An Architecture of Domain Independent and Extensible Intelligent Tutoring System based on Concept Dependencies and Subject Paths

Sanjay Singh, Vikram Singh
Department of Computer Science and Engineering
Chaudhary Devi Lal University, Sirsa, India

***Abstract*—Intelligent Tutoring Systems (ITS) seek to provide personalized tutoring to learners, but are often domain specific, and lack extensibility. When featuring extensibility and domain independence, it is a challenge to provide appropriate level of personalization for every learner. In this paper, an architecture of a system that features domain-independence and extensibility with personalization and automatic course improvements without requiring persistent subject expert intervention has been proposed. The proposed architecture utilizes the notion of concept dependencies and the ability to sequence inter-dependent concepts intelligently into subject paths that enable automated tutoring as well as effective course customization per learner. It features a separate interface for subject experts through which they do not require ITS building knowledge to fulfil their appropriately assigned tasks assisted intelligently by the system, and an API based interface layer that supports today's mobile requirements for better engagement.**

Keywords*—Personalized tutoring; intelligent tutoring system; adaptive learning

I. INTRODUCTION

Since centuries, students have been taught in classrooms, where there is one teacher and multiple students. A more personalized tuition may involve a teacher personally tutoring a single student. Ever since computers were invented, there has been a considerable effort at using them to mimic the teaching capabilities of a human teacher. Facilitating the job of teaching through the use of any electronic technology falls under the huge umbrella of e-learning, which is a far-reaching discipline that covers the analysis of all conjunctions of technology and education. A more appropriate definition by researchers in [1] states "Educational technology is the study and ethical practice of facilitating learning and improving performance by creating, using, and managing appropriate technological processes and resources".

Since a human teacher, besides barely delivering a classroom lecture, performs considerable communication with the students and adjusts their teaching in response to the students' learning progress, researchers have been aiming to bring the exact qualities into e-learning systems to provide better tutoring. The most important and in-fact the ideal characteristic for an intelligent tutor is the ability of this communication [2]. The years around the 1960s and 1970s saw many new Computer-Assisted Instruction (CAI) projects funded by big names such as IBM and HP, that looked at tutoring through a behaviorist perspective based on Skinner's theories [3]. While the CAIs were attracting interests, the researchers in [4] introduced Intelligent Tutoring Systems (ITS) and shed light on the idea that computers could act as a teacher rather than barely a tool. These systems realize established teaching-learning processes by way of AI (Artificial Intelligence) with an intention of delivering learner-adapted tutoring without any direct mediation of a human teacher. Intelligent tutoring systems combine AI, education theories, and psychological models of the student and the expert [5]. Thus, building truly intelligent tutoring systems requires experts from the AI community, psychology community and education community to come together. In a nutshell, an ITS aims to put AI technologies to use for the delivery of a teaching, which would have been branded as "Good Teaching" if it were delivered by a human teacher [6]. Many of the works have employed ethnographic and design research methods [7] to study the scenarios in which the teachers and learners actually use the ITSs, many a times disclosing unexpected needs that they met, failed to meet or even create in some cases.

Present day ITSs try to mimic the role of a teaching assistant, by trying to automate pedagogical functions such as problem generation and selection. Many recent works have focused on how ITSs can supplement the duties of an existing human teacher [8] in a classroom or a peer [9] in other social contexts. The field of intelligent tutoring systems have evolved into various sub-areas, like Dialogue Based Tutoring Systems [10] which provide tutoring to the student using natural language dialogue, Cognitive Tutors [11] which utilize a cognitive model (an approximation of animal cognitive processes) to provide feedback during the learning process, tutoring systems for Intelligent Computer Assisted Language Learning, etc.

The task of individualized one-to-one tutoring can be made more efficient if the tutoring machine is portable, leading us to the field of mobile learning, which allows a student to carry the tutor with them wherever they go. According to [12] M-learning is "learning across multiple contexts, through social and content interactions, using personal electronic devices". The researchers in [13] suggest that mobile technology combined with network technology can connect formal learning to informal learning. Formal learning refers to the education delivered in a traditional classroom environment by trained teachers and informal learning is any type of learning

which is not formal. Mobile learning has the strength of being portable, which leads to more responsiveness and the ability to provide instant feedback. The work of [14] observed that mobile learning can increase exam scores from the 50th to the 70th percentile and reduce the dropout rate by 22 percent in technical fields. Mobile learning is also relatively cheaper because the cost of mobile devices is relatively lesser than that of laptops and personal computers.

Intelligent tutoring systems are expensive to develop, are often built for a specific domain of study, and the domain knowledge contained in them is limited because it is usually only fed-in at the time of development of the system. This paper proposes a framework for developing a domain independent and extensible system, that does not require persistent availability of subject experts, and the following sections discuss the related work, the proposed system's architecture with description of its components and modules, and an evaluation of the modules of the system.

II. Related Work

The architecture of an ITS has greatly varied throughout the years in the works of various researchers but the core idea has remained the same – it can employ any architecture provided that it delivers intelligent tutoring, even though the original discussions on the architecture of an ITS describe these four crucial modules: The Student Model, The Domain Model, The Tutor (pedagogical) Model and The Interface Model [15–17]. The domain model, which can also be referred to as the knowledge model is the part of the system that contains the concepts, rules and problem-solving strategies of the domain to be learned. It helps fulfilling certain roles such as – being a source of knowledge, being a standard for evaluating learner's performance, etc. The student model is generally built on top of the domain model, as an overlay. It is often referred to as the core component of any ITS, as it deals closely with the learner's cognitive and affective states and their evolution as the learning process advances. The tutor (or pedagogical) model connects with the domain and student model of the system and makes decisions about the tutoring actions and strategies. This model is responsible for guiding the learner through the overall learning process making sure the learner does not deviate from the particular tutoring strategy adopted by the system. The interface model "integrates three types of information that are needed in carrying out a dialogue: knowledge about patterns of interpretation (to understand a speaker) and action (to generate utterances) within dialogues; domain knowledge needed for communicating content; and knowledge needed for communicating intent"

It is in-fact extremely rare to find two different scratch ITSs with the same architecture. The word Scratch ITS refers to all those ITSs that have been built from the ground up and have not been authored using some ITS building framework. Out of these, there have been ITSs based on three-model, four-model and other varieties of architecture.

A three-model architecture is based upon the declaration of three major core components – domain knowledge, student knowledge and tutoring knowledge. Researchers in [18] have proposed an ITS with the expert domain model, tutoring model and student knowledge model being the three major components. The expert domain model and the student knowledge model guide the procedures in the tutoring model by providing the necessary information. The tutoring model is the most well-defined part of this architecture and it has various sub-components for curriculum planning, tutorial intervention and lesson planning. Another important three-model architecture is in the work of [19]. It also comprises of a domain model, a tutoring model and a student model but the difference between this architecture and the previous one is that it also incorporates an additional process – an overall system control process to co-ordinate the three models. This architecture also extends the lesson planning and dynamic adaptation concepts from the previous architecture to facilitate multiple tutoring strategies and information representations.

The three-model architectures made way for the classical standard four-model architectures. These architectures contain the three core components discussed in the previous architectures and add a fourth – the user interface component. A typical example for the four-model ITS architecture would be the work in [20] that has a knowledge base, a student model, a pedagogical model and a user interface. This architecture embodies cognitive and meta-cognitive processes in the student model and contains domain dependent tutoring rules in the pedagogical model. The key difference between this architecture and the architectures discussed in the previous section is that this architecture regards the user interface as an integral and internal component of the system whereas the previous section regarded this as a component external to the whole ITS system. This inclusion is helpful in that it concretizes the fact that a user-interface has a huge impact on the overall tutoring process, hence more efforts in user-interface design and development have become a concerning part of the overall ITS development process.

There are some architectures that take a deviance from the three-model and four-model architectures discussed. The idea of an intelligent learning environment has been promoted in MATHEMA [21] a multi-agent architecture which incorporates the notion of a human expert society (HES), a micro-society of artificial tutoring agents (MARTA) and external human motivators. This architecture surrounds the constituents of the classical architectures, although through a distinctive representation. The domain model is implanted within HES and MARTA, the user interface component is represented as interface agent, the role and functions of the tutoring model are dispensed among MARTA, and the human learner is not represented as a student model component as in other architectures, it is instead represented as a component of the learning environment itself. This architecture is suggested for well-structured, formal and specific knowledge domains.

There has been very limited work that has been done in the direction of domain independent ITSs, for they require considerable effort in representing each domain fairly. The researchers in [22] put forward ASSISTment builder tool that allows easy creation of ITSs that mimic cognitive tutors but the ITSs made with this are limited in that they only work for a single problem. These tutors provide a simple cognitive model based upon a state-graph tailored to a specific problem. Other aspects that have been rarely touched are extensibility – the ability to considerably extend the knowledge contained in the

system even after the system has been deployed. AutoTutor [23] is an intelligent tutoring system that helps students learn science, technology, and other technical subject matters by holding conversations with the student in natural language. Being able to extend the domain knowledge easily without requiring expert ITS development knowledge and doing that on-the-fly still remains a challenge. Another challenge is portability – for having a greater amount of time of the day with the learner, which brings various opportunities for better tutoring motivation. For solving all these challenges and more, in the following section, an architecture that ensures domain independence, extensibility, and portability, while ensuring personalization and adaptivity has been proposed.

III. CAPTAIN: Modules, Architecture and Workflow

This paper proposes Computer Assisted Personal Tutor with Adaptive INstruction (CAPTAIN) - a domain-independent ITS based upon concept dependencies and subject paths. It features extensibility – through automatic data generation, learner submitted questions and impersistent expert intervention, and portability – through an API based interface model that allows both desktop and mobile clients to interact with the ITS.

Its main strength is being able to expand its knowledge, improve itself, and be adaptable to any study area. It is based on five-module architecture and the modules have further specifications for sub-components as illustrated in detail in Fig. 1.

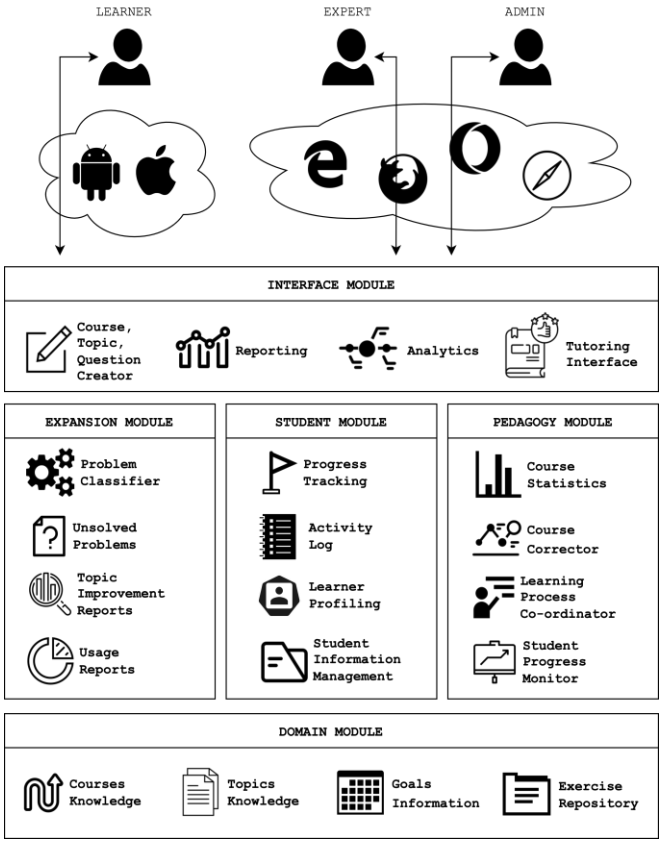

Fig. 1. Architecture Diagram of the System.

The domain module stores the domain knowledge. The architecture is not designed for any specific study domain, so the domain module is capable of storing knowledge about almost any subject area that can be represented through storing and linking modular topics, courses and study paths, along with exercises represented as questions and activities, and information about goals that drive the study paths and the whole tutoring flow.

The pedagogy module is made up of different components that share the common goal of assisting the tutoring process. It is responsible for using learner activity data for improving the quality of the courses that have been crafted by subject experts through its course corrector component, as well as coordinating the learning process and monitoring the progress of the learner.

The student module deals with storing and managing all the information about the learner. It deals with learner profiling – customizing the course according to the knowledge level of the learner in the course, and also deals with tracking the progress and logging the activities of the learner which are in turn used to by various other components of the system.

The expansion module is responsible for the ever-growing nature of the system. It has the problem classifier which maps the user submitted problems to their closest subject area for effective resolution by the appropriate subject expert, or prompts for creation of new topics if there aren't any appropriate ones available. It also provides the subject experts with topic improvement reports and information about how the courses are performing as well as the unsolved problems. It also provides the administrators the comprehensive usage reports of the system.

The interface module is responsible for facilitating the interaction between the learner and the system through mobile based platforms, and between the subject expert and the system as well as between the administrators/management and the system through the web-based platforms.

The system is designed and developed and has been deployed as three interconnected components – a backend server based on Django (python) that uses PostgreSQL as the database engine and serves a RESTful API, a web-app primarily for subject experts and administrators based on React JS, and a mobile app for the learners based on React Native.

In the following sections the key parts of the proposed architecture are explained in detail.

A. Basis

At the core of the system, the smallest and most re-usable atomic unit of information that exists to teach a fine-grained concept or skill has been termed a topic. Any subject area that is intended to be tutored must be fed into the system by subject experts in the form of small topics. The subject experts are expected to create topics that consist of that atomic bit of information which is content in itself.

The amount of information on a concept which is small enough that it is always expected to be studied together in one go, and that dividing which further will not make enough sense as there will always be another bit of information that always needs to be studied along with it. This makes the topics re-

usable and they can be used again and again in different courses, which in turn can be re-used in different study paths. As depicted in Fig. 2, in the proposed architecture, a topic is made up of any sequence of theory activities and question activities, along with an evaluation activity.

As it stated above that a topic needs to be fine grained, it may naturally require prior knowledge of zero or more other topics, which is necessary to represent complex concepts as simple learnable units. Topics are related with other topics via the 'requires' relation. This relationship can be realized via a graph, more precisely through a Directed Acyclic Graph representing topics as nodes and a topic's requirements as an edge from the topic's node to its required topic's node as illustrated in Fig. 3.

Since specifying a topic as a requirement for another topic is susceptible to possible creation of cyclic requirements (a situation in which topic A requires topic B and, directly or indirectly, topic B requires topic A), a measure has been implemented to prevent generation of cycles at the point where the subject expert adds a topic as a requirement for a topic.

Studying individual topics is good, but to study a subject or a large topic, one has to study a large number of topics in a certain sequence. This feature has been termed as a **course**, which is crafted by a subject expert by linking a set of topics together in a particular sequence so as to fulfill the aim of tutoring a subject or a large or complex topic. Since a topic may require prior knowledge of other topics, this puts some constraints on the sequencing of topics in a course, because a topic's required topics must only be sequenced before the topic. Also, how many of the topics' required topics will be included in the course is another decision the subject expert makes. If a course contains topics whose pre-required topics are not a part of the course, they become the requirements of the course itself, as illustrated in Fig. 4, with reference to the topic relationships illustrated in Fig. 3, as topics 9 and 7 require topic 2 and topics 10 and 7 require topic 6 but topic 2 and 6 are not part of the course, they become the requirements of the course, rest all other topics' requirements are being satisfied within the course.

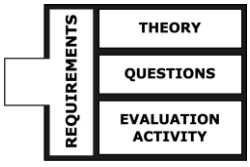

Fig. 2. Structure of a Topic in the System.

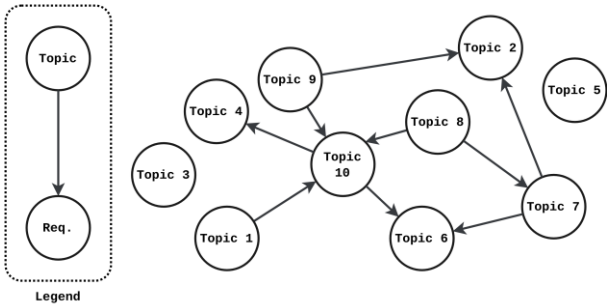

Fig. 3. Topic Relationships Illustrated through a DAG.

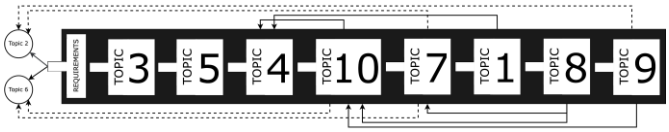

Fig. 4. Illustration of the Structure of a Course.

A course is also designed with such a level of granularity that it is independent in itself in teaching a subject or a large topic, and in this sense, it becomes a re-usable unit that can be studied by learners who have different goals but they are to study certain courses, in a certain sequence to achieve their specific goal. This level of sequencing has been termed a study path, which is an ordered collection of courses intended to prepare the learner for a specific goal, and this is the largest learnable unit in the system, and the least re-usable one.

In the system a goal refers to any concrete motive or objective for studying. The best example of a goal is any national level exam that is to be conducted, and there are many students that need to prepare for it. Goals are added by system administrators, the subject experts then create the study paths for them by re-using the courses in the system or by creating new courses, by in-turn re-using the topics in the system or by creating new topics, while mapping dependency relations to other topics if necessary. Fig. 5 illustrates the operation from the learner's goal selection to the system's tutoring process.

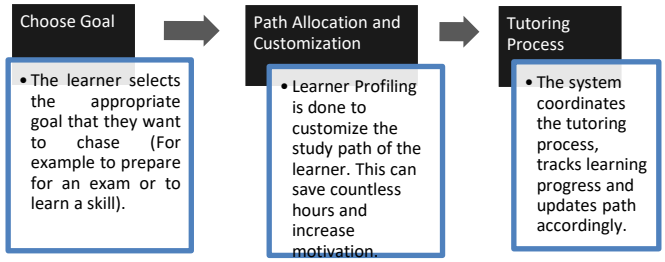

Fig. 5. Flow from Goal Selection to Tutoring.

B. *Extensibility*

The limited domain knowledge contained in a system can be extended from two ends. One of these is when the subject experts manually feed new domain knowledge into the system using the appropriate interface. This has a limitation – it is slow and it only expands the system in the direction which is determined by what the subject experts think should be and should not be a part of the system. Another way to truly steer the direction of expansion of the system towards what the learners of the system actually need is by allowing the learners to submit new questions.

In the proposed architecture, the new questions submitted by any learner are mapped on to the topics that currently exist in the system, and if there are no appropriate topics that fully relate to the questions, the architecture prompts the subject expert to create new topics that map with the question more appropriately. This leads to two outcomes, the first one being improvement and expansion of already existing topics, which is in turn going to help other learners as well. The second one being creation of new topics, which will eventually lead to generation of new courses, and new courses lead to new

subject paths that lead to new goals that increase the overall reach and usability of the system. Fig. 6 depicts the overall process of expansion.

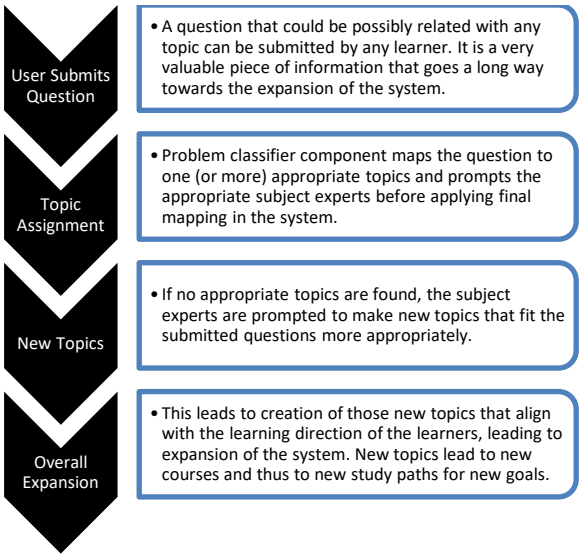

Fig. 6. Flow of System Expansion.

There is a mapping between questions and topics, that a question could be linked to one or more topics. The task of mapping a submitted question/problem to a topic is achieved using the problem classifier component of the system, which is an integral contributor towards the extensibility of the system.

1) Problem classifier: The ability of a learner to ask questions to a subject expert for an effective resolution is, at its core, one of the most fundamental activities involved in any learning process. In fact, a good question, combined with its solution will continue to help future learners who want to learn the subject, who can learn a lot by reading other people's questions and their solutions. For this, and many other reasons, in an e-learning environment that deals with questions, it becomes very helpful to classify the questions according to subject areas for an organized study. This problem of question classification falls under the umbrella of a domain of research called text classification.

Text classification deals with utilizing machine learning techniques to assign a class/label to any input text. It has been used widely to classify product reviews by corporations that want to understand the sentiment of their customers, or many specific cases such as to predict the ideological direction of court cases [24], or to classify fake news or hoax, which have become the most prevalent cybercrime in today's day and age that have immeasurable harms [25] or to identify a person's distinctive habits and behaviors through personality classification [26]. Various papers resolve the issue of programmed text classification proposing various strategies and arrangements. Extensive thorough reviews also exist that document text classification in detail [27-30].

When the data-set consists of user-submitted short questions with a vastly disproportionate number of questions in different subjects, the overall classification becomes a challenge. Researchers in [31] have proposed a general-purpose approach to assigning user-submitted questions their appropriate topic labels, without any extra vocabulary information related to the subjects. Several grid hyperparameter-searched iterations of Generalized Linear Model, Deep Learning (ANN), Gradient Boosting Model, Extreme Gradient Boosting Model (XGBoost), Distributed Random Forests and Extremely Randomized Trees were trained and evaluated, and it was found that out of these, XGBoost performs the best with a small and imbalanced dataset of very short text documents. To further improve the classification performance, a general-purpose approach to handle the unbalanced classes was used utilizing class weights.

C. Personalization

Personalization is the gist of intelligent tutoring. There is no point in calling a system intelligent unless it has some sort of personalization or adaptivity in it. In the context of ITSs, personalization and adaptivity can be practiced at various stages, but the most common one of these is the customization of the course a learner is taking according to the knowledge levels of the learner. This level of adaptivity is helpful at various aspects, one of which is it shortens the overall duration of the course. It is also helpful for uplifting the motivation of the learner to study because if the system presents those topics to the learner which they already know, it can lead to loss of interest, possibly leading to the learner leaving the course in between.

It is particularly challenging to implement course personalization in a system that features extensibility, as the system is going to require personalization in the courses of various varied domains that are going to be created automatically (or subject expert assisted) in the future and do not exist at the time the system was built.

1) Learner profiling: To achieve personalization and adaptivity in a system, sufficiently granularized domain knowledge is needed which can be broken-down or combined in multiple ways. This is best achieved when a limited piece of domain knowledge about a certain subject or a fixed number of subjects is structured into the domain model of the system in any rule-based analogy, as utilized by researchers in various domain-specific ITSs discussed in the previous sections. It is easy to form specific strategies to estimate the knowledge level of a specific subject, but to achieve learner profiling in a general-purpose system that supports all types of textual subjects is a challenge that researchers in the field of intelligent tutoring have been trying to solve since decades.

Researchers in [32] have proposed a learner profiling algorithm which is able to adapt any general-purpose course that the learner wants to take, to the knowledge level of the learner, in a very short time quantum, improving the motivation of the learner, shortening the total amount of time needed to complete the course tremendously and hence the overall learning process. The amount of time saved for the learner depends upon the number of topics detected to be known and the individual lengths of those topics. The algorithm requires topic inter-dependency information, which is typically provided by subject experts while drafting the

course. In addition to this, data of responses of past students that undertook the same course is used to improve the topic interdependency information – taking a safer approach for a general-purpose tutoring system when there is a possibility that the subject expert might not be able to map accurate relationships between topics perfectly. These two help the minimal learner profiling algorithm build the learner's knowledge profile as illustrated in Fig. 7 and it does that in the minimum amount of time possible.

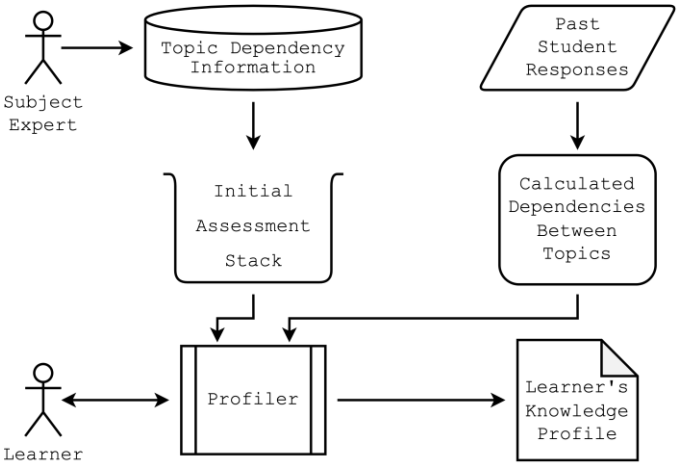

Fig. 7. The Proposed Model for Learner Profiling.

As there is no way to actually look inside the brain of the learner and check the neural connections to determine whether the learner knows the topic or not, the actual accuracy of predicting the knowledge level of the student with respect to a specific topic depends upon the quality of the assessment activity of the topic and the honesty of the learner's responses. If the quality of the assessment activities is assumed to be perfect the algorithm guarantees accurate adaptation of the course to the learner in the minimum possible time.

D. Course Improvements

A system that allows extensibility and teaches any types of courses to all types of learners without need of persistent intervention from a subject expert, it is necessary to have a mechanism that improves the domain knowledge and course information in the system. In the proposed architecture course improvement is achieved through two means as discussed in the following sections.

1) Automatic course improvement: The subject expert specifies the dependencies between topics and creates relationships mappings illustrating which topic depends upon which zero or more topics. These dependencies assist the overall instruction procedure for the learner, but the actual degree of dependencies between the topics is further refined using the data obtained from the initial learner profiling algorithm as covered in the work of [32]. This improves the overall mappings between topics, thus creating even better tutoring procedures and experiences for future students.

2) Collaborative course Improvement: This is achieved through manual intervention by the learners. The learner – the ultimate end-users of the system can also, collaboratively, seed course improvements for everyone. Since the expansion in the system happens dynamically, at any point can have gaps and/or faults in any grain of knowledge anywhere from a simple question to a topic to a course to an entire goal. The learners can report errors or gaps in any grain of the system, and it is prompted to the appropriate subject expert. Since every grain of knowledge is interconnected deeply with all the other parts of the system, improvements in any tiny fragment leads towards refinement of the entire system.

IV. Evaluation

For the evaluation of the effectiveness of the proposed framework, the components that make up the framework were evaluated thoroughly. The said evaluations have been stated in this section grouped under the respective modules they serve.

A. Expansion Module

In the implementation of the system for the evaluation of the module which is responsible for the ever-growing nature of the system, majorly because of the problem classifier component, a dataset of 6925 problems made up of short text documents from 13 different topics as illustrated in Fig. 8 was taken.

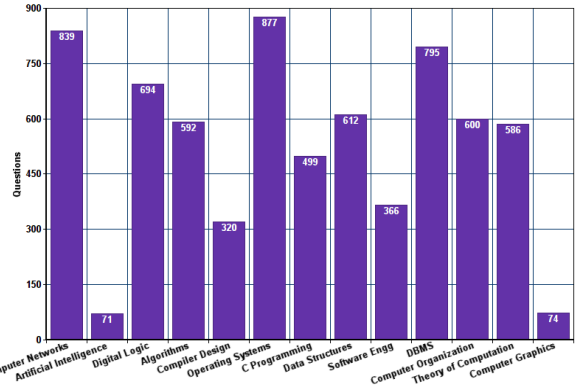

Fig. 8. Problem Dataset Distribution.

The system trained and evaluated around 20 industry standard classification algorithms on the dataset, the four best performing ones of which are shown in Fig. 9 with their per-class error distributions.

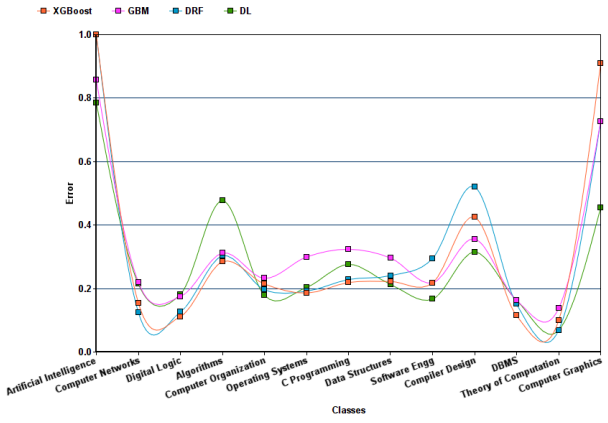

Fig. 9. Per Class Error.

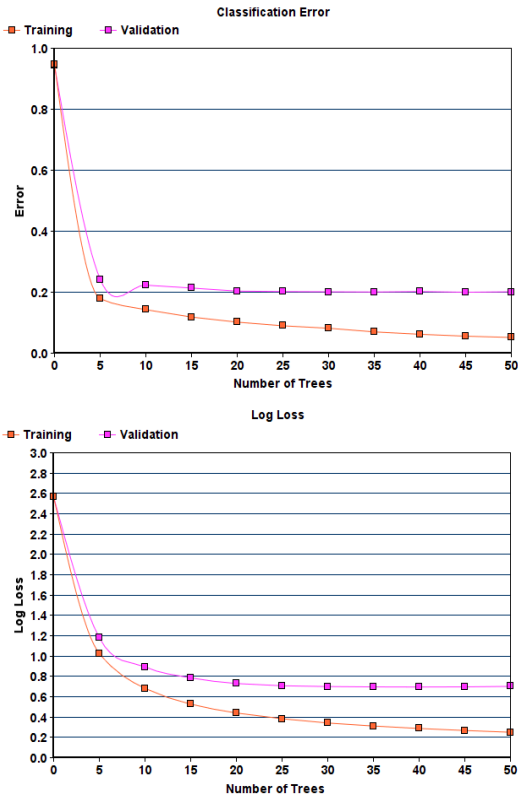

Fig. 10. Classification Error and Log Loss of Question Classifier.

Out of these algorithms XGBoost had the minimum per class error and was chosen as the algorithm of choice for the problem classifier component in the implementation of the system. The classification error and log loss of the algorithm are illustrated in Fig. 10. The figure illustrates the changes in classification error and log loss with respect to the number of trees in the iteration of the algorithm for training as well as validation data.

This problem of imbalanced classes refers to the situation that all the subjects have not been represented equally. More precisely, there is a huge disproportionality between classes, with certain subjects having more than 800 samples and some not even having 80 samples. The problem of imbalanced classes has been greatly explored in excellent reviews in [33], [34]. Class weights were calculated from the dataset distribution frequency and the model was retrained with the best hyperparameters found by the previous search. As a result, the F1 score as well as precision and recall values improved as seen in Table I.

In the quest of proposing a general-purpose approach to assigning user-submitted questions their appropriate subject labels, without any extra vocabulary information related to the subjects, several grid hyperparameter-searched iterations of Generalized Linear Model, Deep Learning (ANN), Gradient Boosting Model, Extreme Gradient Boosting Model (XGBoost), Distributed Random Forests and Extremely Randomized Trees were trained and evaluated, and it was found that out of these, XGBoost performs the best with a small and imbalanced dataset of very short text documents.

TABLE I. CLASSIFICATION REPORT FOR XGBOOST

Class	After Class Weights			Support
	Precision	Recall	F1-Score	
Artificial Intelligence	0.29	0.14	0.19	14
Computer Networks	0.87	0.83	0.85	168
Digital Logic	0.79	0.86	0.83	143
Algorithms	0.72	0.72	0.72	109
Computer Organization	0.73	0.79	0.76	112
Operating Systems	0.78	0.75	0.76	167
C Programming	0.79	0.71	0.75	105
Data Structures	0.69	0.8	0.74	108
Software Engineering	0.87	0.79	0.83	78
Compiler Design	0.74	0.73	0.73	73
DBMS	0.86	0.87	0.86	166
Theory of Computation	0.89	0.88	0.88	131
Computer Graphics	0.13	0.18	0.15	11

To further improve the classification performance, a general-purpose approach to handle the unbalanced classes was used utilizing class weights. The overall average performance of XGBoost on the validation data has been shown in Table II.

The workflow also same training can be repeated when implementing the system for any other subject domain. Through the ability of problem classification, the system is able to classify new and unseen learner submitted problem to their appropriate topics, and if there aren't any appropriate topics, the system allows the subject experts on the backend to create new more appropriate topics for the problem at hand, resulting in the expansion of the system.

B. Pedagogy Module and Student Module

Based on the topic relationships illustrated in Fig. 3, learner response data was collected from 60 learners assessed over 10 topics and using the algorithm proposed in [32] the precision of the relative prediction of knowledge of other topics when knowledge of one topic is given was calculated as illustrated in Table III. The intensity of the shade in each cell denotes the magnitude of the precision.

TABLE II. PERFORMANCE METRICS OF XGBOOST ON THE CLASSIFICATION OF VALIDATION DATA

	Before Class Weights Adjustment			After Class Weights Adjustment			Support
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	
Macro Average	0.69	0.68	0.68	0.7	0.7	0.7	1385
Weighted Average	0.79	0.8	0.79	0.79	0.79	0.79	1385

TABLE III. PREDICTION PRECISION TABLE

	Given A1	Given A2	Given A3	Given A4	Given A5	Given A6	Given A7	Given A8	Given A9	Given A10
A1		0.457143	0.444444	0.512821	0.444444	0.377778	0.551724	0.791667	0.666667	0.542857
A2	0.695652		0.666667	0.589744	0.666667	0.644444	0.896552	0.791667	0.809524	0.742857
A3	0.173913	0.171429		0.102564	0.777778	0.133333	0.206897	0.250000	0.238095	0.114286
A4	0.869565	0.657143	0.444444		0.444444	0.688889	0.620690	0.833333	0.857143	0.857143
A5	0.173913	0.171429	0.777778	0.102564		0.111111	0.206897	0.250000	0.238095	0.114286
A6	0.739130	0.828571	0.666667	0.794872	0.555556		0.862069	0.833333	0.809524	0.885714
A7	0.695652	0.742857	0.666667	0.461538	0.666667	0.555556		0.833333	0.523810	0.628571
A8	0.826087	0.542857	0.666667	0.512821	0.666667	0.444444	0.689655		0.714286	0.571429
A9	0.608696	0.485714	0.555556	0.461538	0.555556	0.377778	0.379310	0.625000		0.485714
A10	0.826087	0.742857	0.444444	0.769231	0.444444	0.688889	0.758621	0.833333	0.809524	

The improvements in the course suggested by the course corrector component through the use of the prediction precision table are illustrated in Fig. 11. After manually choosing a threshold value of 0.8 and combining it with the prediction precision information calculated before (as shown in Table III), the course corrector component predicted the relative dependencies between topics, mapped over the topic relationships specified in Fig. 3. In the figure, the dotted lines show new suggested dependencies and the numbers written on the arrows suggest the predicted strength of the dependency. This way the component can be used for learner-response-data-driven course improvements.

The learner profiling algorithm uses the topic dependencies fed in by the subject experts, and possibly improved by the course corrector component, and combine it with the past learner response data to achieve learner profiling with the minimum possible number of assessment activities based on a controllable parameter that has been called the trust threshold. Fig. 12 illustrates the various assessment sequences generated by the algorithm with respect to the value of the trust threshold parameter.

So, through the use of a non-subject-specific arbitrary topic relationship structure and hierarchy and past learner response data, the system is able to generate assessment sequences of varying lengths and complexities. This show that the framework can perform course customization through learner profiling on any course that is made up of a sequence of inter-dependent topics, be it of any subject.

C. Domain Module and Interface Module

The domain model realized the entire course, topic, goal and exercise information the system revolves around as entities and the relationships mapped among them. In the current implementation of the system the domain model was implemented using the popular open-source object-relational database system PostgreSQL. The following describe the highlights of the implementation of the architecture of the domain module in the system:

- It realizes the building blocks that the framework is based around adequately.
- It can represent all expected data over time.
- The model ensures that it avoids repetitive storage of the same information.
- The model ensures the maintenance of data integrity over time.
- It is clean, consistent and easy to understand.
- The model provides efficient access to data.

The interface module is able to successfully control access to the data is through the use of token-based authentication mechanisms. The module, apart from giving access to appropriate interfaces to appropriate users and learners, ensures the proper functioning of various other modules through various integrity checks in place. For illustration, in the creation of courses that do not possess cyclic dependencies between topics, because a topic A requiring prior knowledge of a topic B while simultaneously topic B requires prior knowledge of topic A is a problem, and it can inhibit the proper functioning of the pedagogy and student module. The rest of the evaluation of the interface module has been avoided for the sake of brevity.

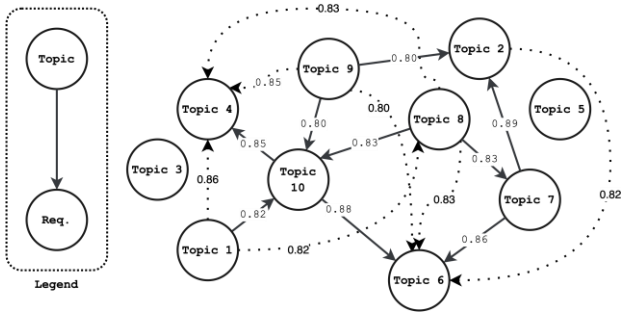

Fig. 11. Topic Relationships Calculated by Course Corrector Component.

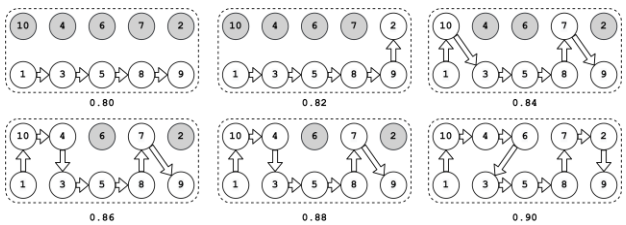

Fig. 12. Different Learner Assessment Sequences According to Trust Threshold.

V. Conclusion

In this paper, an architecture for a domain-independent intelligent tutoring system that features extensibility, personalization and automatic course improvements has been described. It features an interface module that provides separate interfaces for learners and subject experts. The architecture allows expansion of the system without requiring persistent intervention or any knowledge of building ITSs from the subject experts and portable (mobile) interfaces for learners for better learning, motivation and engagement. This provides countless avenues for cost-effective tutoring.

The proposals for future work would be to add support in the architecture that allows the ability of creation of new portable modules that could be built by anyone using a possibly defined format, and added to the system to incorporate various features into any part of the system. An example would be a module that replaces the pedagogy algorithms, or a module that utilizes neural networks for a part of the system, etc. Also proposed is the ability to integrate with other apps, since this architecture has an interface module that works through APIs, the possibility of what can be achieved after integration with other apps and platforms would be exciting to explore.

References

[1] Januszewski and M. Molenda, Educational technology : a definition with commentary. Association for Educational Communications and Technology, 2008.

[2] V. J. Shute and J. Psotka, “Intelligent Tutoring Systems: Past, Present, and Future.,” Armstrong Lab Brooks AFB TX Human Resources Directorate, 1994.

[3] C. Dede and K. Swigger, “The evolution of instructional design principles for intelligent computer-assisted instruction,” J. Instr. Dev., pp. 15–22, 1988.

[4] J. R. Carbonell, “AI in CAI: An Artificial-Intelligence Approach to Computer-Assisted Instruction,” IEEE Trans. Man-Mach. Syst., vol. 11, no. 4, pp. 190–202, Dec. 1970, doi: 10.1109/TMMS.1970.299942.

[5] J. H. Larkin and R. W. Chabay, “Computer-assisted instruction and intelligent tutoring systems - shared goals and complementary approaches,” undefined, 1992.

[6] M. Elsom-Cook, O. University, and M. K. (GB) C. A. L. R. G. O. University, Intelligent Computer-aided Instruction Research at the Open University. Open University, 1987. [Online]. Available: https://books.google.co.in/books?id=ptCgHAAACAAJ.

[7] J. W. Schofield, R. Eurich-Fulcer, and C. L. Britt, “Teachers, computer tutors, and teaching: The artificially intelligent tutor as an agent for classroom change,” Am. Educ. Res. J., vol. 31, no. 3, pp. 579–607, 1994.

[8] W. L. Miller, R. S. Baker, M. J. Labrum, K. Petsche, Y.-H. Liu, and A. Z. Wagner, “Automated detection of proactive remediation by teachers in Reasoning Mind classrooms,” in Proceedings of the Fifth International Conference on Learning Analytics and Knowledge, 2015, pp. 290–294.

[9] D. Diziol, E. Walker, N. Rummel, and K. R. Koedinger, “Using intelligent tutor technology to implement adaptive support for student collaboration,” Educ. Psychol. Rev., vol. 22, no. 1, pp. 89–102, 2010.

[10] A. C. Graesser, K. VanLehn, C. P. Rosé, P. W. Jordan, and D. Harter, “Intelligent tutoring systems with conversational dialogue,” AI Mag., vol. 22, no. 4, pp. 39–39, 2001.

[11] K. R. Koedinger, A. Corbett, and others, Cognitive tutors: Technology bringing learning sciences to the classroom. na, 2006.

[12] H. Crompton, “A historical overview of mobile learning: Toward learner-centered education.,” Handb. Mob. Learn., no. August 2013, pp. 3–14, 2013.

[13] G. Trentin and M. Repetto, Using Network and Mobile Technology to Bridge Formal and Informal Learning. Chandos Publishing, 2013. doi: 10.1533/9781780633626.

[14] N. Bukharaev and A. W. Altaher, “Mobile Learning Education has Become More Accessible,” Am. J. Comput. Sci. Inf. Technol., vol. 5, no. 2, Oct. 2017, doi: 10.21767/2349-3917.100005.

[15] R. Freedman, S. S. Ali, and S. McRoy, “What is an Intelligent Tutoring System?,” Int. J. Artif. Intell. Educ., vol. 11, no. 3, pp. 15–16, Sep. 2000, doi: 10.1145/350752.350756.

[16] R. Nkambou, R. Mizoguchi, and J. Bourdeau, Eds., Advances in Intelligent Tutoring Systems. Berlin Heidelberg: Springer-Verlag, 2010. Accessed: Aug. 08, 2019. [Online]. Available: https://www.springer.com/gp/book/9783642143625.

[17] H. S. Nwana, “Intelligent tutoring systems: an overview,” Artif. Intell. Rev., vol. 4, no. 4, pp. 251–277, Dec. 1990, doi: 10.1007/BF00168958.

[18] S. J. Derry, L. W. Hawkes, and U. Ziegler, “A plan-based opportunistic architecture for intelligent tutoring,” Proc. Intell. Tutoring Syst. ITS-88, pp. 116–123, 1988.

[19] J. Siemer and M. C. Angelides, “A comprehensive method for the evaluation of complete intelligent tutoring systems,” Decis. Support Syst., vol. 22, no. 1, pp. 85–102, 1998.

[20] C. Dede, “A review and synthesis of recent research in intelligent computer-assisted instruction,” Int. J. Man-Mach. Stud., vol. 24, no. 4, pp. 329–353, 1986.

[21] E. de Barros Costa and A. Perkusich, “Modeling the cooperative interactions in a teaching/learning situation,” in International Conference on Intelligent Tutoring Systems, 1996, pp. 168–176.

[22] T. E. Turner, M. A. Macasek, G. Nuzzo-Jones, N. T. Heffernan, and K. R. Koedinger, “The Assistment Builder: A Rapid Development Tool for ITS.,” in AIED, 2005, pp. 929–931.

[23] A. C. Graesser et al., “AutoTutor: A tutor with dialogue in natural language,” Behav. Res. Methods Instrum. Comput., vol. 36, no. 2, pp. 180–192, 2004.

[24] C. I. Hausladen, M. H. Schubert, and E. Ash, “Text classification of ideological direction in judicial opinions,” Int. Rev. Law Econ., vol. 62, p. 105903, Jun. 2020, doi: 10.1016/J.IRLE.2020.105903.

[25] J. P. Haumahu, S. D. H. Permana, and Y. Yaddarabullah, “Fake news classification for Indonesian news using Extreme Gradient Boosting (XGBoost),” IOP Conf. Ser. Mater. Sci. Eng., vol. 1098, no. 5, p. 052081, Mar. 2021, doi: 10.1088/1757-899X/1098/5/052081.

[26] A. S. Khan, H. Ahmad, M. Z. Asghar, F. K. Saddozai, A. Arif, and H. A. Khalid, “Personality Classification from Online Text using Machine Learning Approach,” Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 3, 2020, doi: 10.14569/IJACSA.2020.0110358.

[27] T. Joachims, Learning to classify text using support vector machines, vol. 668. Springer Science & Business Media, 2002.

[28] SebastianiFabrizio, “Machine learning in automated text categorization,” ACM Comput. Surv. CSUR, vol. 34, no. 1, pp. 1–47, Mar. 2002, doi: 10.1145/505282.505283.

[29] C. C. Aggarwal and C. Zhai, Mining text data. Springer Science & Business Media, 2012.

[30] A. Chaturvedi, S. Yadav, M. A. M. H. Ansari, and M. Kanojia, “Comparative Multinomial Text Classification Analysis of Naïve Bayes and XGBoost with SMOTE on Imbalanced Dataset,” pp. 339–349, 2021, doi: 10.1007/978-981-16-2543-5_29.

[31] S. Singh and V. Singh, “Mapping User-submitted Short Text Questions to Subjects of Study: A Multinomial Classification Approach,” presented at the 3rd International Conference on Communication and Intelligent Systems (ICCIS 21), National Institute of Technology, Delhi, Dec. 2021.

[32] S. Singh and V. Singh, “A Graph Based Approach to Learner Profiling in an Intelligent Tutoring System,” Indian J. Comput. Sci. Eng., to be published.

[33] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," GESTS Int. Trans. Comput. Sci. Eng., vol. 30, no. 1, pp. 25–36, 2006.

[34] S. Datta and A. Arputharaj, "An Analysis of Several Machine Learning Algorithms for Imbalanced Classes," in 2018 5th International Conference on Soft Computing Machine Intelligence (ISCMI), 2018, pp. 22–27. doi: 10.1109/ISCMI.2018.8703244.