6-20-2024

BERT-Boosted Adaptive Learning

Dibyanshu Chatterjee
dc7017@rit.edu

Rochester Institute of Technology
Department of Computer Science
Master of Science

Thesis Committee

Chair:	Dr. Zachary Butler
Reader:	Dr. Carlos R. Rivero
Observer:	Dr. Xumin Liu

Chair: **Reader:**

Dr. Zachary Butler Dr. Carlos R. Rivero

Observer:

Dr. Xumin Liu

BERT-BOOSTED Adaptive Learning

by

Dibyanshu Chatterjee

Date of Approval - 06/20/2024

Submitted to the
B. Thomas Golisano College of Computing and Information Sciences
Department of Computer Science
in partial fulfillment of the requirements for the
Master of Science Degree in Computer Science
at the Rochester Institute of Technology

Abstract

Knowledge Tracing is a machine learning technique that aids in monitoring a learner's knowledge in a given learning environment. This thesis introduces a unified approach to adaptive learning, specifically with the use of Knowledge Tracing (KT) and other relevant machine learning models, that can aid in creating a personalized learning environment for a learner on an Intelligent Tutoring System (ITS). While contemporary ITS exemplars, such as Knewton and Adaptemy, strive to provide seamless adaptive learning experience, most such ITS platforms do so by employing individualized models, each tailored to distinct tasks, for example, some models may predict correctness of students' next responses, while others track progress in skill attainment by tracing interactions within the ITS. However, the development of these models necessitates extensive experimentation to optimize input attributes for practical application, alongside the exploration of optimal model architectures. In addition, certain Knowledge Tracing (KT) models, such as RA-BKT and RA-ANN, incorporate metacognitive inputs to enhance the prediction accuracy of student responses. Conversely, models like Deep knowledge tracing with transformers (DKTT) emphasize treating data as a sequence-to-sequence problem and prioritize attributes such as time to improve correctness prediction. However, the multitude of available models underscores the challenge of determining the optimal model for specific use cases, particularly regarding

input attributes availability/requirements posed by ITS. To address this issue, this thesis examines the development of a comprehensive framework named: "BERT-Boosted Knowledge Tracing", which will be able to deal with multitude of KT datasets and spawn off KT or other relevant adaptive learning models as output through its unified architecture. This framework could be imagined as a tool that an ITS can use to readily train required KT models and the framework could be set to train models that would be most relevant to a specific ITS use. The focus in this study would be to explore the power of BERT to create such models and to be at the core of such a framework. All experiments set would be through an example framework which would warrant the use of metacognitive inputs to train KT models. This is done to help in assessing the model's performance against other such existing models (RA-BKT and RA-ANN) and also to emulate few of the scenarios where having such a framework at disposal could be of great help to create an ensemble of relevant models that can aid in creating an Adaptive Learning experience in an ITS.

Creating such a framework and the resultant models poses the challenge of establishing robust evaluation criteria, given the scarcity of comparable works. To address this, several models are trained based on three distinct datasets, with an aim to showcase such a framework's capabilities in: being able to handle different kind of input attributes, training models based on uniquely specified target attributes, creating models that can be trained on very less training data and also to create models that are domain adaptive.

Through rigorous experimentation and evaluation, this thesis showcases the efficacy of the proposed framework in revolutionizing KT model development. By providing a comprehensive solution that seamlessly integrates BERT-based contextual learning with adaptable training methodologies and the incorporation of metacognitive insights, this research significantly advances the field of Knowledge Tracing.

Acknowledgments

I would like to express my sincere gratitude to Dr. Zachary Butler for his invaluable guidance, unwavering support, and insightful feedback throughout the course of this research. His expertise, encouragement, and mentorship have been instrumental in shaping this thesis.

I am also deeply thankful to Rochester Institute of Technology (RIT) for providing the resources and environment conducive to academic growth and research excellence.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Knowledge tracing, a pivotal concept in the realm of educational technology, holds the key to unlocking personalized learning experiences and optimizing educational outcomes. As we traverse the digital landscape, where vast amounts of data are generated and processed each day, the ability to accurately gauge and guide an individual's learning journey becomes increasingly imperative. In this context, the advent of sophisticated machine learning techniques, such as BERT, has ushered in a new era of knowledge tracing, promising enhanced adaptability and efficacy [12].

The fundamental question arises: why is knowledge tracing useful? The answer lies in its potential to revolutionize education by tailoring instruction to the unique needs and capabilities of each learner. Traditional educational approaches often rely on standardized assessments and one-size-fits-all curricula, which may inadvertently overlook individual differences in learning pace, style, and comprehension. Knowledge tracing offers a paradigm shift by leveraging data-driven insights to construct personalized learning pathways, thus maximizing student engagement, retention, and mastery.

Furthermore, in an era characterized by rapid technological advancements and evolving pedagogical paradigms, the ability to adapt to diverse learning domains and modalities is paramount. The flexibility inherent in novel frameworks, such as the one developed using BERT in this study, underscores the potential of knowledge tracing to transcend disciplinary boundaries and accommodate diverse educational contexts. Whether in traditional classroom settings, online courses, or immersive learning environments, the ability to accurately assess and monitor an individual's knowledge acquisition journey empowers educators to make informed instructional decisions and foster continuous improvement.

In this report, a novel framework for knowledge tracing is presented that harnesses the power of BERT, a state-of-the-art natural language processing model, to enhance adaptability and effectiveness of Knowledge Tracing (KT) and to take a step towards a generalized KT framework which is flexible over various input features. Through empirical validation and real-world application, the utility of the approach is demonstrated in facilitating domain-adaptive and flexible knowledge tracing. By elucidating the significance of knowledge tracing and showcasing its practical implications, this report endeavors to contribute to the ongoing discourse on leveraging technology to optimize learning experiences and nurture the next generation of learners.

1.1 Introduction to ITS and Knowledge Tracing

1.1.1 Intelligent Tutoring System (ITS)

An ITS is essentially a system that is meant to imitate human tutors. Unlike traditional classroom settings, where a single instructor caters to many students with diverse learning needs, ITS offer a tailored learning experience by adapting to individual students' abilities, preferences, and progress. This personalization is achieved through the integration of artificial intelligence (AI), cognitive psychology, and educational theory.

ITS are designed to emulate the one-on-one interaction between a student and a human tutor. They typically incorporate several key components: a domain model, which contains the subject matter to be taught; a student model, which tracks the learner's knowledge, skills, and attributes; a tutoring model, which determines the pedagogical strategies to be used; and an interface, which facilitates interaction between the system and the user. By continuously analyzing student responses and behavior, ITS can diagnose misconceptions, provide immediate feedback, and adjust the difficulty level of tasks to match the learner's current understanding.

The benefits of ITS are manifold. They can offer scalable and cost-effective educational support, making high-quality tutoring accessible to a wider audience. Moreover, use of ITS in convectional educational institutions have shown promising methodologies through which it can collaborate with instructors to provide personalized learning experience and knowledge monitoring for learners.

1.1.2 Knowledge Tracing (KT)

Knowledge Tracing (KT) is a task of modeling student knowledge over time. At its core, knowledge tracing essentially uses either mathematical formulas, machine learning algorithms, or AI algorithms, to trace learners' interactions in an ITS. The Knowledge Tracing models serve as an integral part of an ITS, as through KT, an ITS identifies the knowledge state of a learner in a given learning environment.

Ever since the introduction of Bayesian Knowledge Tracing (BKT) [3], which is a hidden Markov Model (HMM), the field of KT has seen significant enhancements. The recent innovations around KT involve the use of transformer models and LLMs to identify the mastery of a knowledge component shown by a learner in a ITS learning environment. The evolution of the KT models begs a question about the factors that urges researchers to develop new KT models. The few key prompts that motivate the development of new KT models are discussed below:

- Improved Prediction Accuracy:
  - Enhanced Algorithms: Advances in machine learning and deep learning techniques often lead to more accurate models. Researchers explore these advancements to create more sophisticated KT models that can better predict learner performance.
  - Handling Complexity: New models may better handle the complexity of learning processes, including factors like the interdependence of skills, varying difficulty of questions, and learner-specific traits.
- Generalization and Robustness:
  - Different Contexts: Educational settings vary widely, and a model that works well in one context may not perform as well in another. New models are developed to ensure robustness and generalizability across different subjects, age groups, and learning environments.
  - Adaptability: Some models are designed to be more adaptable to individual learner profiles, thus providing personalized predictions and interventions.
- Incorporation of New Data Types and Features:
  - Multimodal Data: With the increasing availability of diverse data types (e.g., eye-tracking, interaction logs, physiological data), new

models are developed to incorporate these additional features, potentially leading to better predictions.

  - Contextual Factors: New models might incorporate contextual factors such as time of day, learner's emotional state, or even social interactions, which can influence learning outcomes.

- Explainability and Interpretability:

  - Transparent Models: There is a growing demand for models that are not just accurate but also interpretable. Researchers are motivated to develop models that provide insights into the learning process and can explain why a particular prediction was made.
  - Actionable Insights: Teachers and educators benefit from models that offer actionable insights, helping them understand and support learners more effectively.

- Scalability and Efficiency:

  - Computational Efficiency: Some new models aim to reduce computational requirements, making them more feasible for real-time applications in large-scale educational settings.
  - Scalability: Models that can scale effectively to handle large datasets or be deployed across various platforms and institutions are highly sought after.

- Addressing Limitations of Existing Models:

  - Overfitting and Underfitting: Researchers develop new models to address issues of overfitting (where a model performs well on training data but poorly on new data) and underfitting (where a model fails to capture the underlying trends in the data).
  - Bias and Fairness: Ensuring that models are fair and unbiased across different demographics is a critical area of research. New models might be designed to mitigate biases found in previous approaches.

- User and Stakeholder Feedback:

  - Practical Utility: Feedback from educators, learners, and other stakeholders can drive the development of new models that better meet the practical needs of users.

- User Experience: Enhancing the usability and integration of KT models within existing educational technologies can prompt the development of more user-friendly models.

1.2 The Problem Statement

The goal of this thesis is to develop a unified framework that can train Knowledge Tracing (KT) models using BERT on any balanced KT dataset, regardless of the specific attributes in the dataset. This framework aims to predict not only traditional targets such as the correctness of a learner's response but also additional meaningful targets, such as a learner's confidence level, to enhance the functionality of an Intelligent Tutoring Systems (ITS). By addressing the complexities of learning processes and incorporating multimodal data, this framework seeks to create robust KT models. Generalizing KT model training methods can help improve the adaptability and scalability of educational technologies, allowing them to perform well across diverse subjects, age groups, and learning environments. The effectiveness of the framework will be demonstrated using a Reflection Assistant-like ITS environment, which is designed to address shallow learning by considering students' metacognitive levels. This will prove the framework's proficiency in predicting a student's response correctness and their confidence before answering a question. These predictions will help evaluate the learner's awareness (KMA) and outlook (KMB), although this evaluation is not the focus of this study and is derived from how Reflection Assistant functions. KMA and KMB are calculated to showcase how the framework supports an ITS, specifically Reflection Assistant, which aims to train learners in metacognition while acquiring cognitive abilities. Success will be measured by the framework's adaptability to various datasets and accuracy in making predictions, showcasing its potential to improve various ITS environments and generalize across diverse educational contexts.

1.3 The need for a generalized knowledge tracing framework

The pressing need for a generalized knowledge tracing framework stems from the inherent challenges posed by the diversity of input features and the evolving nature of educational data. Traditional approaches often necessitate modifying model architectures to accommodate changes in input features, turning each adaptation into a new experimental endeavor. Recent advances, particularly in KT utilizing transformers and Large Language Models (LLMs) like

BERT, have demonstrated state-of-the-art performance by treating KT data as sequential information, thereby offering a promising avenue for comprehensive knowledge assessment.

However, one critical aspect that deserves attention is the adaptability and efficiency of the knowledge tracing framework, especially concerning shifts in learners' subjects or academic years. Traditional models may struggle to adapt seamlessly to such changes, often requiring extensive retraining or fine-tuning to maintain performance levels. This is where the incorporation of BERT (Bidirectional Encoder Representations from Transformers) into the framework offers a transformative solution.

A generalized framework becomes imperative to streamline the process of knowledge tracing across diverse input feature sequences. By abstracting away from specific model architectures and focusing on the underlying principles of sequential analysis, such a framework enables adaptability to any input feature sequence, thus obviating the need for extensive model reconfigurations with each new attribute exploration. This adaptability not only enhances efficiency, but also facilitates the integration of emerging attributes into the KT process seamlessly.

Moreover, the development of a generalized framework fosters a holistic approach to knowledge tracing, allowing educators and researchers to explore new attributes and dimensions of learning without being constrained by the limitations of existing models. By providing a standardized methodology for incorporating diverse input features into the KT process, the framework empowers practitioners to conduct future experiments with greater ease and confidence, thereby advancing the field of educational technology.

In this report, we advocate for the development and adoption of a generalized knowledge tracing framework that embraces the diversity of input features and accommodates the evolving landscape of educational data. By leveraging the insights gleaned from recent experiments and advancements, we aim to catalyze efforts towards creating a unified approach to knowledge tracing that transcends disciplinary boundaries and fosters innovation in education.

1.4 Meta-cognitive inputs for knowledge tracing.

Metacognition is the practice of being aware of one's own thinking. Imagine reaching the bottom of a page and realizing, "I'm not sure what I just read". In that moment, we become aware of something we don't know, prompting us to reread or re-scan the content. This awareness of what we know or don't know is metacognition. Metacognition involves three vital aspects to learn:

1. **Planning**: Thinking ahead and strategizing. 2. **Monitoring**: Reflecting on one's learning or problem-solving processes. 3. **Evaluation**: Assessing the effectiveness of one's thinking strategies.

This study relies on the inputs from the evaluation phase of metacognition, specifically on the confidence rating a learner gives themselves Confident (C), Partially-Confident (P), Not-Confident (I)) before they get into the problem-solving phase. Metacognitive inputs for knowledge tracing represent a compelling avenue for enhancing the adaptability and effectiveness of educational models. In exploring this facet within the context of knowledge tracing frameworks, one delves into a realm where learners' self-awareness and monitoring of their own cognition play pivotal roles in shaping their learning trajectories. The integration of metacognitive inputs into the framework not only enriches the predictive capabilities but also fosters metacognitive skill development, offering multifaceted benefits to both learners and educators [4].

The choice to explore metacognition within the framework of knowledge tracing was motivated by a desire to demonstrate the flexibility and versatility of the developed framework. While it's evident that incorporating metacognitive inputs improves knowledge tracing, the prospect of adding metacognitive training on top of cognitive training may present challenges for some learners. Recognizing this, the exploration aimed to showcase how the framework seamlessly adapts to different input modalities, thereby accommodating various learning preferences and needs.

In some scenarios, where capturing explicit metacognitive inputs from learners might prove burdensome, the framework allows for an alternative approach. By leveraging correctness as an input to predict learners' confidence levels, the model circumvents the need for additional metacognitive data collection while still providing valuable insights into learners' self-assessment processes. This approach exemplifies the framework's adaptability and versatility in accommodating different input paradigms.

Conversely, in instances where explicit metacognitive inputs are readily available and desirable, the framework incorporates them seamlessly into the knowledge tracing process. By predicting correctness based on learners' confidence levels, the model not only assesses learners' knowledge states but also evaluates their metacognitive awareness. This dual approach enables educators to gain deeper insights into learners' cognitive and metacognitive processes, thereby informing targeted interventions to support their learning journey.

At the core of both approaches lies the framework's ability to seamlessly integrate diverse input modalities, whether cognitive or metacognitive, and

adapt to new kinds of inputs with ease. By leveraging the framework's flexibility, educators can assess learners' awareness (KMA) and outlook (KMB) [13], informed by both correctness and confidence predictions, to tailor interventions that address both cognitive and metacognitive aspects of learning. This holistic approach to knowledge tracing underscores the framework's potential to advance personalized learning experiences and foster metacognitive skill development in learners.

1.5 Proposed Framework and its Motivation

The framework established in this study serves as a versatile example, designed to be adaptable based on specific requirements. This framework demonstrates the capabilities of BERT and its potential to support adaptive learning environments. As previously discussed, adaptive learning models, particularly Knowledge Tracing (KT) models, operate in various ways depending on their architecture and input attributes. Therefore, to establish a benchmark and evaluate the robustness of this framework, the model architecture is fixed to use BERT, and the input attributes focus on metacognition. This choice is made to explore the relatively uncharted domain of training KT models with metacognitive inputs.

For the purposes of experimentation, the framework is designed to accept any dataset containing metacognitive inputs from learners, irrespective of other attributes. It utilizes these datasets and relevant defined targets to train models suitable for adaptive learning environments. It is important to note that the emphasis on metacognitive inputs pertains only to the experiments conducted in this study. In practice, this framework can be modified to explore various directions for training adaptive learning models.

The framework provides users with the flexibility to define hyper-parameters for each KT model they wish to produce or retrain. Additionally, users can specify which attributes from the dataset should be used for training and identify the target variable. By empowering users to tailor the framework to their specific needs, adaptability across various KT scenarios is ensured.

The Proposed Framework (as loosely illustrated in Figure 1.1)operates under the assumption that: 1. **Pre-processed and Balanced Datasets:** It is assumed that all KT datasets passed to the framework are pre-processed and balanced. This assumption streamlines the training process and ensures consistent performance across different datasets. 2. **Sequence Problem:** Regardless of the specific requirement KT model, our framework treats all KT data as a sequence problem. This approach leverages the inherent temporal

dependencies in student interactions.

At the heart of the architecture lies BERT. BERT is employed for sequence classification on the KT dataset, and in the experiments conducted, the KT task of predicting learner's correctness to a response were examined.

The framework's versatility makes it an efficient tool for Intelligent Tutoring Systems (ITS). The motivation behind having such a framework was to have a robust model training framework that could be used as a tool for an ITS to train different models for different scenarios/use-cases and something which also gives way to make quick experimentation to introduce potential new features to the ITS.

Imagine an ITS that leverages metacognitive inputs to guide learners effectively. Several scenarios arise:

- **Active Metacognitive Learning:** Learners actively engage in metacognitive processes alongside cognitive learning. In this case, the ITS receives metacognitive data about the student, including evaluation aspects and the KT model can predict correctness based on this rich information.
- **Metacognition-Optional Setup:** Alternatively, we consider scenarios where metacognitive inputs are not explicitly requested. Under certain circumstances, learners may find metacognitive training burdensome when done in a cognitive learning environment [1]. In such a case, the framework should be able to accommodate this by excluding metacognitive inputs from the input sequence. However, we also explore an experimental avenue: **Predictive Metacognition:** Can we predict learner confidence without directly asking for metacognitive inputs? Our study investigates this approach, aiming to design personalized intervention strategies.

1.6 Evaluation of Performance

The framework is designed to facilitate knowledge tracing (KT) tasks, leveraging metacognitive inputs. While this study focuses on metacognitive data, the framework is versatile and can handle various types of KT data. As a part of this study, in retrospect, there are essentially two types of models being trained, over different KT datasets.

- **Traditional KT Model (Correctness Prediction):** This model predicts the correctness of a learner's response.

- **Confidence Prediction Model:** In addition to correctness, and as an attempt to showcase the framework's usefulness in experimenting new avenues of adaptive learning, models that predict the learner's confidence in solving a problem were also trained. This aspect aligns with enabling ITS to design learner specific interventions, and other uses of such predictions could also be explored in future work.

Evaluating the models posed a challenge due to the scarcity of literature on metacognitive inputs for KT. For the traditional KT task (correctness prediction), the model is compared against existing metacognition-based models: RA-BKT and RA-ANN. These models were originally proposed in the paper titled "Knowledge Tracing for Adaptive Learning in a Metacognitive Tutor." To ensure robust comparison, we create a synthetic dataset that mimics the one used by the authors, steps for which are detailed in the paper.

Unfortunately, no existing benchmark exists for models predicting confidence levels. Therefore, an analysis of key metrics such as AUC (Area Under the Curve), ACC (Accuracy), and RMSE (Root Mean Square Error) for the confidence prediction models was done. The exploration of the practical significance and applications of confidence prediction remains an avenue for future research.

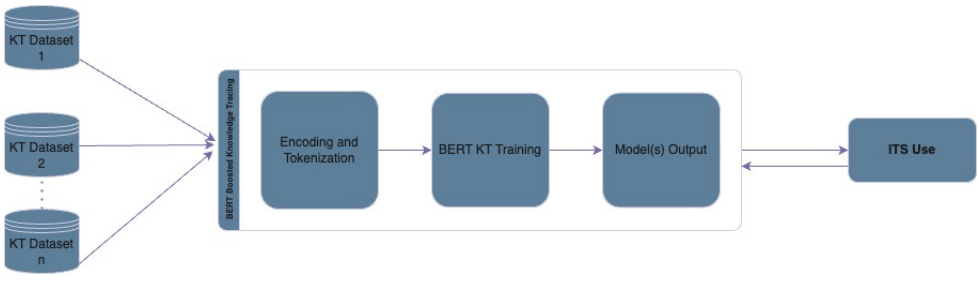

Figure 1.1: BERT Boosted KT

Chapter 2

Background

In the realm of educational technology, the quest for improving learning experiences has led to the exploration of various methodologies and tools. One notable example is the use of Intelligent Tutoring Systems (ITS), such as the Reflection Assistant (RA) [5], which emerged as a pioneering approach to integrating meta-cognitive awareness into the learning process. Developed by Gamma in 2004, the RA revolutionized the way educators evaluate learners' self-awareness and outlook by incorporating meta-cognitive inputs, particularly learners' confidence and correctness judgments.

In parallel, advancements in machine learning have paved the way for novel approaches to knowledge tracing, particularly through sequence classification techniques. By treating knowledge tracing data as sequential information, these techniques offer a more nuanced understanding of learners' knowledge acquisition processes. One such methodology involves the utilization of BERT (Bidirectional Encoder Representations from Transformers) for sequence classification, leveraging pre-trained language representations to capture rich semantic information and improve predictive performance.

2.1 Evolution of KT models

2.1.1 Bayesian KT model

Bayesian Knowledge Tracing (BKT) is a dynamic Bayesian network model that has been widely used in the field of intelligent tutoring systems. It is a probabilistic model that aims to estimate a student's knowledge state, i.e., what a student knows and doesn't know, based on their interaction history.

The BKT model is based on two fundamental assumptions:

- **Binary Knowledge State:** The student's knowledge state for a specific skill is binary, i.e., the student either knows the skill or doesn't know it.
- **No Forgetting:** Once a student has learned a skill, they do not forget it.

The BKT model uses four parameters:

- **Initial Knowledge (L0)**: The probability that the student knows the skill before interacting with the system.
- **Learning Rate (T)**: The probability that the student transitions from a state of not knowing to knowing the skill after an opportunity to practice.
- **Guessing (G)**: The probability that the student answers correctly despite not knowing the skill.
- **Slipping (S)**: The probability that the student answers incorrectly despite knowing the skill.

The BKT model updates the estimate of the student's knowledge state after each interaction, using the evidence from the student's correctness on the exercises. The updated knowledge state is then used to predict the student's performance on the next exercise. A comprehensive overview of BKT's working as HMM is demonstrated in figure 2.1.

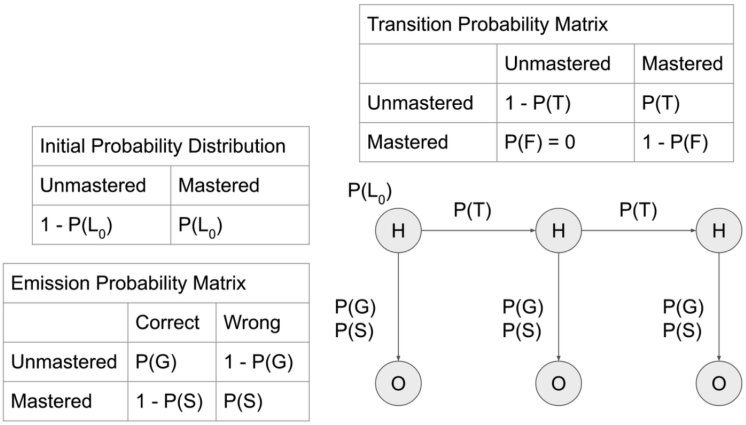

Figure 2.1: Representation of BKT as HMM [4]

2.1.2 Use of Large Language Models in KT

Large Language Models (LLMs) have been increasingly used in the field of Knowledge Tracing (KT) due to their ability to handle complex data and provide user-friendly access to information. LLMs simplify information retrieval from knowledge graphs and allow anyone to directly ask questions and get summaries instead of searching databases through traditional programming languages.

One of the key applications of LLMs in KT is in the area of sequence classification. Sequence classification is a predictive modeling problem where you have some sequence of inputs over space or time and the task is to predict a category for the sequence. What differentiates this from standard classification tasks is that the independence of observations cannot be assumed [9]. One such example work includes BiDKT: Deep Knowledge Tracing With BERT [12]. BiDKT uses the BERT model, a transformer-style bidirectional model, which has outperformed numerous Recurrent Neural Network (RNN) models on several Natural Language Processing (NLP) tasks. The model is trained under a masked correctness recovery task where the model predicts the correctness of a small percentage of randomly masked responses based on their bidirectional context in the sequences. The model architecture of BiDKT can be viewed in figure 2.2. The BiDKT's architecture is developed such that it considers two types of tokens: correctness and subject. The correctness token range from 0 to 3, where 0 serves as the padding token, 1 serves as the masked token, 2 serves as 'incorrect' token and 3 serves as the 'correct' token. Much like the pre-training of native BERT, authors of BiDKT randomly mask the correctness tokens for the model to recover and mix some sequences with only the last token being masked to simulate fine-tuning.

Experiments on several real-world knowledge tracing datasets show that BiDKT can outperform some of the state-of-the-art approaches on predicting the correctness of future student responses for some of the datasets. The detailed transition analyses by BiDKT could illuminate weaknesses in existing LLMs' knowledge mastery and guide the development of refinement.

In conclusion, the use of LLMs in KT, particularly in sequence classification, has shown promising results. The development of models like BiDKT represents a significant step forward in the field, providing more accurate and personalized learning experiences.

2.2 Use of Reflection Assistant (RA)

The Reflection Assistant (RA) represents a pioneering approach in the integration of metacognitive skills within cognitive tutoring environments, aiming to foster metacognitive development alongside cognitive learning [5]. Unlike a traditional Knowledge Tracing model which essentially serves as a prediction model, RA model is targeted to be situated within a cognitive learning context. The RA seeks to establish a connection between learners' metacognitive and cognitive performance, making it an ideal candidate for investigation in educational research.

Central to the RA's design is the hierarchical model of metacognition, which delineates various metacognitive skills essential for effective learning. These skills, including planning, selecting strategies, evaluating learning, knowledge monitoring, and control, form a hierarchical structure that guides learners in monitoring and regulating their cognitive processes [13]. Additionally, the RA draws inspiration from the conceptual stages of problem-solving, which suggest that problem-solving sessions typically involve preparation, actual problem-solving, and verification stages [7].

The RA incorporates strategically placed questions within these stages to elicit the metacognitive participation of the learners. In the preparation stage, learners are prompted to provide free-form short answers, thereby invoking Planning and Selecting Strategies metacognitive skills. Subsequently, prompts in the verification stage target Evaluating Learning and Knowledge Monitoring skills, encouraging learners to reflect on their problem-solving process and

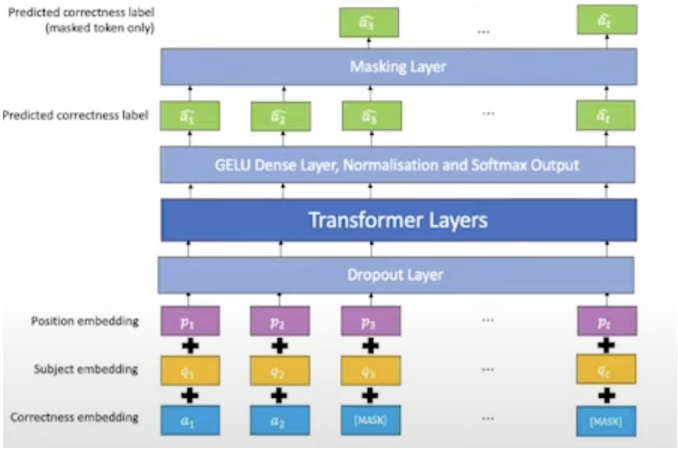

Figure 2.2: BiDKT: Model Architecture [12]

monitor their comprehension. Furthermore, the verification stage also presents learners with a Learner Profile, providing feedback on their metacognitive performance.

Figure 2.3 illustrates the user interface of a web-based tutor developed following the RA model, offering a simplified version of the RA's metacognitive prompts and learner feedback mechanisms. By embedding metacognitive prompts within the cognitive learning context, the RA aims not only to equip learners with metacognitive skills but also to enhance their cognitive learning processes. This integration of metacognition within educational environments underscores the importance of fostering self-awareness and self-regulation to

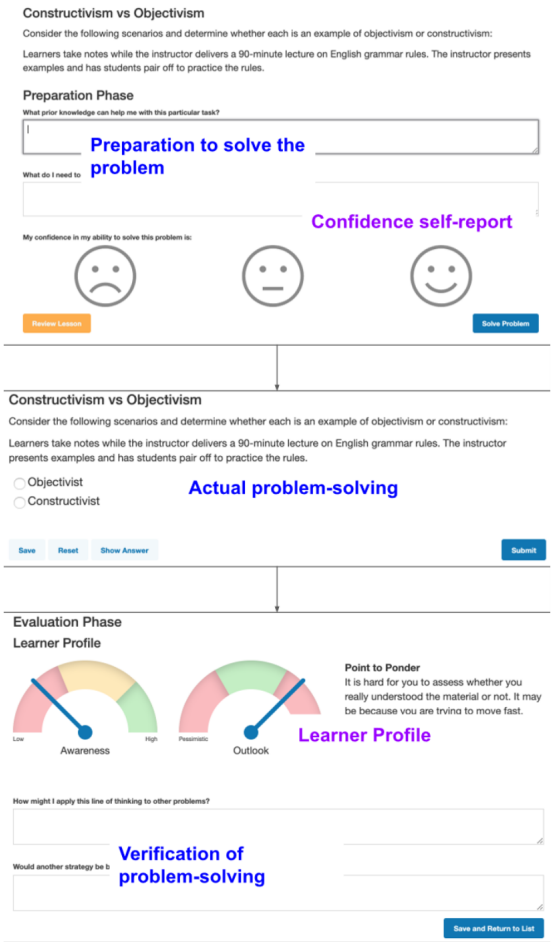

Figure 2.3: Web-based interface following RA model. [4]

promote effective learning strategies and improve overall learning outcomes.

Given that this research centers on the use of metacognitive inputs to train the required models, the Reflection Assistant (RA) setup is assumed to be an integral component of the Intelligent Tutoring System that the proposed framework will support. This setup or assumption of the setup is crucial for collecting the necessary datasets (real-world and synthetic) to train the models in subsequent stages.

2.3 Learner's Awareness and Outlook (KMA and KMB)

A crucial aspect of the background in educational technology research involves understanding learners' metacognitive skills and tendencies. One widely used metric for this purpose is the Learner Profile, which incorporates two key indicators: Knowledge Monitoring Accuracy (KMA) and Knowledge Monitoring Bias (KMB).

2.3.1 Knowledge Monitoring Accuracy (KMA)

KMA, a metric developed by Tobias and Everson [13], measures a learner's ability to monitor their own knowledge effectively. This metric is calculated based on learners' self-reported confidence levels in their ability to answer exercises during the preparation stage, compared to their actual performance. The KMA computation involves updating cumulative metrics for correct (FC), partially correct (PC), and fully incorrect (FI) predictions, according to the following matrix represented in Table 2.1:

Score	**Confidence Self-Report**		
	Confident (C)	**Partially Confident (P)**	**Not Confident (I)**
Correct	FC	PC	FI
Partially Correct	PC	FC	PC
Wrong	FI	PC	FC

Table 2.1: KMA Values Matrix

KMA is then computed using the formula in Equation 2.1:

$$KMA = \frac{FC - 0.5 \times PC - FI}{FC + PC + FI} \tag{2.1}$$

Where FC is the Fully Correct predictions, PC is the Partially Correct predictions, and FI is the Fully Incorrect predictions. Fully correct predictions receive full positive credit, while fully incorrect predictions receive full negative credit. Partially correct predictions receive negative half credit instead of positive half credit to prevent incentivizing not taking a stance. The resulting KMA has values ranging from -1 to 1. The closer the value to 1 is, the better is the learner's performance in terms of KMA. The learners are classified from the computed KMA scores, whether they have low, average, or high KMA. In

the case where only a single partially correct prediction is recorded, the KMA is -0.5, indicating low awareness.

2.3.2 Knowledge Monitoring Bias (KMB)

Knowledge Monitoring Bias (KMB) is a metric that, like Knowledge Monitoring Accuracy (KMA), is derived from learners' confidence self-reports. The computation of KMB involves updating cumulative metrics for each problem-solving opportunity. These metrics include No Bias (NB), Partial Pessimistic Bias (PPB), Full Pessimistic Bias (FPB), Partial Optimistic Bias (POB), and Full Optimistic Bias (FOB). The updating process is guided by the matrix of values shown in Table 2.2.

Score	**Confidence Self-Report**		
	Confident (C)	**Partially Confident (P)**	**Not Confident (I)**
Correct	NB	PPB	FPB
Partially Correct	POB	NB	PPB
Wrong	FOB	POB	NB

Table 2.2: KMB Values Matrix

The KMB is then computed using the formula given in Equation 2.2.

$$KMB = \frac{FOB + 0.5 * (POB - PPB) - FPB}{FOB + POB + NB + PPB + FPB} \tag{2.2}$$

The KMB, like the KMA, ranges from -1 to 1. However, unlike KMA, a value closer to 0 indicates better performance in terms of KMB. Positive values are associated with optimism, while negative scores are associated with pessimism. Therefore, values closer to 0 indicate less bias. When the KMA is not high, learners are further classified as having a pessimistic, random, or optimistic outlook. The score ranges used for this classification are detailed in Table 2.3.

The RA-ANN model [4], utilized as a benchmark in this study, employs KMA and KMB as inputs. However, a detailed examination of this training method reveals a significant limitation: these inputs are impractical in a real-world setting. The KMA and KMB attributes require both correctness and confidence inputs from the learner. For a model tasked with predicting the correctness of a learner's response, these attributes would not be available prior to making the prediction. This is because correctness, which the model aims to predict, is a necessary component for generating the KMA and KMB inputs.

Therefore, using KMA and KMB as inputs for model training is not feasible since they depend on information that is not available before prediction.

In this study the KMA and KMB is used to showcase the utility of models which is trained on metacognitive and other relevant input attributes, excluding KMA and KMB themselves, within an Intelligent Tutoring System (ITS) setup. The models trained as a part of this study using the BERT-Boosted KT framework are of mainly two types one that predicts learner's response's correctness and other that predicts the learner's confidence. In either of these cases, both correctness and confidence are going to be present once the required prediction is made, allowing for the evaluation of KMA and KMB for the given problem the learner was attempting to solve. The projection of KMA and KMB could be particularly useful in an ITS setup, as it may help identify intervention strategies for students exhibiting shallow learning. Future research can explore the interventions that can be devised with the use of KMA and KMB. For example, if a learner's evaluated KMA (awareness) and KMB (outlook) are low, the instructor could be alerted to this shallow learning experience. Alternatively, if the ITS allows, automated interventions using Large Language Models could be implemented.

Score Range	**Classification**	
	KMA	**KMB**
$[-1, -0.25)$	Low	Pessimistic
$[-0.25, 0.25)$	Average	Random
$[0.25, 0.5)$	Average	Optimistic
$[0.5, 1]$	High	Optimistic

Table 2.3: Classification Based on KMA and KMB values

2.4 Sequence classification for knowledge tracing

Knowledge Tracing (KT) is a fundamental problem in the field of intelligent tutoring systems, which involves modeling the knowledge state of a learner over time. Traditional KT models, such as Bayesian Knowledge Tracing (BKT), treat this problem as a binary classification task, where the goal is to predict whether a learner will answer a question correctly or not based on their past performance.

However, recent advances in the field have led to a paradigm shift, with researchers now treating KT as a sequence classification problem. This approach views a learner's interaction with an educational system as a sequence

of events, each event representing the learner's interaction with a particular educational item (e.g., a question or a learning resource). The goal is then to classify each event in the sequence (i.e., predict the learner's performance) based on the history of past events.

This sequence classification approach to KT has several advantages over traditional methods:

- **Modeling temporal dependencies:** Sequence classification models can capture the temporal dependencies between different events in a learner's interaction sequence. This is crucial for KT, as a learner's performance on a particular item is often dependent on their past interactions.
- **Handling variable-length sequences:** Unlike traditional KT models, which often require a fixed-length input, sequence classification models can handle variable-length sequences. This makes them more flexible and better suited to real-world educational data, which often consists of interaction sequences of varying lengths.
- **Leveraging powerful sequence models:** Treating KT as a sequence classification problem allows us to leverage powerful sequence models developed in the field of machine learning, such as Recurrent Neural Networks (RNNs) and Transformer models. These models have shown state-of-the-art performance on a variety of sequence classification tasks and are well-suited to the KT problem.

One of the most promising sequence classification models for KT is the Transformer model, which uses a mechanism called self-attention to capture dependencies between events in a sequence. The Transformer model has been used to develop several state-of-the-art KT models, such as the Deep Knowledge Tracing with Transformers (DKTT) [10] model.

The DKTT model uses a Transformer to encode a learner's interaction sequence into a set of continuous representations, which capture both the learner's current knowledge state and the temporal dependencies between different interactions. These representations are then used to predict the learner's performance on future interactions.

The use of self-attention allows the DKTT model to focus on the most relevant past interactions when making a prediction, which can lead to more accurate and interpretable KT models. Furthermore, the DKTT model can be trained end-to-end using standard backpropagation, making it easy to optimize and scale to large datasets.

In conclusion, treating KT as a sequence classification problem is a promising new direction in the field, which leverages powerful sequence models and provides a more flexible and accurate approach to modeling learner knowledge.

This study will explore and examine the use of sequence classification and its benefits for training the required models.

2.5 BERT for Sequence Classification in Knowledge Tracing

Bidirectional Encoder Representations from Transformers (BERT), a powerful machine learning technique for natural language processing (NLP) pre-training, has garnered significant attention in recent years. Its ability to capture intricate context dependencies within text data has made it a valuable asset across various NLP tasks. One such area where BERT is being explored is Knowledge Tracing (KT).

Traditionally, KT models have relied on diverse approaches to tackle the challenge of assessing learners' skill mastery. However, recent work, such as BiDKT, has specifically delved into BERT's sequence classification capabilities for KT. BiDKT leverages relevant attributes (rather than metacognition-based inputs) to trace learners' skill acquisition. Notably, BiDKT trains under a masked correctness recovery task, where the model predicts the correctness of a small percentage of randomly masked responses based on their bidirectional context within the sequences. The BiDKT architecture is inspired by BERT's pre-training strategy as described in the paper "Contrastive Bidirectional Transformer for Temporal Representation Learning" [11], and it does not require a fine-tuning step on downstream data. In contrast, this study uses the existing BERT architecture and fine-tunes it on downstream data (KT dataset). The results produced by BiDKT, which closely rival state-of-the-art KT models, support the idea that using a BERT architecture and treating Knowledge Tracing data as a sequence problem is a promising approach for KT tasks.

With the evolution of more managed implementations of BERT, it becomes even more efficient to focus on dealing with KT problems as sequence tasks and abstract away the implementation details of BERT. This approach allows for quicker experimentation and framework development. Unlike traditional KT models, which often require heavy experimentation to determine the optimal input data format, leveraging BERT's inherent capabilities for solving sequence-based tasks streamlines the process.

The 'BERTForSequenceClassification' class from the Hugging Face library

is particularly useful for this task. This class is a variant of the standard BERT model with an added linear layer for classification. The model takes a sequence of learner interactions as input and outputs a probability distribution over the possible outcomes (e.g., correct or incorrect) for the next interaction.

Here is a high-level overview of how to use 'BERTForSequenceClassification' for Knowledge Tracing:

1. **Preprocessing:** The learner interaction sequences are preprocessed into a format suitable for BERT. Each interaction is tokenized, and special tokens ('[CLS]' and '[SEP]') are added at the beginning and end of each sequence, respectively.

2. **Model Training:** The 'BERTForSequenceClassification' model is trained on the preprocessed data. The model learns to predict the outcome of the next interaction based on the history of past interactions.

3. **Prediction:** Once the model is trained, it can be used to predict the outcome of future interactions. Given a sequence of past interactions, the model outputs a probability distribution over the possible outcomes for the next interaction.

The use of BERT for Knowledge Tracing offers several advantages. First, BERT's ability to capture complex context dependencies makes it well-suited to modeling the temporal dependencies in learner interaction data. Second, the 'BERTForSequenceClassification' class provides a flexible and powerful tool for sequence classification, which can be easily adapted to the Knowledge Tracing task.

However, it's important to note that while BERT represents a promising approach to Knowledge Tracing, it also comes with its own set of challenges. For instance, BERT models can be computationally intensive to train and require large amounts of data to perform well. Therefore, careful consideration must be given to these factors when deciding to use BERT for Knowledge Tracing.

2.6 Introduction to bert-base-uncased

The `bert-base-uncased` model is a widely utilized variant of the Bidirectional Encoder Representations from Transformers (BERT) architecture, developed by Google. The `bert-base-uncased` model is specifically designed for English language text, employing a lowercased vocabulary, which means that all the text is converted to lowercase before tokenization.

2.6.1 Architecture

The `bert-base-uncased` model comprises 12 layers (transformer blocks), 768 hidden units, and 12 attention heads, amounting to a total of 110 million parameters. This configuration enables the model to capture complex linguistic patterns and relationships within text. Each transformer block consists of multi-head self-attention mechanisms and feed-forward neural networks, allowing the model to attend to different parts of a sentence simultaneously and capture diverse contextual information.

2.6.2 Pre-training and Fine-tuning

BERT's power lies in its two-stage training process: pre-training and fine-tuning. During pre-training, the model is exposed to a large corpus of text, such as Wikipedia and BooksCorpus, and learns to predict missing words in a sentence (Masked Language Modeling) and the next sentence in a pair (Next Sentence Prediction). This extensive training equips BERT with a robust understanding of language nuances and contexts.

Fine-tuning involves adapting the pre-trained BERT model to specific downstream tasks by training it on task-specific datasets. For `bert-base-uncased`, this means leveraging the pre-trained knowledge and adjusting the model weights according to the target task, which, in this study, is sequence classification for Knowledge Tracing (KT) datasets.

2.7 Tokenization and Encoding

2.7.1 Tokenization

Tokenizers perform two essential functions:

1. Splitting the raw text into smaller units—referred to as tokens—at the word, subword, or character level (Figure 2.4).

2. Converting these tokens into a numerical format that a machine learning model can interpret.

The BERT tokenizer employs a subword tokenization algorithm, which divides the text into subword units. This approach strikes an optimal balance between vocabulary size and sequence length, and it effectively handles rare and out-of-vocabulary words, thereby minimizing the necessity of treating them as unknown tokens.

The specific subword tokenization algorithm used in the BERT tokenizer is called WordPiece. WordPiece is designed to construct a vocabulary of tokens of a predetermined size by iteratively selecting and merging the highest scoring character pairs within the text dataset.

The scoring mechanism for selecting these pairs is based on the ratio of the frequency of the pair to the product of the frequencies of the individual characters, formulated as follows:

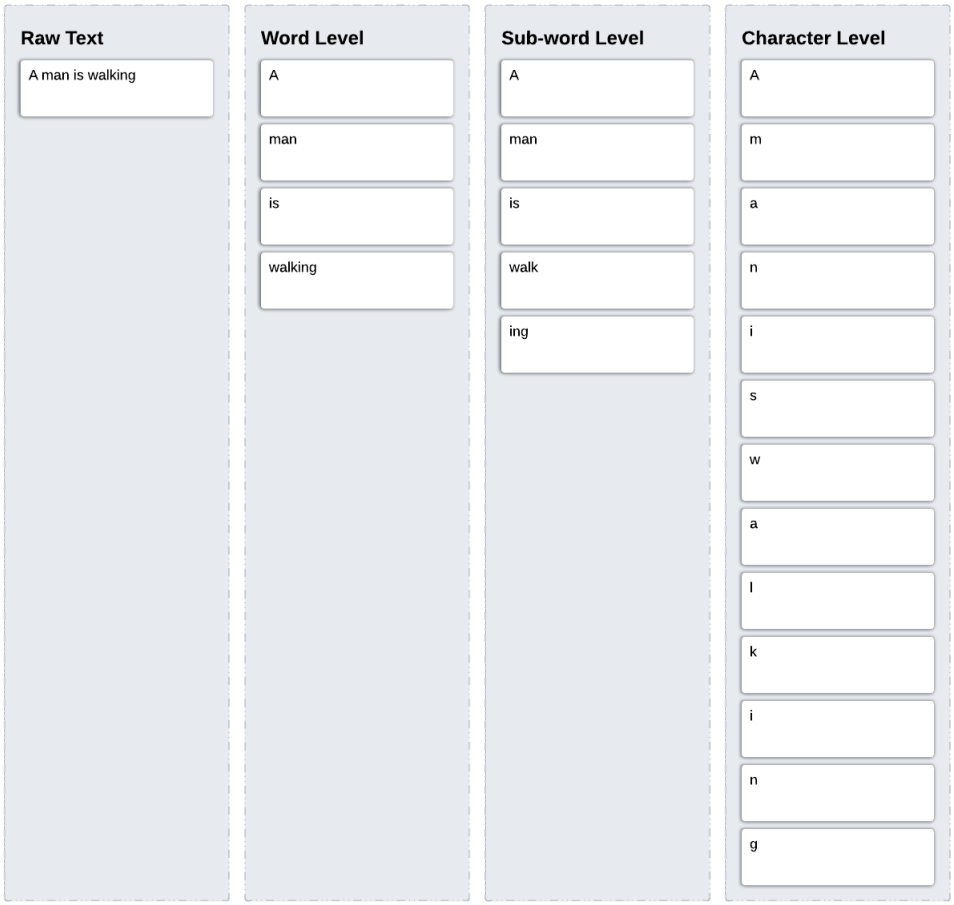

Figure 2.4: Tokenization

$$\text{score} = \frac{\text{freq\_of\_pair}}{\text{freq\_of\_first\_element} \times \text{freq\_of\_second\_element}} \tag{2.3}$$

This method ensures that the most frequently co-occurring character pairs are merged first, thereby optimizing the tokenization process for efficiency and effectiveness.

The figure 2.5 illustrates a batch of two raw text samples being processed by a tokenizer. The tokenizer generates the processed text (string tokens) along with three distinct integer encodings.

The processed text includes three special tokens:

- **Classification token:** The `[CLS]` token is positioned at the very start of a sequence. It can serve as a summary representation for the entire sequence.
- **Separation token:** The `[SEP]` token is placed at the end of each sentence within a sequence. This helps BERT to distinguish between sentences and recognize sentence boundaries.
- **Padding token:** The `[PAD]` token is appended at the end of the sequence — as many as necessary — to extend the sequence length so that all sequences in a batch have uniform length. This is required for matrix operations to process multiple samples (a batch) in parallel.

The three types of integer encodings derived from the processed text are as follows:

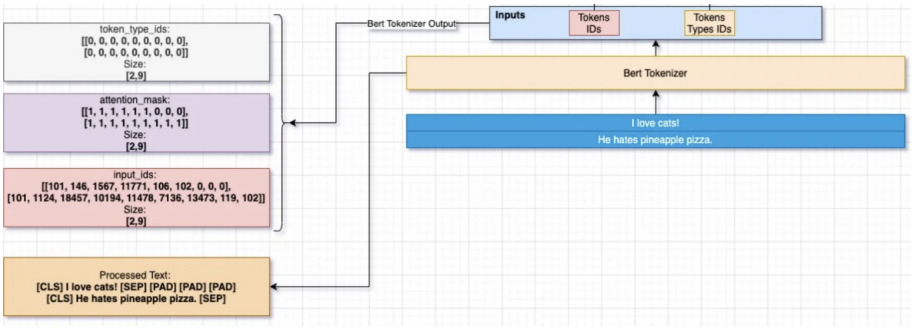

Figure 2.5: Diagram of BERT Tokenizer Inputs and Outputs. Created by George Mihaila.

- **Input IDs:** The BERT base model has a vocabulary of 30,522 tokens. Each unique string token maps to a unique integer ID. These IDs represent each token in a sequence.

- **Attention Mask:** Sequences shorter than the maximum defined length are extended with padding tokens. These padding tokens are placeholders and should not influence the model's output. The attention mask differentiates padding tokens from other tokens. Non-padding tokens have a value of one, while padding tokens have a value of zero.

- **Token Type IDs:** In the task of next-sentence prediction during BERT pretraining, sequences are composed of two sentences. The second sentence is either the immediate successor of the first sentence from the source text or a random sentence from the source text. The goal is to predict whether the second sentence is the actual next sentence or a random one. Token type IDs label tokens in the sequence as belonging to the first or second sentence by assigning zeros and ones, respectively. In sequence classification, however, all tokens are treated equally and encoded with zeros. Thus, token type IDs do not affect the model's output in sequence classification.

2.7.2 Embedding

Embeddings are vector representations of tokens that enable their spatial organization. The dimensionality of the embedding vector is a design decision, determining the space in which the embeddings are arranged. A larger embedding space provides more capacity for token organization.

In practice, embedding dimensions are typically set high to handle extensive vocabularies and the complexities of contextual semantics. For instance, the BERT base model employs an embedding dimension of 768.

Each token is assigned three types of embeddings, which are subsequently summed:

- **Word Embeddings:** Each input ID is mapped to its corresponding row vector in the word embedding lookup table, which contains one row for each unique token in the vocabulary. BERT base has a vocabulary of 30,522 tokens.

- **Token Type Embeddings:** Similar to word embeddings, this lookup table consists of only two rows, corresponding to the sentence (first or second) in which the token appears. This is pertinent primarily during

the pretraining task of next-sentence prediction and does not affect the model output in other tasks.

- **Positional Embeddings:** These embeddings provide the model with a sense of the order of tokens within a sequence. The positional IDs are sequential integers from 0 to n, where n is the maximum sequence length. For BERT base, the maximum sequence length is 512.

Summing these three types of embeddings encapsulates various facets of information into a single embedding. This combined embedding reflects the semantic meaning of the tokens, their positions within the sequence, and the sentence context (relevant only for next-sentence prediction).

These embeddings are learned during the pretraining phase on large text corpora and can be fine-tuned for specific downstream tasks.

2.7.3 Encoder

Unlike traditional sequential models, BERT employs a transformer-based architecture, enabling parallel processing of input tokens and facilitating more efficient learning of contextual representations.

At its core, the BERT encoder comprises multiple layers of self-attention mechanisms and feed-forward neural networks. These layers iteratively refine token representations by considering both the left and right context of each token in the input sequence. Through self-attention, BERT can dynamically weigh the importance of each token's context, enabling the model to capture intricate semantic relationships and contextual nuances within the text.

During fine-tuning, the BERT encoder can be adapted to various downstream tasks, such as text classification, named entity recognition, and question answering, by adding task-specific output layers while keeping the encoder weights fixed or partially updating them. This adaptability, coupled with its superior performance on diverse NLP tasks, has made the BERT encoder a cornerstone in modern natural language understanding systems.

2.8 Leveraging BertForSequenceClassification

BERTForSequenceClassification is a model variant of the BERT architecture tailored for sequence classification and regression tasks. It incorporates a linear layer atop the pooled output, facilitating tasks such as those in the General Language Understanding Evaluation (GLUE) benchmark suite.

This class inherits from the PreTrainedModel class in HuggingFace, inheriting various generic methods for model manipulation. Referencing the superclass documentation provides insights into methods like model downloading, saving, embedding resizing, and head pruning.

Additionally, BERTForSequenceClassification is a PyTorch torch.nn.Module subclass, allowing its utilization as a conventional PyTorch Module.

The forward method of this class accepts various parameters:

- **input_ids:** Indices representing input sequence tokens in the vocabulary.
- **attention_mask:** Mask to avoid attention on padding token indices.
- **token_type_ids:** Segment token indices distinguishing between sentence A and sentence B tokens.
- **position_ids:** Indices of input sequence token positions in position embeddings.
- **head_mask:** Mask to nullify selected heads of the self-attention modules.
- **inputs_embeds:** Optionally, embedded representations instead of input_ids for more control over vector association.
- **output_attentions:** Whether to return attention tensors of all attention layers.
- **output_hidden_states:** Whether to return hidden states of all layers.
- **return_dict:** Whether to return a ModelOutput instead of a tuple.
- **labels:** Labels for computing the sequence classification/regression loss.

The output of the forward method returns either a SequenceClassifierOutput object or a tuple of torch.FloatTensor.

Chapter 3

Design Goal and Mechanisms

The primary design goal of this study is to develop a unified Knowledge Tracing (KT) model training framework. This framework aims to demonstrate a proof of concept capable of handling any balanced KT dataset with a defined target, and generating a KT model, or potentially another type of model beneficial to an Intelligent Tutoring System (ITS). Examples of such alternative models will be discussed in subsequent chapters. The efforts to develop a unified framework stems from the need to have flexibility when dealing with disparate KT datasets and to also have adaptability to serve various educational context.

It is important to emphasize that the framework developed in this study serves as a proof of concept and can be restructured as needed. For instance, unlike the framework created here, one could design a framework that does not rely on the KT dataset being balanced, with the pipeline itself handling the balancing.

To achieve the framework's objective, BERT (Bidirectional Encoder Representations from Transformers) is employed for sequence classification on the given dataset.

More specifically, a model capable of performing sequence classification on a KT dataset is trained through the following steps:

1. The `bert-base-uncased` model from the BERT family is used for this task.

2. The input data is tokenized, embedded and encoded using the tokenizer of the `bert-base-uncased` model.

3. The `BertForSequenceClassification` class from the Hugging Face library is utilized. This class works with BERT models and adds a linear

layer on top of the specified BERT model to perform classification tasks.

4. The model training (fine-tuning) process is initiated.
5. The trained model is saved for future use.

The intention is to ensure that these steps are consistently followed, regardless of the specific KT dataset used, provided it is balanced and the target is defined.

3.1 Tokenization and encoding application

The tokenization and encoding processes outlined in chapter 2 will be applied in the context of the KT dataset. In this study, the target variable, representing either the correctness of a learner's response or the learner's confidence level before answering a question, will be treated as the subsequent token in the sequence to be predicted. All other dataset attributes, such as student ID, skill, and question number, are condensed into a single column within the dataset. These attributes are represented as a unified string, separated by spaces (e.g., student_id skill question_number, etc.), which ensures that regardless of the input attribute sequence of the KT dataset, the preprocessing of the input data is done through an unified approach. During both training and inference stages, this single column containing all attribute data in string format serves as the input sequence to the BERT model, which is then tokenized, embedded and ecncoded using BERT's tokenizer and encoder. The model is then tasked with predicting the subsequent token in the sequence, which corresponds to the defined target variable in this case.

3.2 BertForSequenceClassification application

As a part of this study, the BERTForSequenceClassification class is used with its forward method accepting three parameters, the input_ids, the attention_mask and the labels, and a SequenceClassifierOutput object is received as an output.

3.3 Model Training

During the training phase, several steps are involved to optimize the model's parameters and improve its performance. The process includes the use of an optimizer, a custom dataset loader, and the model's input data (figure 3.3).

3.3.1 Optimizer Usage

An optimizer, in this case, AdamW, is utilized to update the model's parameters based on the computed gradients during backpropagation. The learning rate (lr) is set to 1×10^{-5} to control the size of parameter updates.

3.3.2 Data Loader

A custom dataset, represented by the LearnerDataset class, is defined to organize and preprocess the input data for training. This dataset utilizes encodings to facilitate the retrieval of input_ids, attention_mask, and labels. DataLoader is then employed to batch and shuffle the dataset, enabling efficient data loading during training iterations.

3.3.3 Model Input

The model takes three inputs during each training iteration:

- **input_ids**: Representing the indices of input sequence tokens in the vocabulary.
- **attention_mask**: Indicating which tokens should be attended to and which should be ignored.
- **labels**: Providing the ground truth values for the model's predictions, facilitating the computation of the loss function.

3.3.4 Training Loop

The model is set to training mode using the `model.train()` method to enable gradient computation and parameter updates. The training process spans multiple epochs, with each epoch iterating through batches of data loaded by the DataLoader. Within each iteration, the optimizer's gradients are reset using `optimizer.zero_grad()`, followed by the forward pass through the model to compute predictions and the corresponding loss. The backward pass computes gradients of the loss with respect to the model's parameters, facilitating parameter updates via the optimizer's `step()` method. This process iterates until convergence or a predefined number of epochs.

```
# Initialize optimizer
optimizer = AdamW(model.parameters(), lr=1e-5)

# Train the model using training_data_loader
num_epochs = 100
model.train()
for epoch in range(num_epochs):
    for batch in training_data_loader:
        optimizer.zero_grad()
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['labels'].to(device)
        outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
```

Figure 3.1: Model training

Chapter 4

Methodology

In the pursuit of developing the BERT Boosted KT Framework, a generalized knowledge tracing framework, the utilization of advanced machine learning techniques, particularly leveraging pre-trained language models (LLMs), becomes imperative. To this end, the framework capitalizes on the capabilities offered by BertForSequenceClassification from the Hugging Face Transformers library.

4.1 Data Curation and Preprocessing

The Data Curation and Pre-processing phase is pivotal to the experiments conducted as it provides variety of datasets that vary from each other in terms of setup, amount of data and the overall distribution of the data, making it efficient to test the domain-adaptability and flexibility of the framework developed. In the attempt to do so, three separate datasets were created/prepared, each of which are discussed in detail below. The datasets created in the process are ensured to contain metacognitive inputs (Confidence ratings of learners) which is not typical in open source KT datasets like: ASSISTments and Geometry Angles.

This data curation and pre-processing stage does not serve as a part of the framework itself, as the framework assumes that the input data is pre-processed and balanced, but to make the framework more comprehensive, a future attempt could be made to integrate the said phase in the framework.

4.1.1 Synthetic dataset preparation

To address the challenge of obtaining educational data for knowledge tracing experiments, a synthetic dataset was curated as part of the methodology. Drawing inspiration from previous work on knowledge tracing for adaptive learning in a metacognitive tutor [4], the dataset creation process is aimed at simulating various learner behaviors and performance scenarios.

The dataset was crafted by defining learner personas, with possible behaviors classified based on both performance and confidence reports. These behaviors were combined, resulting in an initial data set that contained information for 104 (8 performance behaviors and 13 confidence behaviors) unique learners. Each synthesized learner was provided with ten opportunities to answer quiz items or demonstrate learning components, aligning with the assumption of Bayesian Knowledge Tracing (BKT) that learners accumulate knowledge over repeated attempts.

To ensure the robustness of the dataset and prevent degeneracy of the model, several conditions were introduced. For example, performance behaviors where learners progressively improved were simulated, with data generated to reflect instances where learners initially answered incorrectly but consistently improved over subsequent attempts. Similar conditions were applied to confidence reports, allowing for partially correct predictions and reflecting varying degrees of learner confidence.

The dataset was further enriched by combining performance and confidence behaviors, resulting in a total of 391 learner data points.

In total, the synthetic dataset comprised 3910 (391 learners with 10 opportunities each) observation records, each capturing the nuances of learner behavior and performance across multiple knowledge tracing opportunities.

Table 4.2 breaks down the assumed learner's behaviours considered for the dataset curation.

Calculating KMA and KMB

After the synthetic dataset is created, each synthesized learner's Knowledge Monitoring Accuracy (KMA) and Knowledge Monitoring Bias (KMB) are calculated based on their performance (correctness) and confidence reports. This step is crucial as it allows us to assess the similarity of the synthetic dataset to real-world knowledge tracing datasets, providing insights into the efficacy of the explored approach.

The formulas mentioned in equations 2.1 and 2.2 are utilized for computing KMA and KMB, respectively. These formulas take into account the

Description	Training	Validation
Data count	3120 (79.79%)	790 (20.20%)
Answer		
- 0	1553 (49.77%)	366 (46.32%)
- 1	1567 (50.22%)	424 (53.67%)
Confidence		
- C	1391 (44.58%)	279 (35.31%)
- P	770 (24.67%)	275 (34.81%)
- I	959 (30.73%)	246 (31.13%)

Table 4.1: Balanced Synthetic Dataset Split

learner's self-reported confidence levels and actual performance on the knowledge tracing tasks. By quantifying the learner's accuracy in monitoring their own knowledge acquisition process (KMA) and their bias towards optimistic or pessimistic predictions (KMB), we gain valuable insights into the dataset's fidelity and its alignment with established knowledge tracing methodologies.

Evaluation of Dataset Realism

To assess the realism of the synthetic dataset and its alignment with real-world educational data, trends between opportunities and correctness, awareness, and outlook were analyzed using generalized linear modeling techniques. This approach allowed to visualize how closely the synthetic data mirrored patterns observed in authentic educational contexts.

To ensure comparability, awareness and outlook values were adjusted to a standardized range of [0, 1], mirroring the scale used for correctness. Awareness values were normalized to the range [0, 1], while the outlook values were corrected to reflect their distance from the optimal value of 1. The resulting trends were then compared with student data obtained from ASSISTments (2015) and the Geometry Angles dataset, both widely recognized sources in the field of learning research.

The ASSISTments platform, known for its extensive use in educational research, provided valuable insights into student performance and behavior [8]. Similarly, the Geometry Angles dataset, stemming from rigorous research on metacognition, offered additional perspectives on learner cognition [2].

To facilitate a meaningful comparison, an "Opportunity" attribute was introduced to both datasets, ensuring consistency in tracking learning opportunities across student-problem combinations. Only student-problem combinations with exactly ten opportunities were included for analysis, maintaining

uniformity across datasets.

Upon analysis, both the ASSISTments and Geometry Angles datasets exhibited trends consistent with the synthetic dataset, validating the realism and fidelity of the synthetic data generation process. The observed upward trends in performance, awareness, and outlook mirrored patterns observed in real-world educational settings, underscoring the efficacy of the synthetic dataset in capturing authentic learner behaviors.

Balancing synthetic dataset

To ensure model training and validation integrity, the dataset was randomly divided into five groups, with four groups designated for model training and the remaining group reserved for validation purposes. This division ensured relatively even distributions across key factors, mitigating concerns related to unbalanced data commonly encountered in machine learning research.

Table 4.1 provides a detailed breakdown of the training and validation set division, illustrating that the distributions of key attributes are fairly balanced.

4.1.2 Real-world data collection

In addition to synthesizing datasets, real-world data collection was conducted to further validate the proposed framework and evaluate its applicability in authentic learning environments. A mathematics quiz was administered to a group of 11 students from the Rochester Institute of Technology (RIT), each tasked with answering 50 questions encompassing five skill components of mathematics, with ten questions dedicated to each skill component. This structure ensured a comprehensive assessment of learners' proficiency across various mathematical concepts, resulting in a total of 550 unique learner data points.

To maintain anonymity and confidentiality, each student's identity was kept anonymous throughout the data collection process. The quiz was presented in a Reflection Assistant (RA) model format, where students first evaluated their confidence level in solving the given problem, by visually seeing the problem, and rating their confidence between, Confident (C), Partially Confident (PC) or Not Confident (NC). Students provided self-assessments of their confidence levels before proceeding to attempt the problem, enabling the collection of valuable data on learners' confidence levels alongside their performance outcomes.

The quiz was hosted on the MyCourses platform provided by RIT, ensuring seamless access and participation for the enrolled students. Following

data collection, the curated dataset underwent a balancing process focused primarily on correctness, representing the learner's performance. To address class imbalance, oversampling and undersampling techniques were employed. Figures 4.1 and 4.2 illustrate the test setting presented to learners.

Balancing real-world dataset

The dataset curated after the conducted test, provided a multitude of attributes which could have been used for the study, but instead, only the most relevant attributes serving to create a KT dataset were used, namely: stu-

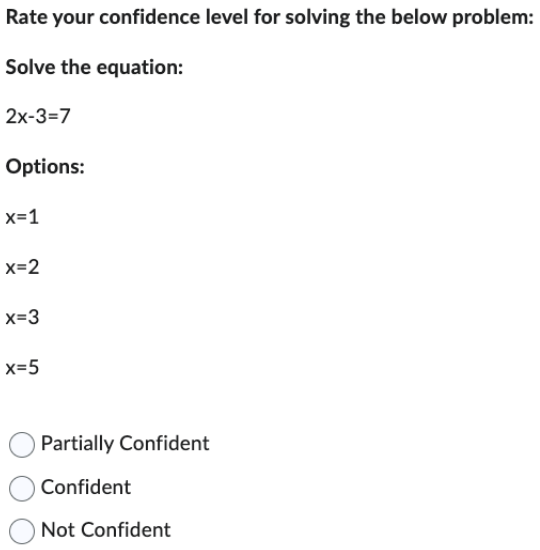

Figure 4.1: Confidence self-report

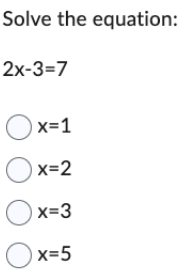

Figure 4.2: Performance report

dent_id, knowledgecomponent_id, question_id, Correctness and Confidence.

Using oversampling, the dataset was balanced to have an equal count of correctness values, resulting in 340 instances each for correctness values of 0 and 1. Conversely, undersampling was applied to balance the dataset, resulting in 159 instances each for correctness values of 0 and 1.

While it is possible to balance the dataset around both correctness and confidence to prepare for various prediction tasks, the limited size of the dataset necessitated a strategic decision to focus on balancing around a single target variable. This approach ensured the dataset's readiness for predicting correctness while maximizing the available data resources effectively.

Hence, the same dataset was balanced again, but this time around Confidence, using both under and over sampling techniques. The original count of Confident (C), Partially Confident (PC) and Not Confident (NC), classes in Confidence target attribute were:

- C: 291
- PC: 105
- NC: 103

After applying the oversampling strategy, each class was evenly distributed, with 291 records per class, aligning with the highest class count. Conversely, after utilizing the undersampling technique, each class contained 103 records.

With the oversampling and undersampling techniques being utilized to address class imbalance, the dataset remained representative of real-world learning scenarios, providing valuable insights into learners' performance and confidence levels in solving mathematical problems.

4.1.3 Open-source KT datasets

In this study, the real-world dataset utilized is the Database Exercises for Knowledge Tracing (DBE-KT22) dataset. This dataset was introduced in a paper titled "DBE-KT22: A Knowledge Tracing Dataset Based on Online Student Evaluation" [6]. The paper highlights the increasing importance of online education in providing affordable high-quality education to students worldwide, particularly accentuated during the global pandemic as more students transitioned to online learning environments.

The DBE-KT22 dataset was collected from an online student exercise system used in a course taught at the Australian National University in Australia. It was specifically curated to address the Knowledge Tracing problem, which

involves tracking students' knowledge progress over time. This dataset serves as a valuable resource for tasks such as course recommendation, exercise recommendation, and automated evaluation in the domain of online education.

The paper discussing the DBE-KT22 dataset provides insights into its characteristics and contrasts it with existing datasets in the knowledge tracing literature. Notably, the dataset is made publicly accessible through the Australian Data Archive platform, facilitating its use and exploration by researchers and educators alike.

Utilizing the DBE-KT22 dataset in this study adds additional real-world dimension to the research, enabling the evaluation and validation of the proposed framework in a practical educational setting. By leveraging this rich source of data, the study aims to contribute to the advancement of knowledge tracing techniques and their application in online education contexts.

Balancing the Open-Source KT Dataset

Following the processing of the DBE-K22 dataset to resemble a KT dataset (as discussed in the next chapter), it was observed that the DBE-K22 dataset contained a total of 305,794 records. Of these, 70% were allocated for training and the remaining 30% were reserved for validation. The open-source dataset was balanced in a manner similar to the real-world dataset, focusing on both Correctness and Confidence, resulting in two balanced datasets.

Initially, the distribution of correctness in the DBE-K22 dataset was:

- 0: 71,021
- 1: 234,773

The initial distribution of confidence in the dataset was:

- C: 236,258
- PC: 46,933
- NC: 22,603

4.2 Model Choice and Framework

In this study, the choice of model for knowledge tracing is critical, influencing the performance and adaptability of the framework. The selected model not only needs to exhibit robust performance but also accommodate the dynamic

nature of educational data effectively. The framework leverages BertForSequenceClassification, a state-of-the-art model from Hugging Face Transformers library, for its powerful capabilities in sequence classification tasks.

4.2.1 Model Size and Hyper-parameters

As discussed earlier, the BertForSequenceClassification model utilized in this study is initialized with the 'bert-base-uncased' configuration, which represents a pre-trained BERT model with 12 transformer layers, 768 hidden dimensions, and 12 attention heads (illustrated in Figure 4.3). This configuration strikes a balance between model complexity and computational efficiency, making it suitable for the knowledge tracing task.

Hyper-parameters, including the learning rate, batch size, and number of training epochs, are carefully chosen to optimize the model's performance. For instance, the AdamW optimizer with a learning rate of 1e-5 is employed for efficient gradient descent optimization. AdamW incorporates weight decay regularization, which helps prevent overfitting by penalizing large parameter values. The choice of a batch size of 32 balances computational efficiency with model convergence during training.

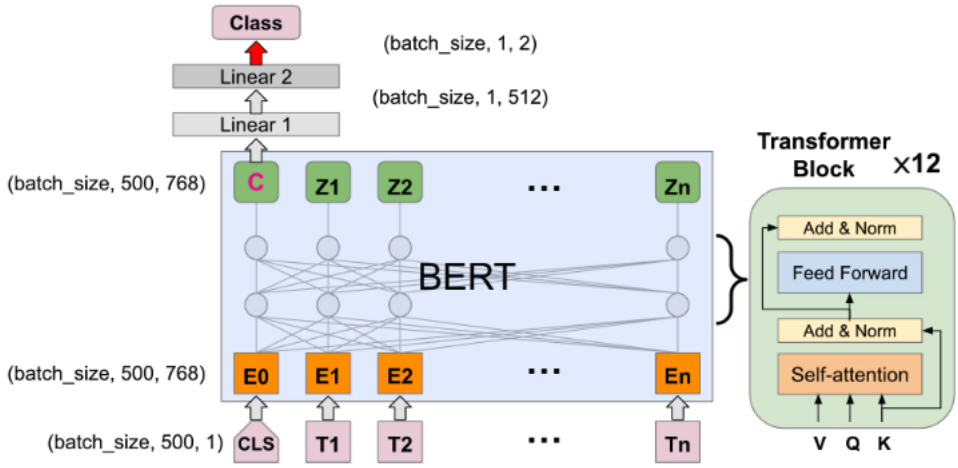

Figure 4.3: BERT's architectural breakdown. [14]

4.2.2 General Framework

The framework encompasses a comprehensive pipeline for data preprocessing, model training, and evaluation. The preprocessing step involves encoding the input data using the BertTokenizer, which tokenizes the textual input and converts it into numerical representations compatible with the BERT model. Additionally, a custom dataset class, LearnerDataset, is defined to facilitate efficient data loading and batching during training.

The data preprocessing phase is adaptable to accommodate varying input attributes dynamically. The preprocess_data function combines different input columns into a single text string and tokenizes it using the tokenizer. This dynamic handling of input attributes ensures flexibility in incorporating new kinds of input features into the knowledge tracing framework seamlessly.

During model training, the encoded data is fed into the BertForSequenceClassification model through DataLoader objects, which enable efficient batch processing. The model is trained using the AdamW optimizer, which updates the model parameters based on the computed gradients. The training process iterates over multiple epochs, allowing the model to learn the underlying patterns in the data and improve its performance over time.

The equation 4.1 encapsulates the model training process:

$$\begin{aligned}[M, T \xrightarrow{\text{Initialize}} D_{\text{enc}} &= \text{Preprocess}(D, L) \\ &\xrightarrow{\text{Preprocess}} \text{DataLoader}(D_{\text{enc}}) \\ &\xrightarrow{\text{Load Data}} O(M.\text{parameters}) \\ &\xrightarrow{\text{Optimize}} \text{for } e \text{ in range}(E) : M.\text{train}(D_{\text{enc}}, O) \\ &\xrightarrow{\text{Train}} \text{Model Trained}]\end{aligned} \tag{4.1}$$

In equation 4.1: M and T represent the BERT model and tokenizer respectively.
D represents the data and L represents the labels column.
D_enc represents the preprocessed data.
E represents the number of epochs.
O represents the optimizer.

4.3 Prediction Process

As previously mentioned, a key aspect of the framework's architecture that ensures insensitivity to input data lies in the preprocessing step (Figure 4.6).

This step involves consolidating all the input attributes of the KT dataset into a single column, which is then encoded using BERT's native tokenizer (Figure 4.4).

Once the encodings are obtained, the input IDs and attention mask for each tokenized row sequence are extracted (Figure 4.5). These are subsequently provided to the `BERTForSequenceClassification` class from HuggingFace for processing.

Let's consider the example inputs provided to the `BertForSequenceClassification` model:

- **Input IDs:** [101, 1023, 1017, 17442, 102, 0]
- **Attention Mask:** [1, 1, 1, 1, 1, 0]

These inputs represent a sequence of tokens and a corresponding attention mask. Here's a step-by-step explanation of how the prediction happens:

```python
def preprocess_data(data, labels_column=None):
    # Combine the columns into a single string
    data['text'] = data['student_id'].astype(str) + ' ' + data['knowledgecomponent_id'].astype(str) + ' ' + data['question_id'].astype(
        str) + data['Correctness'].astype(str)

    # Tokenize the text
    encodings = tokenizer(data['text'].tolist(), truncation=True, padding=True)
    if labels_column is not None:
        encodings['labels'] = data[labels_column].tolist()

    return encodings
```

Figure 4.4: Preprocessing step for RIT dataset with Confidence as target.

```
For training
Input IDs: [101, 1015, 1018, 4185, 2487, 102]
Attention Mask: [1, 1, 1, 1, 1, 1]
Label: 0
For testing
Input IDs: [101, 1023, 1017, 17442, 102, 0]
Attention Mask: [1, 1, 1, 1, 1, 0]
```

Figure 4.5: Example input ids and attention mask for training and validation.

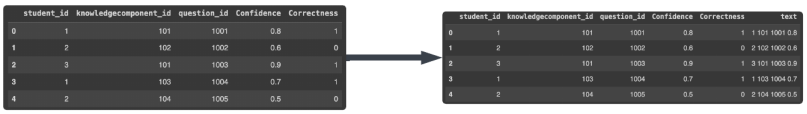

Figure 4.6: An example pre-processing step on dataset.

1. **Model Initialization:** The `BertForSequenceClassification` class is initialized with the configuration. This setup includes:
   - `self.bert` - The BERT model configured as per the specifications.
   - `self.dropout` - A dropout layer with a specified dropout rate.
   - `self.classifier` - A linear layer for classification.
2. **Input Preparation:** The input IDs and attention mask are prepared for the forward pass:
   - **input_ids:** [101, 1023, 1017, 17442, 102, 0]
   - **attention_mask:** [1, 1, 1, 1, 1, 0]
3. **Forward Pass:**
   - The `forward` method is called with `input_ids` and `attention_mask`.
   - The `self.bert` method processes these inputs, generating hidden states and a pooled output. The hidden states capture the contextual information for each token, while the pooled output represents the entire input sequence, typically corresponding to the `[CLS]` token (101).
   - For this example, let's assume the pooled output generated by BERT is a vector representation: `pooled_out`.
4. **Dropout Layer:**
   - The pooled output (`pooled_out`) is passed through the dropout layer to reduce overfitting: `pooled_out = self.dropout(pooled_out)`.
5. **Classification Layer:**
   - The output from the dropout layer is fed into the linear classifier to produce logits: `logits = self.classifier(pooled_out)`.
   - These logits represent the raw, unnormalized scores for each class.
6. **Output:**
   - The method returns the `logits`. If the configuration specifies `return_pooler_output`, it also returns the pooled output.

Example Output:

Given the input IDs [101, 1023, 1017, 17442, 102, 0] and attention mask [1, 1, 1, 1, 1, 0], the model processes the input sequence, applies dropout, and then classifies the pooled output to produce logits. These logits are then used to predict the class of the input sequence.

For instance, if the model is performing binary classification (e.g., Correctness prediction), the logits might be something like `[-0.5, 1.2]`, indicating that the model predicts the sequence belongs to the "incorrect" class (the second class) since the second logit has a higher value.

4.4 Evaluating KMA and KMB

The combination of correctness and confidence, obtained at each step for every learner across various datasets, provides a rich source of information for evaluating the Knowledge Mastery (KMA) and Knowledge Behavior (KMB) of each learner.

After each response from a learner, and once the model completes its prediction of either correctness or confidence, these KMA and KMB values can be evaluated. This evaluation is not just a one-time process but a continuous one, reflecting the dynamic nature of learning.

These KMA and KMB values are pivotal. They can be utilized by Intelligent Tutoring Systems (ITS) or instructors to provide tailored instruction to their students. This instruction is not generic but individualized, catering to the unique cognitive and metacognitive improvements reflected by the KMA and KMB values of each learner.

Therefore, these KMA and KMB values are not just evaluated but also recorded. This allows for future exploration of these values to identify the most effective interventions for a specific KMA-KMB pair for a learner. This approach ensures that the instruction is not only personalized but also adaptive, responding to the evolving needs of the learner.

In this way, the use of correctness and confidence in evaluating KMA and KMB values contributes to a more nuanced understanding of a learner's knowledge state, enabling more effective and personalized instruction.

Performance	**Confidence Report**
Always answers correctly	Always predicts to answer correctly
Always answers incorrectly	Always predicts to answer partially correctly
Occasionally answers correctly	Always predicts to answer incorrectly
Occasionally answers incorrectly	Always predicts to answer correctly but occasionally predicts to answer partially correctly
Progressively performs better	Always predicts to answer correctly but occasionally predicts to answer incorrectly
Regresses in performance	Always predicts to answer partially correctly but occasionally predicts to answer correctly
Progressively performs better then regresses	Always predicts to answer partially correctly but occasionally predicts to answer incorrectly
Regresses then progressively performs better	Always predicts to answer incorrectly but occasionally predicts to answer correctly
	Always predicts to answer incorrectly but occasionally predicts to answer partially correctly
	Progressively improves in prediction
	Regresses in prediction
	Progressively improves then regresses in prediction
	Regresses then improves in prediction

Table 4.2: Assumed Learner Behaviours - Performance Vs Confidence

Chapter 5

Experiments

In this section, we delve into the practical implementation and evaluation of the proposed BERT-Boosted Knowledge Tracing (KT) framework. The experiments are structured to assess the framework's adaptability and efficacy in predicting both correctness and confidence levels, thereby offering a comprehensive understanding of learners' cognitive processes.

Each experiment revolves around three pivotal datasets: a Synthetic Dataset, DBE-K22 Dataset (open-source) and RIT Dataset. Each of these datasets have varying sizes (DBE-K22 being largest and RIT dataset being smallest) and number of attributes. However, each of these dataset have common attributes: correctness and confidence as they serve to be primary target columns. These disparate datasets enables the KT framework to exhibit flexibility in its predictive capabilities, flexibility in handling various kinds of input attributes' combination and to showcase domain adaptability, where the model trains well on a very small number of training data points. Specifically, experiments are conducted by either utilizing confidence and other input parameters to predict correctness or employing correctness alongside additional input parameters to predict confidence.

This dual-pronged approach is essential to accommodate diverse educational settings. For instance, when integrated with Intelligent Tutoring Systems (ITSs) employing Reflection Assistant models aimed at fostering metacognitive skills, the framework can adapt by predicting correctness from confidence levels. Conversely, in scenarios where training for metacognition alongside cognitive skills may pose challenges, the model can predict confidence based on correctness and other input parameters. This strategic flexibility ensures that the framework remains versatile, catering to the unique needs of various educational contexts.

At the core of these experiments lies the evaluation of Knowledge Mastery Assessment (KMA) and Knowledge Mastery Benchmarking (KMB). KMA and KMB are essential metrics to assess learners' comprehension and guide personalized interventions. By being capable of either predicting correctness or confidence, the framework ensures that at the end of the predictions, both correctness and confidence values for the learner are available, which can then be used to evaluate the learner's Awareness (KMA) and Outlook (KMB), thereby enabling the formulation of targeted interventions tailored to individual learning trajectories.

5.0.1 Synthetic Dataset

The synthetic dataset curated for the study was first experimented with, as it gives a good benchmark and indication of the direction we go and also provides significant help to tackle the restrictions around obtaining educational data. In this subsection, we detail the experiments conducted using the Synthetic dataset to evaluate the performance of the BERT-Boosted Knowledge Tracing framework. Initially, two variants of BERT models, namely bert-base-uncased and bert-large-uncased, were employed to assess their effectiveness in predicting correctness and confidence levels.

Choice of BERT Models

The main distinction between bert-base-uncased and bert-large-uncased lies in their scale and computational requirements. While both models utilize the same architecture, bert-large-uncased incorporates a larger number of layers and parameters, allowing for potentially richer representations of text sequences. However, this enhancement comes at the cost of increased computational resources and inference time.

Experimental Findings

Upon conducting experiments, it was observed that both **bert-base-uncased** and **bert-large-uncased** yielded comparable results in predicting correctness when confidence levels were provided as input and vice versa. However, the larger model, bert-large-uncased, incurred significantly longer processing times because of its increased complexity.

Given that the performance gap between the two models was marginal and considering the practical implications of computational efficiency, it was deemed more pragmatic to proceed with bert-base-uncased for further experiments.

Uncased Notation Explanation

It is noteworthy to mention that the "uncased" suffix in the model names refers to the fact that these models are trained on uncased text, meaning that the text input to the models is converted to lowercase before processing. This design choice allows the models to generalize better across different cases and makes them suitable for various text-related tasks without being case-sensetive.

Model Training

Equation 3.1 provides an overview of the model training process, encompassing data preprocessing, model initialization, and optimization.

The training process begins with data preprocessing, denoted by the function Preprocess(), where the dataset D is transformed into a suitable format for training. Specifically, the columns relevant to the prediction task, namely learner skill, opportunity, confidence, and answer, are preprocessed and balanced around the target variable of interest, denoted by L. For instance, when predicting confidence, the dataset is balanced around the confidence column, and L is set to confidence in the preprocessing step.

Subsequently, the preprocessed data Denc is loaded into a DataLoader object for efficient batch-wise processing. The BERT model M and its tokenizer T are initialized, followed by the optimization step where the model parameters are optimized using an optimizer O, which in this case is **AdamW**, with a learning rate of **1e-5**.

The training loop iterates over a predetermined number of epochs E (10 in the case of synthetic dataset experiments conducted), during which the model is trained on the training dataset. For each epoch, the model is trained batch-wise, where the optimizer is first zeroed to reset gradients, input data is fed into the model, and the loss is computed based on the model's predictions compared to the ground truth labels. The loss is then backpropagated through the network, updating the model parameters to minimize the loss. This process iterates until convergence or until the specified number of epochs is reached.

Experimental Results

The training process was conducted on both CUDA and MPS platforms to assess the model's performance across different computing environments. The performance metrics for predicting correctness and confidence is listed in table 5.1.

Metric	AUC	RMSE	Accuracy (ACC)
Correctness (CUDA)	0.8835	0.5843	0.8834
Correctness (MPS)	0.8693	0.6014	0.8692
Confidence (CUDA)	0.8828	0.7483	0.8459
Confidence (MPS)	0.7818	0.8912	0.7422

Table 5.1: Performance Metrics - Synthetic Dataset

These results indicate that the BERT-Boosted Knowledge Tracing framework exhibits robust performance in predicting both correctness and confidence levels across different computational platforms.

5.0.2 DBE-KT22 Dataset

The DBE-KT22 Dataset serves as the cornerstone of these experimental endeavors, providing a comprehensive repository of educational data tailored to Knowledge Tracing (KT) tasks. The dataset architecture adheres to a relational model, wherein each data aspect is assigned to a distinct entity or table, thereby preserving relationships across entities through primary-foreign key pairs. Figure 5.1 depicts the entity-relationship diagram (ERD) encapsulating the structural layout of the DBE-KT22 database and Table 5.2 lists down and describes the entity types pertaining to DBE-K22 datasets.

The ERD is deconstructed into a series of CSV files, following the Comma Separated Value (CSV) format for ease of data sharing and distribution. Additionally, a Python script accompanies the dataset files, facilitating the generation of training sequences for question answering tasks, incorporating relevant meta-data to enrich the learning context.

Central Data Source:

The primary data source driving this analysis is the dbe_transaction table, which records question answering transactions, including answer states and associated meta-data. This table serves as the foundation upon which the experiments are built, providing rich insights into learners' interactions with educational content.

Data pre-processing

Pre-processing the DBE-K22 dataset involved several steps to ensure its suitability for Knowledge Tracing (KT) tasks, including handling missing values, encoding categorical features, and incorporating question text as model input.

Handling Missing Values: One significant aspect of the pre-processing stage was addressing missing values in the confidence column, considering its pivotal role as a target variable alongside correctness. Specifically, missing confidence values were treated as a priority due to their potential to disrupt the sequential structure of the data. In a sequential learning context, missing confidence values can introduce discontinuities in the learning trajectories, hindering the model's ability to capture temporal dependencies and longitudinal patterns effectively. To mitigate this issue, missing values in the confidence column were identified and replaced using a predictive modeling approach. Specifically, a RandomForestClassifier was trained on rows with known con-

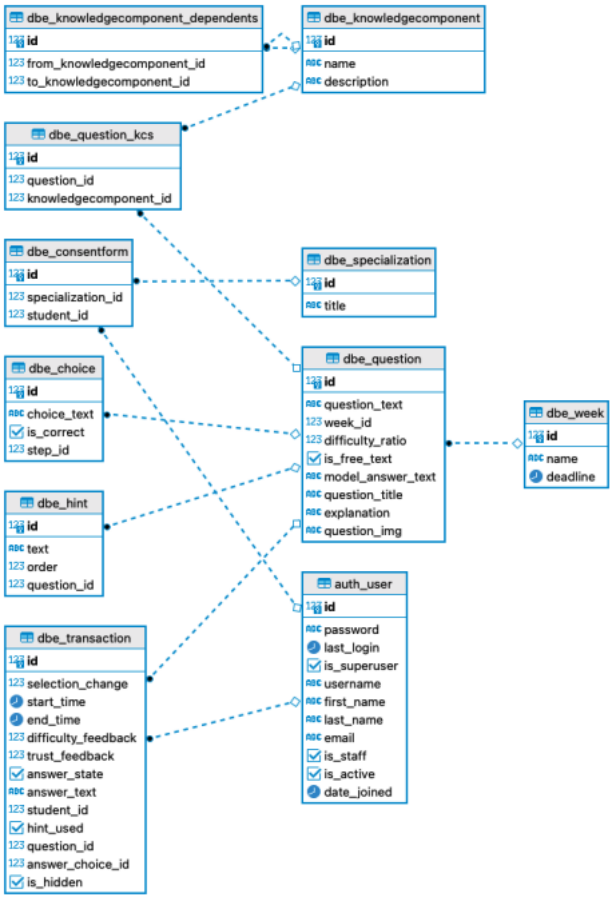

Figure 5.1: ER Diagram for DBE-K22 [6]

fidence values, and the model was used to predict confidence values for rows with missing values.

Encoding Categorical Features: Categorical features such as knowledgecomponent_name, question_text, hint_used, prev_correct, and prev_hint_used were encoded using Label Encoding. This transformation facilitates the incorporation of categorical variables into machine learning models, ensuring compatibility with algorithms that require numerical inputs. Notably, encoding question text allows the model to leverage the semantic information embedded within the text, enhancing its understanding of the underlying concepts and potentially improving KT performance.

Utilizing Previous Columns: In addition to traditional KT features, the dataset includes columns prefixed with "prev_" denoting previous interactions or states. These columns, such as prev_difficulty_feedback, prev_confidence, prev_correct, prev_hint_used, and prev_time_delta, provide valuable context regarding learners' past behavior and performance. Incorporating these features into sequence classification tasks enables the model to capture temporal dependencies and longitudinal learning trajectories, thus enhancing its predictive capabilities.

Significance of Using Question Text: In traditional KT approaches, question features are typically represented by abstract identifiers or attributes. However, by including the actual text of the questions as encoded input, the model gains access to rich semantic information embedded within the questions. This approach not only enriches the feature space but also allows the model to leverage the textual context to infer latent knowledge states more effectively. Consequently, utilizing question text as encoded input enhances the model's interpretability and promotes better learning outcomes by capturing nuanced relationships between questions and learner responses.

Final Dataset Structure and Pre-processing

After pre-processing, the DBE-K22 dataset was transformed into a structured format conducive to Knowledge Tracing tasks. The final dataset retained the columns as described in table 4.3:

In addition to structuring the dataset, special attention was given to maintaining the sequential integrity of the data. Sequences of interactions between students and educational content were preserved to capture temporal dependencies and longitudinal learning trajectories effectively.

Furthermore, to ensure balanced representation of classes and mitigate potential biases in the dataset, a sampling strategy was employed. The dataset was balanced around either correctness or confidence, depending on the specific

use case, using the RandomUnderSampler from the python imbalance-learn package. This approach helped to address class imbalances and ensure that the model learns from a diverse and representative set of examples, thereby enhancing its generalization capability and robustness in Knowledge Tracing tasks.

Model Training

The DBE-K22 dataset presented a sizable volume of data, with an initial source dataset length of 305,794 instances. All experiments related to the DBE-K22 dataset were conducted in a CUDA environment on Google Colab, utilizing a free tier T4 GPU for computational acceleration.

Correctness Prediction

After employing the undersampling balancing strategy, the dataset was balanced around correctness, resulting in the following breakdown:

- Correct (1): 71,021 instances
- Incorrect (0): 71,021 instances

The model training process remained consistent across experiments, employing the AdamW optimizer with a learning rate of 1e-5. Due to limited hardware resources on the free-tier Google Colab, model training was conducted for 3 epochs. From the balanced dataset, 30% of the instances were reserved for testing, while the remaining 70% were used for training.

Results:

- AUC: 0.8558
- RMSE: 0.6163
- Accuracy (ACC): 0.8557

Confidence Prediction

For confidence prediction, a similar undersampling balancing strategy was employed, resulting in a balanced dataset with the following distribution:

- Low Confidence (1): 22,603 instances
- Medium Confidence (2): 22,603 instances
- High Confidence (3): 22,603 instances

All other settings remained consistent with the correctness prediction task.

Results:

- AUC: 0.9369
- RMSE: 0.6297
- Accuracy (ACC): 0.9159

Overall, the experiments conducted on the DBE-K22 dataset yielded promising results, demonstrating the efficacy of the proposed approach in predicting both correctness and confidence levels in learners' responses even while dealing with environmental constraints for training on a relatively bigger dataset.

5.0.3 RIT Dataset

The experiments conducted over the Rochester Institute of Technology (RIT) data were constrained by the limited participation of learners, with only 11 individuals contributing to a total of 550 unique learner data points. Maintaining the anonymity of learners was a priority during data collection to ensure privacy and compliance with ethical guidelines. The data collection procedure was set to simulate the Reflection Assistance setup, where the learners essentially rated their confidence before solving a problem. The table 5.4 describes the curated dataset and it's attributes in detail. Due to the scarcity of data points, oversampling techniques were explored to maximize the utilization of available data.

Oversampling Experiments:

For oversampling, the Synthetic Minority Over-sampling Technique (SMOTE) from the imbalance-learn package was utilized. This technique aims to generate synthetic samples for minority classes, resulting in a more balanced dataset. After applying SMOTE, the resulting balanced dataset contained more data points than the original dataset.

Results for Correctness Prediction

After balancing the dataset using SMOTE for correctness prediction, the results were as follows:

- AUC: 0.7829
- RMSE: 0.6891
- Accuracy (ACC): 0.7745

The balanced dataset, with equal representation of correctness labels, had the following distribution:

- Count of Correctness = 1: 340

- Count of Correctness = 0: 340

Results with Undersampling:
Subsequently, the predictions were repeated after balancing the dataset using undersampling. For correctness prediction, the balanced dataset had the following distribution:

- Count of Correctness = 0: 159
- Count of Correctness = 1: 159

The results obtained with undersampling were notably improved:

- AUC: 0.9677
- RMSE: 0.4204
- Accuracy (ACC): 0.9688

Similarly, for confidence prediction, the balanced dataset after undersampling exhibited the following distribution:

- 0: 103, 2: 103, 1: 103

The results seem to not remain consistent with correctness prediction as the evaluation metrics suggest that confidence prediction performs not as well as correctness prediction for RIT dataset :

- AUC: 0.7262
- RMSE: 0.9252
- Accuracy (ACC): 0.6335

5.0.4 Experimentation Summary and Comparison with other Metacognitive Models

When comparing the performance of the KT Fine-Tuned BERT model from the BERT-Boosted KT Framework with existing models known for efficient knowledge tracing, interesting observations emerge:

Predicting Correctness:

When comparing the performance of the KT Fine-Tuned BERT model from the BERT-Boosted KT Framework with other known models that utilize metacognition, we refer to a seminal paper titled *"Knowledge Tracing for Adaptive Learning in a Metacognitive Tutor"* [4]. In this study, the authors leverage metacognitive inputs (specifically confidence reports from an RA model) to train two distinct models:

- **RA-BKT** (Bayesian Knowledge Tracing): RA-BKT is a Bayesian Knowledge Tracing model equipped to handle metacognitive inputs. It incorporates confidence information to enhance its predictions of learner correctness.
- **RA-ANN** (Artificial Neural Network): RA-ANN is an ANN (Artificial Neural Network) designed for Knowledge Tracing with the use of metacognitive inputs. Similar to RA-BKT, it leverages confidence reports to improve its performance.

The authors then compare the results of these two developed models (RA-BKT and RA-ANN) with a baseline BKT (Bayesian Knowledge Tracing) and ANN (Artificial Neural Network) that predict correctness without considering metacognitive inputs. Interestingly, both RA-ANN and RA-BKT outperform the traditional BKT and ANN for the Knowledge Tracing task, highlighting the importance of incorporating metacognitive information.

Now, let's turn our attention to the BERT-based KT model fine-tuned using the BERT Boosted Framework and similar metacoginitve inputs as used in RA-ANN and RA-BKT. In an attempt to ensure the robustness of the comparison, the synthetic dataset creation process in this study, follows the steps closely as described by May Kristine Jonson Carlon and Jeffrey S. Cross in their paper. This model demonstrates superior performance compared to RA-BKT and RA-ANN. The comparison results are summarized in the table 5.5.

It's essential to note that while developing RA-ANN and RA-BKT, the authors consider KMA and KMB as inputs to the model. However, in real-world scenarios of KT models, KMA and KMB cannot be evaluated beforehand. These metacognitive inputs depend on both confidence and correctness values for a learner's interaction with a problem. Since our goal is to predict correctness, KMA and KMB are not available as input parameters upfront. Therefore, the comparison results for the BERT-based KT model do not take KMA and KMB into account.

Predicting Confidence:

The landscape of predicting learner confidence in problem-solving scenarios remains relatively uncharted. A comprehensive literature review revealed a scarcity of existing work that specifically addresses this domain. Surprisingly, no established benchmarks or widely recognized models have emerged to guide researchers and practitioners in this area.

However, the exploration done in this study pertaining to confidence prediction has yielded promising results. The accuracy of confidence predictions suggest that this avenue holds potential for future investigation. As we continue to unravel the intricacies of learner behavior and cognition, predicting confidence levels could become a pivotal aspect of personalized learning experiences.

Experimentation Summary

The models developed as a part of the experimentation conducted for learners' correctness and confidence prediction are summarized in table 5.6 along with their respective evaluation metrics.

Table Name	Description
dbe_question	This pivotal table encapsulates all question meta-data, serving as the primary repository for question-related information.
dbe_choice	Contains meta-data pertaining to choices associated with each question, facilitating the exploration of multiple-choice questions.
dbe_hints	Stores hint data tailored to questions, aiding learners in navigating complex problem-solving scenarios.
auth_user	Houses user login credentials and contact information, essential for user authentication and communication purposes.
dbe_transaction	Records question answering transaction data, including the state of answers, forming the central data source for our analysis.
dbe_week	Contains weekly exercise set data, organizing exercises into weekly structures to facilitate course management.
dbe_consentform	Stores student consent data, including specialization inquiries and confirmation on course policies.
dbe_specialization	Serves as a lookup table for specialization data, facilitating the categorization of students based on their areas of focus.
dbe_knowledgecomponent	Holds meta-data pertaining to knowledge components (KCs) within the course, crucial for understanding the underlying cognitive processes.
dbe_knowledgecomponents_dependendts	Stores relationships between knowledge components, enabling the exploration of hierarchical knowledge structures.
dbe_question_kcs	Records relationships between questions and knowledge components, facilitating the alignment of questions with specific learning objectives.

Table 5.2: DBE-K22 Entity Types

Field	Description
student_id	Identifier for individual students participating in the learning activities.
knowledgecomponent_id	Identifier for knowledge components (KCs) associated with the learning content.
knowledgecomponent_name	Name or label corresponding to knowledge components.
question_id	Identifier for individual questions posed to learners.
question_text	Textual content of the questions, encoded for machine processing.
gt_difficulty	Ground truth difficulty level assigned to questions by course instructors.
difficulty_feedback	Student's feedback on the perceived difficulty level of questions.
hint_used	Binary indicator denoting whether a hint was utilized for a given question.
prev_difficulty_feedback	Student's feedback on the perceived difficulty level of previous questions.
prev_confidence	Confidence level associated with previous responses.
prev_correct	Binary indicator denoting correctness of previous responses.
prev_hint_used	Binary indicator denoting whether a hint was utilized for previous questions.
prev_time_delta	Time difference between current and previous responses, capturing temporal dynamics.
confidence	Confidence level associated with current responses (target variable).
correct	Binary indicator denoting correctness of current responses (target variable).
text	Additional textual information, such as comments or explanations, if available.

Table 5.3: Description of Fields in the DBE-K22 Dataset

Field	Description
student_id	Identifier for individual students participating in the learning activities.
knowledgecomponent_id	Identifier for knowledge components (KCs) associated with the learning content.
question_id	Identifier for individual questions posed to learners.
Confidence	Confidence level associated with current responses (target variable).
Correctness	Binary indicator denoting correctness of current responses (target variable).

Table 5.4: Description of Fields in the RIT Dataset

Table 5.5: Model Comparison Summary

Model	Accuracy
BERT-Boosted Correctness Prediction (Synthetic Dataset (CUDA Training))	0.8834
BERT-Boosted Correctness Prediction (Synthetic Dataset (MPS Training))	0.8692
RA-BKT	0.845
RA-ANN	0.864

Table 5.6: Performance summary of models trained

Model	**AUC**	**RMSE**	**Accuracy**
BERT-Boosted Confidence Prediction (Synthetic Dataset (CUDA Training))	0.8828	0.7483	0.8459
BERT-Boosted Confidence Prediction (Synthetic Dataset (MPS Training))	0.7818	0.8912	0.7422
BERT-Boosted Correctness Prediction (Synthetic Dataset (CUDA Training))	0.8835	0.5843	0.8834
BERT-Boosted Correctness Prediction (Synthetic Dataset (MPS Training))	0.8693	0.6014	0.8692
BERT-Boosted Correctness Prediction (DBE-K22 Dataset (CUDA Training))	0.8558	0.6163	0.8557
BERT-Boosted Confidence Prediction (DBE-K22 Dataset (CUDA Training))	0.9369	0.6297	0.9159
BERT-Boosted Correctness Prediction (Over-sampled RIT Dataset (CUDA Training))	0.7828	0.6891	0.7745
BERT-Boosted Correctness Prediction (Under-sampled RIT Dataset (CUDA Training))	0.9677	0.4204	0.9687
BERT-Boosted Confidence Prediction (Under-sampled RIT Dataset (CUDA Training))	0.7262	0.9252	0.6335

Chapter 6

Conclusion

In conclusion, the experiments conducted across various datasets, including synthetic data, open-source dataset, and data collected from the Rochester Institute of Technology (RIT), have demonstrated the versatility and efficacy of the proposed framework for Knowledge Tracing tasks. Through these experiments, it is evident that the framework exhibits the capability to adapt to different input sequences based on the specific use case requirements.

The framework's flexibility was showcased through its ability to accommodate varying input parameters, such as correctness, confidence, question text, and metadata, allowing for customization tailored to the learning context. Moreover, the framework showcased robust performance across different datasets, indicating its potential applicability in diverse educational settings.

6.1 Observations over each model creation

Synthetic Dataset Models:

The models created using synthetic datasets played a crucial role in establishing a baseline success metric for the approach discussed in this report. Given that metacognitive input for knowledge tracing (KT) models is not abundantly explored in the KT literature, it was challenging to ascertain whether the framework developed would yield the desired results. By meticulously constructing synthetic datasets following the steps outlined in the paper "Knowledge Tracing for Adaptive Learning in a Metacognitive Tutor," the BERT KT models trained through our framework were compared with the originally proposed RA-BKT and RA-ANN by the authors. This comparison provided valuable insights into the experimental direction. Subsequently, we extended these ideas to other datasets.

DBE-K22 Dataset Models:

Finding a real-world learner dataset with metacognitive attributes proved to be a challenge. While widely used KT datasets like ASSISTments lack metacognitive attributes, the DBE-K22 dataset came to the rescue. Its rich and abundant data allowed us to explore the framework's capabilities further. Notably, the use of DBE-K22 data demonstrated that with more training examples, the framework could curate more accurate KT models. The dataset's unique attributes, such as historical learner interactions with course material, facilitated easy experimentation and highlighted the framework's helpfulness.

RIT Dataset Models:

Despite the limited data points in the RIT dataset, the KT models performed remarkably well for correctness prediction. Intriguingly, when the data was augmented through oversampling to balance the dataset, the resulting models were less accurate, even though they were trained on more data. However, when the framework was leveraged to develop models with undersampled data, accuracy increased. This observation underscores the framework's domain adaptability, allowing seamless adjustments to changing learning environments.

In summary, these observations collectively emphasize a grand theme: the framework opens up a multitude of possibilities for exploring and understanding KT models. Its flexibility, adaptability, and ability to handle diverse datasets empower researchers and educators alike in their pursuit of effective adaptive learning solutions.

6.2 Future Work

The Knowledge Tracing (KT) framework developed in this study holds promise for future advancements in Intelligent Tutoring Systems (ITS). The framework's flexibility and adaptability make it a valuable tool for exploring various hypotheses and addressing diverse educational challenges. It is envisioned that the output capabilities of the framework could be leveraged by future researchers and developers to quickly explore new ideas and innovations in the field of KT.

6.2.1 Meta-cognitive monitoring along with cognitive monitoring

One promising avenue for future research is the integration of meta-cognitive monitoring alongside cognitive monitoring within the KT framework. Meta-cognitive monitoring involves assessing learners' awareness (KMA) and outlook (KMB) regarding their own learning process. By evaluating KMA and KMB alongside traditional cognitive measures, such as correctness and confidence prediction, the framework can provide a more holistic understanding of learners' cognitive processes.

Having access to KMA and KMB values for each learner opens up opportunities for personalized interventions within ITS platforms. For example, if a learner exhibits low KMA or KMB, automated interventions can be triggered using fine-tuned Language Model prompts. These interventions can address specific learning needs, such as addressing shallow learning by providing additional challenging questions that require deeper understanding.

In ITS systems facilitated by instructors, KMA and KMB values can be provided to instructors to guide their interventions. For instance, instructors can intervene when learners exhibit signs of shallow learning, thereby enhancing the effectiveness of the learning experience. The flexibility provided by the framework enables the prediction of confidence values even in scenarios where learner-derived data is not available. This ensures that metacognitive training can be integrated seamlessly into the learning process without adding unnecessary burden.

Shallow learning occurs when learners acquire superficial knowledge without fully grasping underlying concepts or principles. By evaluating KMA and KMB alongside cognitive measures, such as correctness and confidence prediction, the KT framework can identify instances of shallow learning more effectively. For example, if a learner consistently demonstrates high correctness but low confidence or exhibits inconsistent meta-cognitive monitoring patterns, it could indicate a reliance on memorization or surface-level understanding rather than deep comprehension.

One of the key advantages of integrating KMA and KMB into the KT framework is the flexibility it offers in assessing confidence levels. Traditional approaches rely on learners' self-reported confidence values, which may not always be available or reliable, especially in automated learning environments. However, by leveraging the framework's predictive capabilities, confidence levels can be inferred from learners' correctness patterns. This eliminates the need for explicit confidence reporting and ensures a seamless integration of meta-cognitive assessment into the learning process.

Furthermore, by automating the prediction of confidence levels, the burden of metacognitive training is significantly reduced. In traditional approaches, explicit metacognitive training often requires additional time and resources, potentially detracting from the primary learning objectives. However, with the predictive capabilities of the KT framework, metacognitive assessment can be seamlessly integrated into the learning process without imposing an additional burden on learners or instructors.

In summary, integrating meta-cognitive monitoring alongside cognitive monitoring within the KT framework opens up new avenues for addressing educational challenges and enhancing personalized learning experiences. By leveraging the framework's capabilities, future research can explore innovative approaches to support learners' cognitive and meta-cognitive development in educational settings.

6.2.2 Leveraging Domain Adaptability

The utilization of pre-trained language model, such as BERT (Bidirectional Encoder Representations from Transformers), facilitates domain adaptability within the Knowledge Tracing (KT) framework. Specifically, employing the bert-base-uncased model allows for seamless adaptation to various educational domains, as evidenced by the experiments conducted, particularly with the RIT dataset. Despite being the smallest dataset among the three examined, the model demonstrated the ability to quickly adapt, underscoring its domain adaptability.

Domain adaptability in KT is paramount for accommodating changes in learner subjects or classes. As learners transition between different subjects or courses, their learning patterns and behaviors may vary. By leveraging domain-adaptable models like BERT, the KT framework can efficiently adjust to these changes without the need for extensive retraining or customization. This ensures that the model remains effective and accurate across diverse educational contexts.

The significance of domain adaptability in KT lies in its ability to enhance the scalability and applicability of intelligent tutoring systems (ITS). As educational environments continue to evolve, ITS platforms must be capable of accommodating dynamic changes in curriculum, instructional methods, and learner populations. By incorporating domain-adaptable KT frameworks, educators and developers can create more robust and flexible ITS solutions that effectively address the evolving needs of learners.

Moreover, domain adaptability enables KT frameworks to generalize knowledge and insights gained from one educational domain to others. This transfer-

ability of knowledge enhances the efficiency of model training and accelerates the deployment of KT solutions in new educational settings. Additionally, it promotes knowledge sharing and collaboration among educators and researchers across different domains, fostering innovation and advancement in educational technology.

6.2.3 Leveraging Framework's Flexibility

The framework's inherent flexibility lies in its ability to adapt to varying target variables and input attribute sequences, offering researchers the opportunity to explore the significance of different attributes in Knowledge Tracing (KT) problems. This flexibility is exemplified by the framework's capability to seamlessly switch between predicting confidence and correctness, depending on the availability of learner responses and the specific use case.

One of the key aspects of the framework's flexibility is its adaptability to different target variables. Researchers can choose to predict either confidence or correctness based on the requirements of their study or the characteristics of their dataset. For instance, predicting confidence may be more relevant in scenarios where learners' self-assessment is critical for understanding their learning progress, while predicting correctness may be preferable in contexts where objective performance evaluation is prioritized.

Furthermore, the framework's flexibility extends to the sequence of input attributes used for prediction. Researchers have the flexibility to experiment with different attribute sequences, allowing them to explore the impact of various factors on KT performance. For example, some researchers may investigate the influence of time attributes, such as response time or time since last interaction, on model training and prediction accuracy. Others may focus on attributes related to learner behavior, such as hint usage or previous correctness patterns, to gain insights into learning strategies and cognitive processes.

Moreover, the framework's flexibility opens up possibilities for exploring the significance of additional attributes beyond the standard features commonly used in KT models. Researchers may choose to incorporate domain-specific attributes, learner demographics, or instructional features into the input sequence to better capture the complexities of the learning environment. This flexibility enables researchers to tailor the KT framework to their specific research questions and hypotheses, fostering innovation and advancement in the field of educational data mining.

Bibliography

[1] Vincent Aleven and Kenneth R Koedinger. Limitations of student control: Do students know when they need help? pages 292–303, 2000.

[2] Vincent AWMM Aleven and Kenneth R Koedinger. An effective metacognitive strategy: Learning by doing and explaining with a computer-based cognitive tutor. *Cognitive science*, 26(2):147–179, 2002.

[3] Okan Bulut, Jinnie Shin, Seyma N Yildirim-Erbasli, Guher Gorgun, and Zachary A Pardos. An introduction to bayesian knowledge tracing with pybkt. *Psych*, 5(3):770–786, 2023.

[4] May Kristine Jonson Carlon and Jeffrey S. Cross*. Knowledge tracing for adaptive learning in a metacognitive tutor. *Journal Name*, X(Y):Z–W, Year.

[5] Claudia Gama. Metacognition in interactive learning environments: The reflection assistant model. pages 668–677, 2004.

[6] Qing Wang Yu Lin Ghodai Abdelrahman, Sherif Abdelfattah. Dbe-kt22: A knowledge tracing dataset based on online student evaluation. *arXiv*, 3, 2023.

[7] Diane F Halpern. *Thought and knowledge: An introduction to critical thinking*. Psychology press, 2013.

[8] Neil T Heffernan and Cristina Lindquist Heffernan. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, 24:470–497, 2014.

[9] Sein Minn, Michel C Desmarais, Feida Zhu, Jing Xiao, and Jianzong Wang. Dynamic student classiffication on memory networks for knowledge tracing. In *Advances in Knowledge Discovery and Data Mining: 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, April 14-17, 2019, Proceedings, Part II 23*, pages 163–174. Springer, 2019.

[10] Shi Pu, Michael Yudelson, Lu Ou, and Yuchi Huang. Deep knowledge tracing with transformers. pages 252–256, 2020.

[11] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Contrastive bidirectional transformer for temporal representation learning. corr abs/1906.05743 (2019), 1906.

[12] Weicong Tan, Yuan Jin, Ming Liu, and He Zhang. Bidkt: Deep knowledge tracing with bert. pages 260–278, 2021.

[13] Sigmund Tobias, Howard T Everson, and Vytas Laitusis. Towards a performance based measure of metacognitive knowledge monitoring: Relationships with self-reports and behavior ratings. 1999.

[14] Huaibo Zhao. Deep learning in security: text-based phishing email detection with bert model, 2023.