A Template-Based Spaced Repetition Learning Solution Designed for Educators

Ayman Hajja
Department of Computer Science
College of Charleston
Charleston, SC, USA
hajjaa@cofc.edu

Austin J. Hunt
Department of Computer Science
College of Charleston
Charleston, SC, USA
huntaj@g.cofc.edu

***Abstract*—This Innovative Practice Work in Progress paper introduces a freely available API-based platform for spaced repetition education. Spaced repetition, or spaced repetition learning, is a technique used by learners to improve long-term knowledge retention through repeated exposure to information spread out through time, traditionally in the form of flashcard questions.**

The concept of repeated exposure to information at varying lengths of time, and its effectiveness to improve human memory, has been evolving since first being investigated in the 19th century. Recently, new mobile and web learning applications have been employing spaced repetition algorithms and techniques to improve students' retention; although increasingly popular amongst students, we believe there are certain drawbacks to these systems that we set out to address in this work.

First, existing solutions primarily target only students as users without providing instructor-focused functionality for supplementing their teaching; we explore, through the inclusion of instructor-focused interfaces and reporting mechanisms, potential insights to be gained about the effect of spaced repetition learning on student retention. Secondly, to the best of our knowledge, current popular spaced repetition algorithms are purely question-based; that is, re-exposure delays are calculated only per question, and not per topic. To address this, we present a novel question-templating framework that 1) enables knowledge retention to be assessed and analyzed in terms of groups of questions representing topics instead of only specific questions, and 2) provides a tool for instructors to efficiently manage learning material by creating randomizable templates to feed on the fly generation of similar-but-unique question sets for each student. Third, existing solutions widely employ spaced repetition to aid memorization without leveraging it to aid generalization; we aim to highlight a potential for the spaced repetition model to promote retention of generalized skills (transferable among different challenges). Lastly, and most importantly, we seek to provide a free application programming interface (API) that can be integrated with any custom mobile or web user interface to provide the research community with a highly integrable toolkit for exploring new questions about spaced repetition learning.

***Index Terms*—spaced repetition, optimal re-exposure delay, cloud services, pluggable API**

I. Introduction

A. Online Education and Changing Philosophies

The rapidly growing role of online education platforms in a landscape of internet-driven globalization and information access [1], accelerated by the 2020 onset of the COVID-19 pandemic and its necessitation of online learning for public health purposes [2] [3] [4], carries with it a shift in education philosophy defined by heightened focus on the learner, entailing greater attention to personalization and enhancement of their active learning experience. [5]

B. Spaced Repetition Learning in an Online World

With our project, we approach this digital-age, learner-centric education philosophy in the domain of spaced repetition learning, a methodology resting on psychologist Hermann Ebbinghaus's 19th century discovery of the *spacing effect* [7] which details the superiority of spaced learning events to massed, or "crammed", learning events for promoting long-term learning – that is, especially in the context of long-term memory, it is much less effective to cram learning events (e.g. flashcard exposure) in immediate succession over one long sitting than it is to study the same material for short periods over a longer time interval. Using online learning platforms designed for spaced repetition learning allows the capturing of data around this time interval – which we refer to in this paper as the *re-exposure delay*, abbreviated *RD* – especially as it relates to subject material. As seen with Settles' and Meeder's application of their half-life regression model of spaced repetition to Duolingo learning [8], this data can offer actionable insight about optimal times to deliver (and re-deliver) practice material, both 1) for a particular learner, which promotes personalization, and 2) for a particular subject area if patterns arise across many learners for that subject area, which can improve the effectiveness of overall course content delivery.

C. The Gap in Existing Solutions

There are many existing online learning environments that offer tools for spaced repetition practice, but as discussed by Vlach and Sandhofer [9] who explored the multi-tiered benefits of spaced repetition as an alternative to massed learning, existing spaced repetition tools primarily employ it as a method of memorization but tend not to employ it as a method of learning generalized (transferable) skills, even though empirically, the learning method promotes both types of learning. Moreover, while popular existing tools for spaced repetition learning (which largely include flashcard applications such as Quizlet [10] and language acquisition applications like

Duolingo [8], which again are prime tools for *memorization*) offer personalized and customizable experiences for learners, they are not designed to be used by instructors for facilitating learning. Enabling instructor facilitation in a modern spaced repetition learning platform enables the instructor to 1) promote active engagement and critical thinking through interactivity and intentionally designed courses, as outlined by Mandernach et. al [11], and 2) gain insight via reporting mechanisms about student retention patterns to understand how to improve future delivery of their course material. Lastly, existing popular tools for spaced repetition, to the best of our knowledge, only focus on Ebbinghaus's aforementioned "spacing effect" without consideration for the complementary "lag effect" that was later introduced by Arthur Melton [12] which describes the value of increasing the RD over time instead of keeping it constant; existing tools do not leverage the proposed value of gradually increasing the RD and instead primarily allow the learner to determine the delay themselves, which most generally is one day (illustrated, for example, by Duolingo's "day streak" feature, which resets to 0 if a learner skips a day).

D. How We Address the Gap

With our project, we aim to contribute to the online spaced repetition learning ecosystem with a scalable, integrable, cloud-service-based API that 1) allows for instructor facilitation of learning and 2) extends the spaced repetition methodology beyond rote memorization into more complex generalization. We do this in a practical way that uses a novel question templating framework available to instructors, enabling them to create courses composed not of predefined, static question sets shared by every student, but question templates that drive dynamic generation of similar-but-different questions for desired subject areas, which render differently on the fly for each student as discussed in Subsection IV-B. Moreover, as will be discussed later in Subsection IV-A, we explore the "lag effect" mentioned previously by using answer submission data to dynamically and continuously refine the RD (instead of keeping it constant) for each student-topic pairing (e.g. student A will be asked about topic B after RD days, where RD is iteratively refined after each response).

II. Personalized Learning

At the core of our system lies the constructivist idea of the learner as an active agent in their education and the instructor as a facilitator of that active agency. With our system, an instructor can supplement their teaching through question-template-driven courses that they feel will best promote their students' understanding of subject material. Each of their students can then actively subscribe to a created course, and can practice unique, quasi-random instances of the defined question templates via spaced repetition, where both the content delivered to them and the rate at which it is re-delivered (the RD) is personalized and refined through calculations driven by their own decisions, all while within the intentional, planned bounds defined by the facilitator. At each point of re-delivery, the student gets a notification that the respective course has content ready for practice.

III. System Architecture

With a goal of deploying a highly available, highly scalable API for use by any number of spaced repetition learning applications, we decided to implement using the Amazon API Gateway and AWS Lambda. With this architecture, we are able to easily create usage plans and API keys that enable authorized, secure requests from registered external applications, while also setting throttling and quota limits on those usage plans to keep accountability high and costs low. We're also able to run the core business logic of the API using serverless, parallel computation with minimal administration that scales precisely with the computation requests. While these benefits make AWS a great tool for the project long-term, using this cloud architecture does carry certain challenges, particularly around its complexity compared to simpler architectures. For example, we considered an API built solely on the Django REST Framework running on a web server; this would have definitely simplified both the development and deployment of the API, but this would not offer the desired scalability or availability guarantees that AWS offers, nor would it provide the same easy API usage plan configurations.

The following two sections outline the two main components of our project: 1) a pluggable API and 2) SpaceMan, a central API management app.

A. A Pluggable API for Spaced Repetition

Using the AWS Serverless Application Model (SAM), an open-source framework for using YAML to easily, declaratively build entire stacks of cloud services, we created an AWS CloudFormation Stack composed of:

- A set of independent AWS Lambda functions for performing the core business logic of our spaced repetition system, including 1) student and instructor account creation, 2) course creation and editing for instructors, 3) question template creation and editing for instructors, 4) course listing and subscriptions for students, 5) practice notifications for students (for practice content delivery notifications), 6) quasi-random instance rendering of question templates (as discussed in Subsection IV-B), 7) answer submission handling and RD adjustments, and 8) reporting for instructors.
- A secure API Gateway allowing abstracted interaction with the Lambda functions, requiring each request to include an allowed API key (where each registered application has two allowed API keys, discussed below)

B. A Centralized Management Application

The second part of our system is a central web application where interested users can register their own custom spaced learning application to use our spaced repetition API. Upon registering successfully (which requires approval), the newly registered app is assigned two API keys, one for student-related API requests, the other for instructor-related API

requests. These API keys are in turn added to a Usage Plan for the API Gateway mentioned in Subsection III-A. We aim to include details on proper usage of these API keys in our API documentation before the system is fully deployed.

IV. Reinventing spaced repetition Systems

It would be fair to state that most, if not all, spaced repetition systems are designed primarily for learners, although based on the literature review that we conducted, there is little to no scientific research on the use of spaced repetition systems to gain insights on student learning. We believe that the data collected from such systems can provide invaluable, actionable insights on knowledge retention among students, which instructors could use to enhance material delivery, thus optimizing aspects for both learning and teaching. In this section, we will introduce a few novel ways of aggregating the data collected from students' answers in an intentional way with respect to data collection and insight extraction.

A. Re-exposure Delay Factor and Decay Rate

In Subsection I-B, we mentioned the term *re-exposure delay* (*RD*), which can be defined as the duration (in days) after which a particular question will be re-exposed to a student based on an earlier answer by the same student to the same question. For example, we can define the RD to be 7 for a correct answer, and 1 for an incorrect answer, so if a student answers a question correctly, they will be prompted with the same question after one week, and if they answer it incorrectly, they will be prompted with the same question the following day. We can alternatively define the RD using a *factor* instead of a constant. For example, an RD factor of 2 for a correct answer means that the RD (in days) will be 2 multiplied by the last RD for the same question. For questions answered incorrectly, the factor would typically be less than 1.

Before we introduce new ways of formulating a spaced repetition learning system, it would also be helpful to introduce a new term: *decay rate*, which to the best of our knowledge, has not been used in this context before. The knowledge decay rate (or *KDR*) can be defined as the rate at which a student forgets the material needed to answer a question correctly; this value can be mathematically defined as the reciprocal of the RD because when the RD is set to be some number of days, we are establishing an assumption that after that number of days, the student will need to be reminded of that material. This is a perhaps different, yet still valid (and rather necessary for the template-based approach presented in the next subsection), way of interpreting these RDs.

For the sake of simplicity, we would like to begin the discussion by assuming that the KDR is constant, meaning that if a student is going to forget the material for a specific topic in x number of days, that student will forget the same amount of material everyday; and while this assumption might not be true, we believe that it is still a reasonable first hypothesis to test. Table I shows a simple example of RDs and KDRs based on an RD factor of 2 for a correct answer, and an RD of .25 for an incorrect answer.

As can be seen in the Table I, when we have a new RD of 2, that automatically implies that the KDR is .5, which means that each day the student will forget (or almost forget) half the knowledge amount needed to answer the question; hence, the system should prompt the student with the same question after 2 days. The introduction of KDR might not seem necessary at first, but once we introduce the template-based approach, it will be clear to the reader that such re-framing is rather necessary to achieve optimal results.

	Answer	**Old RD**	**New RD**	**Old KDR**	**New KDR**
Day 1	Correct	1	2	1	.5
Day 3	Correct	2	4	.5	.25
Day 7	Incorrect	4	1	.25	1
Day 8	Correct	1	2	1	.5

TABLE I
An example of how RDs and KDRs change over time

B. Question-based vs Template-based

While there are different ways of calculating RDs (as defined in Subsection IV-A), existing spaced repetition platforms employ a single-question-based approach, which means that an answer provided by a student to a particular question will determine the RD for that particular question only. Calculating accurate RDs using this single-question-based approach is rather convenient, which justifies why this approach has been adopted by most, if not all, platforms. From the point of view of an instructor interested in finding student learning patterns, this question-based approach is unfortunately not very helpful; treating students' answers and RDs in such a distributive manner does not help with finding patterns in data.

One must introduce a meaningful and integrated way to aggregate the data collected to facilitate the process of insight gathering. Our first novel proposition is to use a template-based approach as opposed to a question-based approach. The specific template-based approach that we will adopt was introduced by Hajja et. al in [13], which allows instructors to create question templates with high variability in a convenient and flexible way. This was done by separating the portion of the logic responsible for element randomization from the portion defining the question's essential structure. These two portions, respectively referred to as the *pre-code* and the *skeleton*, are both input by the instructor; together, they comprise a single question template, from which many unique *instances* are generated. The pre-code input, which is written in Python, allows for random-value–assignment to variables, which are subsequently embedded into the skeleton. As mentioned in Subsection III-A, one API endpoint is responsible for the question template creation and editing for instructors; Fig 1 shows a screenshot of a web interface that invokes that API endpoint. Fig 2 shows the student web interface after submitting an answer; this interface uses two API endpoints (quasi-random instance rendering of question templates and answer submission handling and RD adjustments).

To make things a little more clear, let us think of an example. Assume that an instructor decides to create a single

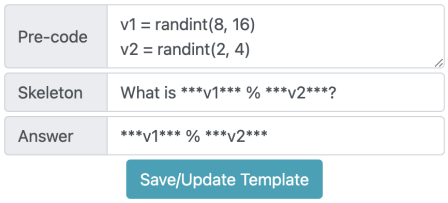

Fig. 1. Instructor template creation web interface

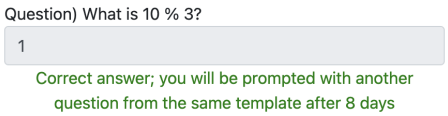

Fig. 2. Student web interface after submitting a correct answer

template assessing students' knowledge on converting from decimal to binary, and based on the question template, the instructor might decide that 10 (let us refer to this number by *count*) questions should roughly cover the entire range of various question instances that could be generated from that template; we will refer to these questions instances by i0, i1, ..., i9. Also, let us assume that the initial RD is 1 day, the new RD factor for a correct answer is defined as 2, and the RD constant for an incorrect answer is defined as 1.

There are two ways of integrating this template-based approach in a spaced repetition system. The first approach is to treat these question instances independently with regards to calculating the RDs. In Table II below, we show an example of the current state of these question instances for a hypothetical student at some point in time.

	Answer	**Old RD**	**New RD**
i0, i1, and i2	Correct	4	8
i3 and i4	Incorrect	2	1
i5, i6, and i7	Incorrect	4	1
i8 and i9	Correct	8	16

TABLE II
AN EXAMPLE OF TREATING INSTANCES INDEPENDENTLY

Since the student answered question instances i0, i1, and i2 correctly, 3 new (and randomly generated) questions will appear after 8 days from the same template, and since the student answered question instances i8 and i9 correctly, he will be prompted with 2 new questions after 16 days. The obvious drawback for calculating the RDs for questions independently (as shown above), is the fact that students will be "re-exposed" to these new question instances at uneven delays. Again, if we assume that these RDs imply the time it would take for a student to forget the knowledge needed to answer a question, and since all these questions are derived from a single template (hence a single topic), it would make much more sense to distribute these questions evenly throughout that time.

The second approach would be to use a single RD for all question instances derived from a single question template; that way, we can distribute the RDs for a group of question instances evenly, and in a way that aligns with the concept of template-based questions. Also, for us to be able to do that in a convenient and measurable way, we would need to start using the KDR instead of the RD. Let us provide another example to demonstrate this approach of calculating the KDR for an entire template (representing a topic). We will assume that the initial KDR for every question instance will be defined as 1 at the beginning of the student assessment, which simply means that we would like to ask all 10 (question template count) questions at the beginning of the assessment. The KDR for a question template will be defined as the sum of all KDRs for the question instances generated from that template. If the sum of the KDRs (let us refer to this number by *total decay*) is less than one, that means we need to prompt the student with a random instance after 1 divided by the *total decay* number of days; for example, .2 total decay means that we should prompt a student with a random instance after 5 days. If the *total decay* is more than 1 on the other hand, then that would mean that we need to prompt the student with that number of questions the following day.

Using the *total decay* to determine the next question re-exposure results in a more evenly distributed question instances. One simple way to demonstrate the desired effect that results from using the *total decay* would be to assume that a student is going to answer all question instances (generated from a single template) correctly for a number of times, as shown in Table III. Keep in mind that the total decay is the sum of the KDRs for the last 10 (template count) question instances generated from that template.

	Instances Asked	**Total Decay Rate**
Day 1	10	$5(10 * .5)$
Day 2	5	$3.75(5 * .5^2 + 5 * .5)$
Day 3	round(3.75) = 4	$2.25(4 * .5^3 * 5 * .5^2 + 1 * .5)$
Day 4	round(2.25) = 2	$1.63(2 * .5^4 + 4 * .5^3 + 4 * .5^2)$
Day 5	round(1.63) = 2	$1.44(2 * .5^5 + 2 * .5^4 + 2 * .5^3 + 4 * .5^2)$

TABLE III
DISTRIBUTION OF INSTANCES ASKED USING TOTAL DECAY

As can be seen in the table above, the distribution of the number of instances asked is reasonably distributed over time, as opposed to asking the entire set of questions (10) from the template in powers of two increments (2 days, 4 days, 8 days, ...), which would have been the case if we were to treat these questions independently with regards to calculating their RDs.

V. CONCLUSIONS AND EXPECTED RESULTS

With this project, the goal in sharpest focus is to introduce a freely available API for education researchers to interact with a toolkit of abstracted, consistent, integrable business logic allowing for expanded research and increased focus on the spaced repetition learning model – especially as it relates to the acquisition and retention of generalized, transferable knowledge and skills, as well as to the constructivist principles of learning enhancement and personalization promoted by an increasingly digital age of education, which has surged

due to the 2020 global pandemic and the public-health-driven social distancing it requires. This goal comprises two sub-goals corresponding, respectively, to the perspectives of students and instructors. On the student side, we aim to collect data from students who use the system for spaced repetition learning to understand the degrees to which they a) feel their learning experiences are personalized by the system, and more generally, b) feel the system had a positive effect on their learning. On the instructor side, we plan to survey instructors who use the system to understand to what extent the system helped them to 1) efficiently manage content delivery through question templating, 2) identify optimal RDs for different topics (via question templates) within their material, and 3) find actionable insights for improving future course content delivery. Upon official deployment, we plan to build and register our own sample application for API usage to use as a supplemental teaching tool with a computer architecture course; we will use the API from this application (as anyone would) to provide students with an interface for learning course topics in a spaced repetition format, and we will use the reporting component of the API to analyze learning patterns and retention rates among the students in the course. From this planned deployment and sample application, we expect to gain valuable and actionable insight about optimal delivery of material for this computer architecture course.

REFERENCES

[1] Allen, I. E., & Seaman, J. (2008). Staying the course: Online education in the United States, 2008. Sloan Consortium. PO Box 1238, Newburyport, MA 01950.

[2] Sun, L., Tang, Y., & Zuo, W. (2020). Coronavirus pushes education online. Nature Materials, 19(6), 687-687.

[3] Dhawan, S. (2020). Online learning: A panacea in the time of COVID-19 crisis. Journal of Educational Technology Systems, 49(1), 5-22.

[4] Daniel, J. (2020). Education and the COVID-19 pandemic. Prospects, 49(1), 91-96.

[5] Zhuoxuan, J., Yan, Z., & Xiaoming, L. (2015). Learning behavior analysis and prediction based on MOOC data. Journal of computer research and development, 52(3), 614.

[6] Kiryakova, G. (2017). Application of Cloud services in education. Trakia Journal of Sciences, 15(4), 277-284.

[7] Ebbinghaus, H. (2013). Memory: A contribution to experimental psychology. Annals of neurosciences, 20(4), 155.

[8] Settles, B., & Meeder, B. (2016, August). A trainable spaced repetition model for language learning. In Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers) (pp. 1848-1858).

[9] Vlach, H. A., & Sandhofer, C. M. (2012). Distributing learning over time: The spacing effect in children's acquisition and generalization of science concepts. Child development, 83(4), 1137-1144.

[10] Barr, B. (2016). Checking the effectiveness of Quizlet® as a tool for vocabulary learning (Doctoral dissertation).

[11] Mandernach, B. J., Forrest, K. D., Babutzke, J. L., & Manker, L. R. (2009). The role of instructor interactivity in promoting critical thinking in online and face-to-face classrooms. MERLOT Journal of online Learning and Teaching, 5(1), 49-62.

[12] Melton, A. W. (1967). Repetition and retrieval from memory. Science, 158(3800), 532-532.

[13] Hajja, A., Hunt, A. J., & McCauley, R. (2019, October). PolyPy: A Web-Platform for Generating Quasi-Random Python Code and Gaining Insights on Student Learning. In 2019 IEEE Frontiers in Education Conference (FIE) (pp. 1-8). IEEE.