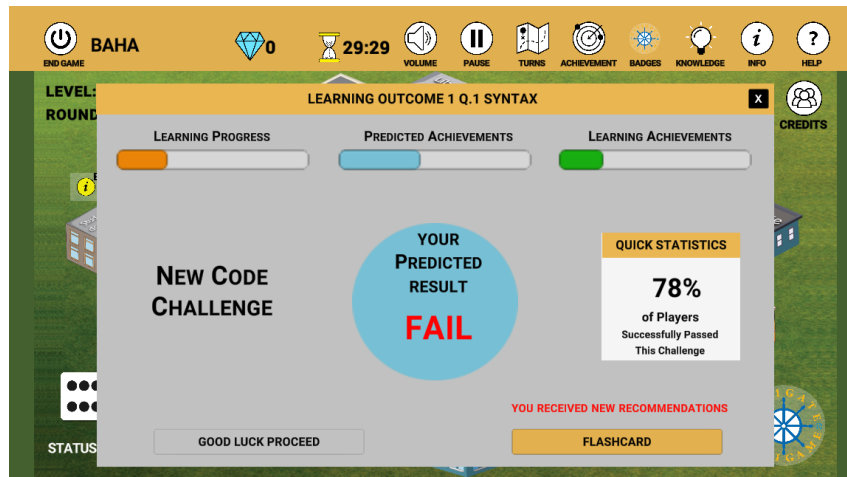


Figure 26. Mock-up of Prediction and Recommendations in Action

Bibliography

- [1] M. Zyda, "From visual simulation to virtual reality to games," *Computer*, vol. 38, no. 9, pp. 25–32, Sep. 2005, doi: 10.1109/MC.2005.297.
- [2] C. Alonso-Fernández, A. Calvo-Morata, M. Freire, I. Martínez-Ortiz, and B. Fernández-Manjón, "Applications of data science to game learning analytics data: A systematic literature review," *Comput. Educ.*, vol. 141, p. 103612, Nov. 2019, doi: 10.1016/j.compedu.2019.103612.
- [3] K. R. Koedinger, E. Brunskill, R. S. J. d. Baker, E. A. McLaughlin, and J. Stamper, "New Potentials for Data-Driven Intelligent Tutoring System Development and Optimization," *AI Mag.*, vol. 34, no. 3, pp. 27–41, Sep. 2013, doi: 10.1609/aimag.v34i3.2484.
- [4] A. Hasanov, T. H. Laine, and T.-S. Chung, "A survey of adaptive context-aware learning environments," *J. Ambient Intell. Smart Environ.*, vol. 11, no. 5, pp. 403–428, 2019, doi: 10.3233/AIS-190534.
- [5] C. Piech *et al.*, "Deep Knowledge Tracing," in *Advances in Neural Information Processing Systems*, 2015, vol. 28. [Online]. Available: <https://proceedings.neurips.cc/paper/2015/file/bac9162b47c56fc8a4d2a519803d51b3-Paper.pdf>
- [6] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Model. User-Adapt. Interact.*, vol. 4, no. 4, pp. 253–278, 1995, doi: 10.1007/BF01099821.
- [7] R. S. J. d. Baker, A. T. Corbett, and V. Aleven, "More Accurate Student Modeling through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing," in *Intelligent Tutoring Systems*, vol. 5091, B. P. Woolf, E. Aimeur, R. Nkambou, and S. Lajoie, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 406–415. doi: 10.1007/978-3-540-69132-7_44.
- [8] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon, "Individualized Bayesian Knowledge Tracing Models," in *Artificial Intelligence in Education*, vol. 7926, H. C. Lane, K. Yacef, J. Mostow, and P. Pavlik, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 171–180. doi: 10.1007/978-3-642-39112-5_18.
- [9] Y. Mao, C. Lin, and M. Chi, "Deep Learning vs. Bayesian Knowledge Tracing: Student Models for Interventions," Oct. 2018, doi: 10.5281/ZENODO.3554691.
- [10] C.-K. Yeung and D.-Y. Yeung, "Addressing two problems in deep knowledge tracing via prediction-consistent regularization," in *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, London United Kingdom, Jun. 2018, pp. 1–10. doi: 10.1145/3231644.3231647.
- [11] A. Vaswani *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, 2017, vol. 30. [Online]. Available: <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>
- [12] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," 2019.
- [13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2018, doi: 10.48550/ARXIV.1810.04805.
- [14] M. Lewis *et al.*, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, 2020, pp. 7871–7880. doi: 10.18653/v1/2020.acl-main.703.
- [15] C. Raffel *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." arXiv, Jul. 28, 2020. Accessed: May 25, 2022. [Online]. Available: <http://arxiv.org/abs/1910.10683>
- [16] P. Smolen, Y. Zhang, and J. H. Byrne, "The right time to learn: mechanisms and optimization of spaced learning," *Nat. Rev. Neurosci.*, vol. 17, no. 2, pp. 77–88, Feb. 2016, doi: 10.1038/nrn.2015.18.

- [17] D. S. Dunn, B. K. Saville, S. C. Baker, and P. Marek, "Evidence-based teaching: Tools and techniques that promote learning in the psychology classroom," *Aust. J. Psychol.*, vol. 65, no. 1, pp. 5–13, Mar. 2013, doi: 10.1111/ajpy.12004.
- [18] C. Kelleher and R. Pausch, "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers," *ACM Comput. Surv.*, vol. 37, no. 2, pp. 83–137, Jun. 2005, doi: 10.1145/1089733.1089734.
- [19] P. Gestwicki and F.-S. Sun, "Teaching Design Patterns Through Computer Game Development," *J. Educ. Resour. Comput.*, vol. 8, no. 1, pp. 1–22, Mar. 2008, doi: 10.1145/1348713.1348715.
- [20] M. Muratet, P. Torguet, F. Viallet, and J. P. Jessel, "Experimental Feedback on Prog&Play: A Serious Game for Programming Practice: M. Muratet et al. / Experimental Feedback on Prog&Play," *Comput. Graph. Forum*, vol. 30, no. 1, pp. 61–73, Mar. 2011, doi: 10.1111/j.1467-8659.2010.01829.x.
- [21] M. Carbonaro, D. Szafron, M. Cutumisu, and J. Schaeffer, "Computer-game construction: A gender-neutral attractor to Computing Science," *Comput. Educ.*, vol. 55, no. 3, pp. 1098–1111, Nov. 2010, doi: 10.1016/j.compedu.2010.05.007.
- [22] A. I. Wang and B. Wu, "Using Game Development to Teach Software Architecture," *Int. J. Comput. Games Technol.*, vol. 2011, pp. 1–12, 2011, doi: 10.1155/2011/920873.
- [23] B. Thabet and F. Zanichelli, "Towards Intelligent Serious Games: Deep Knowledge Tracing with Hybrid Prediction Models," presented at the 2022 17th International Conference on Computer Science & Education (ICCSE), Ningbo, China, Aug. 2022. [Online]. Available: <https://ieeexplore.ieee.org/>
- [24] M. Freire, Á. Serrano-Laguna, B. M. Iglesias, I. Martínez-Ortiz, P. Moreno-Ger, and B. Fernández-Manjón, "Game Learning Analytics: Learning Analytics for Serious Games," in *Learning, Design, and Technology*, M. J. Spector, B. B. Lockee, and M. D. Childress, Eds. Cham: Springer International Publishing, 2016, pp. 1–29. doi: 10.1007/978-3-319-17727-4_21-1.
- [25] C. Alonso-Fernández, I. Martínez-Ortiz, R. Caballero, M. Freire, and B. Fernández-Manjón, "Predicting students' knowledge after playing a serious game based on learning analytics data: A case study," *J. Comput. Assist. Learn.*, vol. 36, no. 3, pp. 350–358, Jun. 2020, doi: 10.1111/jcal.12405.
- [26] *Experience APIs: a new specification for learning technology*. Advanced Distributed Learning. [Online]. Available: <https://xapi.com/>
- [27] A. Calvo-Morata, C. Alonso-Fernandez, M. Freire-Moran, I. Martinez-Ortiz, and B. Fernandez-Manjon, "Game Learning Analytics, Facilitating the Use of Serious Games in the Class," *IEEE Rev. Iberoam. Technol. Aprendiz.*, vol. 14, no. 4, pp. 168–176, Nov. 2019, doi: 10.1109/RITA.2019.2952296.
- [28] L. Chittaro and F. Buttussi, "Learning Safety through Public Serious Games: A Study of 'Prepare for Impact' on a Very Large, International Sample of Players," *IEEE Trans. Vis. Comput. Graph.*, pp. 1–1, 2020, doi: 10.1109/TVCG.2020.3022340.
- [29] K. Graham et al., "Cyberspace Odyssey: A Competitive Team-Oriented Serious Game in Computer Networking," *IEEE Trans. Learn. Technol.*, vol. 13, no. 3, pp. 502–515, Jul. 2020, doi: 10.1109/TLT.2020.3008607.
- [30] T. Mettler and R. Pinto, "Serious Games as a Means for Scientific Knowledge Transfer—A Case From Engineering Management Education," *IEEE Trans. Eng. Manag.*, vol. 62, no. 2, pp. 256–265, May 2015, doi: 10.1109/TEM.2015.2413494.
- [31] L. Chittaro and F. Buttussi, "Assessing Knowledge Retention of an Immersive Serious Game vs. a Traditional Education Method in Aviation Safety," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 4, pp. 529–538, Apr. 2015, doi: 10.1109/TVCG.2015.2391853.
- [32] R. Schulz, B. F. Smaradottir, A. Prinz, and T. Hara, "User-centred Design of a Scenario-based Serious Game: Game-based Teaching of Future Healthcare," *IEEE Trans. Games*, pp. 1–1, 2020, doi: 10.1109/TG.2020.3033437.
- [33] V. Guillén-Nieto and M. Aleson-Carbonell, "Serious games and learning effectiveness: The case of It's a Deal!," *Comput. Educ.*, vol. 58, no. 1, pp. 435–448, Jan. 2012, doi: 10.1016/j.compedu.2011.07.015.

- [34] C. Alonso-Fernandez, A. Calvo, M. Freire, I. Martinez-Ortiz, and B. Fernandez-Manjon, "Systematizing game learning analytics for serious games," in *2017 IEEE Global Engineering Education Conference (EDUCON)*, Athens, Greece, Apr. 2017, pp. 1111–1118. doi: 10.1109/EDUCON.2017.7942988.
- [35] C. Alonso-Fernández, A. Calvo-Morata, M. Freire, I. Martínez-Ortiz, and B. Fernández-Manjón, "Evidence-based evaluation of a serious game to increase bullying awareness," *Interact. Learn. Environ.*, pp. 1–11, 2020.
- [36] A. Calvo-Morata, D. C. Rotaru, C. Alonso-Fernandez, M. Freire-Moran, I. Martinez-Ortiz, and B. Fernandez-Manjon, "Validation of a Cyberbullying Serious Game Using Game Analytics," *IEEE Trans. Learn. Technol.*, vol. 13, no. 1, pp. 186–197, Jan. 2020, doi: 10.1109/TLT.2018.2879354.
- [37] L. Chittaro, "Designing Serious Games for Safety Education: 'Learn to Brace' versus Traditional Pictorials for Aircraft Passengers," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 5, pp. 1527–1539, May 2016, doi: 10.1109/TVCG.2015.2443787.
- [38] M. Callaghan, M. Savin-Baden, N. McShane, and A. G. Eguiluz, "Mapping Learning and Game Mechanics for Serious Games Analysis in Engineering Education," *IEEE Trans. Emerg. Top. Comput.*, vol. 5, no. 1, pp. 77–83, Jan. 2017, doi: 10.1109/TETC.2015.2504241.
- [39] W. Walk, D. Görlich, and M. Barrett, "Design, Dynamics, Experience (DDE): An Advancement of the MDA Framework for Game Design," in *Game Dynamics*, O. Korn and N. Lee, Eds. Cham: Springer International Publishing, 2017, pp. 27–45. doi: 10.1007/978-3-319-53088-8_3.
- [40] "MDA: A Formal Approach to Game Design and Game Research," presented at the Game Developers Conference, 2001-2004, San Jose, 2004.
- [41] B. M. Winn, "The Design, Play, and Experience Framework:," in *Handbook of Research on Effective Electronic Gaming in Education*, R. E. Ferdig, Ed. IGI Global, 2009, pp. 1010–1024. doi: 10.4018/978-1-59904-808-6.ch058.
- [42] K. Robson, K. Plangger, J. H. Kietzmann, I. McCarthy, and L. Pitt, "Is it all a game? Understanding the principles of gamification," *Bus. Horiz.*, vol. 58, no. 4, pp. 411–420, Jul. 2015, doi: 10.1016/j.bushor.2015.03.006.
- [43] A. Pendleton and J. Okolica, "Creating Serious Games with the Game Design Matrix," in *Games and Learning Alliance*, vol. 11899, A. Liapis, G. N. Yannakakis, M. Gentile, and M. Ninaus, Eds. Cham: Springer International Publishing, 2019, pp. 530–539. doi: 10.1007/978-3-030-34350-7_51.
- [44] L. Wang, A. Sy, L. Liu, and C. Piech, "Deep Knowledge Tracing On Programming Exercises," in *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, Cambridge Massachusetts USA, Apr. 2017, pp. 201–204. doi: 10.1145/3051457.3053985.
- [45] Z. Wang, X. Feng, J. Tang, G. Y. Huang, and Z. Liu, "Deep Knowledge Tracing with Side Information," in *Artificial Intelligence in Education*, vol. 11626, S. Isotani, E. Millán, A. Ogan, P. Hastings, B. McLaren, and R. Luckin, Eds. Cham: Springer International Publishing, 2019, pp. 303–308. doi: 10.1007/978-3-030-23207-8_56.
- [46] L. Sha and P. Hong, "Neural Knowledge Tracing," in *Brain Function Assessment in Learning*, vol. 10512, C. Frasson and G. Kostopoulos, Eds. Cham: Springer International Publishing, 2017, pp. 108–117. doi: 10.1007/978-3-319-67615-9_10.
- [47] D. Hooshyar, Y.-M. Huang, and Y. Yang, "GameDKT: Deep knowledge tracing in educational games," *Expert Syst. Appl.*, vol. 196, p. 116670, Jun. 2022, doi: 10.1016/j.eswa.2022.116670.
- [48] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
- [49] A. Aziz Sharfuddin, Md. Nafis Tihami, and Md. Saiful Islam, "A Deep Recurrent Neural Network with BiLSTM model for Sentiment Classification," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, Sylhet, Sep. 2018, pp. 1–4. doi: 10.1109/ICBSLP.2018.8554396.
- [50] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," 2014, doi: 10.48550/ARXIV.1409.1259.

- [51] S. Yang, X. Yu, and Y. Zhou, "LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example," in *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, Shanghai, China, Jun. 2020, pp. 98–101. doi: 10.1109/IWECAI50956.2020.00027.
- [52] S. Kiranyaz, T. Ince, O. Abdeljaber, O. Avci, and M. Gabbouj, "1-D Convolutional Neural Networks for Signal Processing Applications," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, United Kingdom, May 2019, pp. 8360–8364. doi: 10.1109/ICASSP.2019.8682194.
- [53] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, Antalya, Aug. 2017, pp. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.
- [54] K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," 2014, doi: 10.48550/ARXIV.1406.1078.
- [55] M. A. Niculescu, S. Ruseti, and M. Dascalu, "RoGPT2: Romanian GPT2 for Text Generation," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, Washington, DC, USA, Nov. 2021, pp. 1154–1161. doi: 10.1109/ICTAI52525.2021.00183.
- [56] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-Training," presented at the OpenAI, 2018.
- [57] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," 2020, doi: 10.48550/ARXIV.2005.14165.
- [58] Y. Qu, P. Liu, W. Song, L. Liu, and M. Cheng, "A Text Generation and Prediction System: Pre-training on New Corpora Using BERT and GPT-2," in *2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, Beijing, China, Jul. 2020, pp. 323–326. doi: 10.1109/ICEIEC49280.2020.9152352.
- [59] J.-S. Lee and J. Hsiang, "Patent claim generation by fine-tuning OpenAI GPT-2," *World Pat. Inf.*, vol. 62, p. 101983, Sep. 2020, doi: 10.1016/j.wpi.2020.101983.
- [60] W. Antoun, F. Baly, and H. Hajj, "AraGPT2: Pre-Trained Transformer for Arabic Language Generation," 2020, doi: 10.48550/ARXIV.2012.15520.
- [61] J. van Stegeren and J. Myśliwiec, "Fine-tuning GPT-2 on annotated RPG quests for NPC dialogue generation," in *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*, Montreal QC Canada, Aug. 2021, pp. 1–8. doi: 10.1145/3472538.3472595.
- [62] J.-S. Lee and J. Hsiang, "PatentTransformer-2: Controlling Patent Text Generation by Structural Metadata," 2020, doi: 10.48550/ARXIV.2001.03708.
- [63] T. Klein and M. Nabi, "Learning to Answer by Learning to Ask: Getting the Best of GPT-2 and BERT Worlds," *ArXiv191102365 Cs*, Nov. 2019, Accessed: Apr. 20, 2022. [Online]. Available: <http://arxiv.org/abs/1911.02365>
- [64] R. Zellers *et al.*, "Defending Against Neural Fake News," in *Advances in Neural Information Processing Systems*, 2019, vol. 32. [Online]. Available: <https://proceedings.neurips.cc/paper/2019/file/3e9f0fc9b2f89e043bc6233994dfcf76-Paper.pdf>
- [65] A. R. Fabbri, W. Kryściński, B. McCann, C. Xiong, R. Socher, and D. Radev, "SummEval: Re-evaluating Summarization Evaluation," *Trans. Assoc. Comput. Linguist.*, vol. 9, pp. 391–409, Apr. 2021, doi: 10.1162/tacl_a_00373.
- [66] R. Zhang, J. Guo, L. Chen, Y. Fan, and X. Cheng, "A Review on Question Generation from Natural Language Text," *ACM Trans. Inf. Syst.*, vol. 40, no. 1, pp. 1–43, Jan. 2022, doi: 10.1145/3468889.
- [67] V. Pyatkin, P. Roit, J. Michael, R. Tsarfaty, Y. Goldberg, and I. Dagan, "Asking It All: Generating Contextualized Questions for any Semantic Role." arXiv, Sep. 10, 2021. Accessed: May 27, 2022. [Online]. Available: <http://arxiv.org/abs/2109.04832>

- [68] K. Grover, K. Kaur, K. Tiwari, Rupali, and P. Kumar, "Deep Learning Based Question Generation Using T5 Transformer," in *Advanced Computing*, vol. 1367, D. Garg, K. Wong, J. Sarangapani, and S. K. Gupta, Eds. Singapore: Springer Singapore, 2021, pp. 243–255. doi: 10.1007/978-981-16-0401-0_18.
- [69] X. Zhao, F. Xiao, H. Zhong, J. Yao, and H. Chen, "Condition Aware and Revise Transformer for Question Answering," in *Proceedings of The Web Conference 2020*, Taipei Taiwan, Apr. 2020, pp. 2377–2387. doi: 10.1145/3366423.3380301.
- [70] H. Ngai, Y. Park, J. Chen, and M. Parsapoor, "Transformer-Based Models for Question Answering on COVID19," 2021, doi: 10.48550/ARXIV.2101.11432.
- [71] S. G. Aithal, A. B. Rao, and S. Singh, "Automatic question-answer pairs generation and question similarity mechanism in question answering system," *Appl. Intell.*, vol. 51, no. 11, pp. 8484–8497, Nov. 2021, doi: 10.1007/s10489-021-02348-9.
- [72] D. C. L. Tsai, W. J. W. Chang, and S. J. H. Yang, "Short Answer Questions Generation by Fine-Tuning BERT and GPT-2," in *29th International Conference on Computers in Education Conference, ICCE 2021 - Proceedings*, Nov. 2021, pp. 509–515.
- [73] W. Qi *et al.*, "ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Online, 2020, pp. 2401–2410. doi: 10.18653/v1/2020.findings-emnlp.217.
- [74] G. Kurdi, J. Leo, B. Parsia, U. Sattler, and S. Al-Emari, "A Systematic Review of Automatic Question Generation for Educational Purposes," *Int. J. Artif. Intell. Educ.*, vol. 30, no. 1, pp. 121–204, Mar. 2020, doi: 10.1007/s40593-019-00186-y.
- [75] T. Ayadat, D. Ahmed, S. Chowdhury, and A. Asiz, "Measurable performance indicators of student learning outcomes: a case study," *Glob. J. Eng. Educ.*, vol. 22, p. 40, Feb. 2020.
- [76] L. K. Gudeva, V. Dimova, N. Daskalovska, and F. Trajkova, "Designing Descriptors of Learning Outcomes for Higher Education Qualification," *Procedia - Soc. Behav. Sci.*, vol. 46, pp. 1306–1311, 2012, doi: 10.1016/j.sbspro.2012.05.292.
- [77] R. T. Hays, "The Effectiveness of Instructional Games: A Literature Review and Discussion:," Defense Technical Information Center, Fort Belvoir, VA, Nov. 2005. doi: 10.21236/ADA441935.
- [78] R. Garris, R. Ahlers, and J. E. Driskell, "Games, Motivation, and Learning: A Research and Practice Model," *Simul. Gaming*, vol. 33, no. 4, pp. 441–467, Dec. 2002, doi: 10.1177/1046878102238607.
- [79] J. Schell, *The art of game design: a book of lenses*, Third edition. Boca Raton: Taylor & Francis, a CRC title, part of the Taylor & Francis imprint, a member of the Taylor & Francis Group, the academic division of T&F Informa, plc, 2019.
- [80] L. S. Ferro, "The Game Element and Mechanic (GEM) framework: A structural approach for implementing game elements and mechanics into game experiences," *Entertain. Comput.*, vol. 36, p. 100375, Jan. 2021, doi: 10.1016/j.entcom.2020.100375.
- [81] J. Hamari and V. Eranti, "Framework for Designing and Evaluating Game Achievements," Sep. 2011, vol. Vol. 10, No. 1.224, p. 9966.
- [82] I. Lewis, K. de Salas, and L. Wells, "Features of Achievement systems," in *Proceedings of CGAMES'2013 USA*, Louisville, KY, Jul. 2013, pp. 66–73. doi: 10.1109/CGames.2013.6632608.
- [83] F. Zanichelli, M. Encheva, B. Thabet, A. M. Tammamo, and G. Conti, "Serious games for Information Literacy: assessing learning in the NAVIGATE Project," presented at the 17th Italian Research Conference on Digital Libraries, Padova, Italy, Feb. 2021. [Online]. Available: <http://ceur-ws.org/Vol-2816/paper7.pdf>
- [84] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. WOO, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," in *Advances in Neural Information Processing Systems*, 2015, vol. 28. [Online]. Available: <https://proceedings.neurips.cc/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf>

- [85] J. Zhang, Y. Li, J. Tian, and T. Li, "LSTM-CNN Hybrid Model for Text Classification," in *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing, Oct. 2018, pp. 1675–1680. doi: 10.1109/IAEAC.2018.8577620.
- [86] M. Alhussein, K. Aurangzeb, and S. I. Haider, "Hybrid CNN-LSTM Model for Short-Term Individual Household Load Forecasting," *IEEE Access*, vol. 8, pp. 180544–180557, 2020, doi: 10.1109/ACCESS.2020.3028281.
- [87] A. Agga, A. Abbou, M. Labbadi, Y. E. Houm, and I. H. Ou Ali, "CNN-LSTM: An efficient hybrid deep learning architecture for predicting short-term photovoltaic power production," *Electr. Power Syst. Res.*, vol. 208, p. 107908, Jul. 2022, doi: 10.1016/j.epsr.2022.107908.
- [88] S. Oren, C. Willerton, and J. Small, "Effects of Spaced Retrieval Training on Semantic Memory in Alzheimer's Disease: A Systematic Review," *J. Speech Lang. Hear. Res.*, vol. 57, no. 1, pp. 247–270, Feb. 2014, doi: 10.1044/1092-4388(2013/12-0352).
- [89] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds., *Recommender Systems Handbook*. Boston, MA: Springer US, 2011. doi: 10.1007/978-0-387-85820-3.
- [90] G. F. Tondello, R. Orji, and L. E. Nacke, "Recommender Systems for Personalized Gamification," in *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, Bratislava Slovakia, Jul. 2017, pp. 425–430. doi: 10.1145/3099023.3099114.
- [91] M. Amoretti, L. Belli, and F. Zanichelli, "UTravel: Smart Mobility with a Novel User Profiling and Recommendation Approach," *Pervasive Mob. Comput.*, vol. 38, pp. 474–489, Jul. 2017, doi: 10.1016/j.pmcj.2016.08.008.
- [92] P. K. Agarwal and P. M. Bain, *Powerful teaching: Unleash the science of learning*. John Wiley & Sons, 2019.
- [93] S. Prata, *C++ primer plus*, 6th ed. Upper Saddle River, NJ: Addison-Wesley, 2012.
- [94] "ASSISTments' skill builder." [Online]. Available: <https://sites.google.com/site/assistmentsdata/datasets>
- [95] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *ArXiv14126980 Cs*, Jan. 2017, Accessed: Jan. 29, 2022. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [96] M. Post, "A Call for Clarity in Reporting BLEU Scores," in *Proceedings of the Third Conference on Machine Translation: Research Papers*, Belgium, Brussels, 2018, pp. 186–191. doi: 10.18653/v1/W18-6319.
- [97] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, Philadelphia, Pennsylvania, 2001, p. 311. doi: 10.3115/1073083.1073135.
- [98] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Text Summarization Branches Out*, Barcelona, Spain, Jul. 2004, pp. 74–81. [Online]. Available: <https://aclanthology.org/W04-1013>
- [99] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *ArXiv190810084 Cs*, Aug. 2019, Accessed: Apr. 24, 2022. [Online]. Available: <http://arxiv.org/abs/1908.10084>
- [100] R. Devika, S. Vairavasundaram, C. S. J. Mahenthara, V. Varadarajan, and K. Kotecha, "A Deep Learning Model Based on BERT and Sentence Transformer for Semantic Keyphrase Extraction on Big Social Data," *IEEE Access*, vol. 9, pp. 165252–165261, 2021, doi: 10.1109/ACCESS.2021.3133651.
- [101] B. Dang, T.-T. Dang, and L.-M. Nguyen, "SubTST: A Combination of Sub-word Latent Topics and Sentence Transformer for Semantic Similarity Detection:," in *Proceedings of the 14th International Conference on Agents and Artificial Intelligence*, Online Streaming, --- Select a Country ---, 2022, pp. 91–97. doi: 10.5220/0010775100003116.
- [102] Amazon, *Open Data on AWS Datasets*. [Online]. Available: <https://registry.opendata.aws/>
- [103] RACE English examinations in China Dataset. [Online]. Available: <http://www.cs.cmu.edu/~glai1/data/race/>

- [104] *MS MARCO Datasets*. [Online]. Available: <https://microsoft.github.io/msmarco/>
- [105] *CoQA Conversational Question Answering dataset*. [Online]. Available: <https://stanfordnlp.github.io/coqa/>
- [106] *Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset of Wikipedia articles*. [Online]. Available: <https://rajpurkar.github.io/SQuAD-explorer/>
- [107] *SciQ dataset for science exam questions about Physics, Chemistry and Biology, among others*. [Online]. Available: <https://allenai.org/data/sciq>