Designing a Rapid Adaptive Content Registry (RACR) for Adaptive Learning

Benjamin D. Nye, Aditya Jain, Dilan R. Ramirez, Mark G. Core, William Swartout
University of Southern California, Institute for Creative Technologies
Playa Vista, CA, USA
nye@ict.usc.edu; jainadit@usc.edu; {dramirez, auerbach, core, swartout}@ict.usc.edu

ABSTRACT

Despite meta-analyses showing strong learning gains for adaptive learning, few domain areas are covered by adaptive learning. A key reason for this is a content bottleneck: currently, adaptive systems require highly-trained computer scientists and educational specialists to add new content. To explore this issue, the Rapid Adaptive Content Registry (RACR) project is researching a pipeline of interactive tools designed for content managers with little or no training to incorporate content into an adaptive learning ecosystem. This prototype consists of four components:

1) Adaptive Module Registry for composing a set of learning resources and learning objectives (competencies) in an intuitive content-management UI;
2) Rapid Content Analysis Service, which leverages machine learning to analyze web pages (static or dynamic), PDFs, or short videos to generate metadata tags for competencies, estimated duration, and complexity;
3) Preview and Text Extraction interface to review, test, and manually extract text from resources; and
4) Module Simulator to analyze the ability of the available content to adapt to different simulated student patterns (e.g., struggling learner, learner starting with partial mastery, etc.)

This paper outlines the design principles, machine learning performance, and formative usability testing process for this toolkit. For this research, the performance metrics are authoring time, metadata tag quality, deployment reliability (valid content), and personalized pathways (differentiation between different kinds of learners). A comparison of machine learning models based on BERT-S to generate competency tags is presented, which indicates that a general model (not tag-specific) is reasonable for cold-start labels. Initial testing indicates potential usefulness of such a tool, but frustration with delays and limitations for tagging more complex learning resources (e.g., videos, simulations). Strategies and issues for integrating this tool into an enterprise ecosystem are also discussed, such as how specialized tools should integrate with more traditional content management systems.

Keywords:
AUTHORING TOOLS, COMPETENCY BASED TRAINING, MACHINE LEARNING, PERSONALIZED TRAINING

ABOUT THE AUTHORS

Benjamin Nye, Ph.D., is the Director of Learning Science at the University of Southern California, Institute of Creative Technologies (ICT). Ben's research tries to remove barriers development and adoption of adaptive and interactive learning technology so that they can reach larger numbers of learners. Dr. Nye's research has been recognized for excellence in adaptive and intelligent tutoring systems (ONR ITS STEM Grand Challenge; ITSEC 2021 best paper), cognitive agents (BRIMS 2012), and realistic behavior in training simulations (Federal Virtual Worlds Challenge; 2012). His research is on scalable learning technologies and design principles that promote learning, with the goal of making effective learning tools more broadly available.

Aditya Jain is a graduate research assistant at USC ICT in the Learning Science group, with a focus on natural language programming and machine learning systems. Aditya has led research and development on cold-start classification problems, including short-answer grading for user-created dialog-based tutoring (OpenTutor) and competency recommendations for the Rapid Adaptive Content Registry (RACR) system.

Dilan R. Ramirez is a research programmer at USC ICT in the Learning Science group, with a focus on full-stack intelligent systems such as MentorPal virtual mentors and the Rapid Adaptive Content Registry (RACR) system. He is also engaged with the University of Texas at El Paso as a research assistant contributing to spatial visualizations and web applications. Dilan graduated from the University of Texas at El Paso with a BS in Computer Science.

Mark G. Core, Ph.D., is a Research Scientist at ICT and has over 15 years of experience developing explainable artificial intelligence models for use in training. In particular, he co-developed a simulation-independent, dialogue-based explanation system in the context of tactical simulation and virtual humans (i.e., simulated negotiation partners). Dr. Core's empirical research has explored explanation generation in the form of natural language feedback, graphical visualization, and data analytics in the context of simulations, both military and civilian.

William R. Swartout is Chief Technology Officer and co-founder of the USC Institute for Creative Technologies and a research professor in the Computer Science Department at the USC Viterbi School of Engineering. His research interests include virtual humans, explanation and text generation, knowledge acquisition, knowledge representation, intelligent computer-based education, and the development of new AI architectures. Across this work, he has over 80 publications that have received over 8,400 citations. He oversaw the Mission Rehearsal Exercise project, which won awards at NTSA and first place in the 2001 International Conference on Autonomous Agents. Later, he led the NSF-funded museum guides project, which brought ICT-created virtual humans to the Boston Museum of Science, which reached over 250,000 visitors. The project was selected by the NSF for exhibition in their booths at the 2010 AAAS conference and the 2012 USA Science and Engineering Festival in Washington, DC. In 2009, Swartout received the Robert Engelmore Award from the Association for the Advancement of Artificial Intelligence (AAAI). Swartout is a Fellow of the AAAI, has served on their Board of Councilors, and is past chair of the Special Interest Group on Artificial Intelligence (SIGART) of the Association for Computing Machinery (ACM)

Designing a Rapid Adaptive Content Registry (RACR) for Adaptive Learning

Benjamin D. Nye, Aditya Jain, Dilan R. Ramirez, Mark G. Core, William Swartout
University of Southern California, Institute for Creative Technologies
Playa Vista, CA, USA
nye@ict.usc.edu; jainadit@usc.edu; {dramirez, auerbach, core, swartout}@ict.usc.edu

INTRODUCTION

Personalized, adaptive learning offers compelling benefits, often estimated in the range of 0.25 to over 1 standard deviation of improvement, depending on the domain and granularity of adaptation (VanLehn, 2011; Kulik & Fletcher, 2016). Adaptive learning techniques have correspondingly started to see broader use, with platforms such as ALEKS (McGraw Hill), Khan Academy, and SquirrelAI reaching thousands of regular users. However, adaptive learning remains largely constrained to "walled gardens" of content with little support for an end-user organization's content managers or instructors to add or edit content. Scalability is limited because adding content to adaptive systems currently requires substantial AI expertise. More subtly, relevance is also limited because the barriers to specialized expertise mean that adaptive content in adaptive systems cannot be easily updated to track fast-changing topics or enable instructors to customize content.

To support a wider range of domains and changing career paths, adaptive content ecosystems require infrastructure to support a broader space of content that can be matched to each learner's needs (Barr, A., & Robson, 2019). These issues are particularly relevant to efforts such as the U.S. Navy's Ready Relevant Learning (RRL) and associated research effort on the My Navy Learning (MNL) adaptive learning framework (Schatz & DADLAC, 2022). The scale of a service-wide ecosystem such as MNL for adaptive learning requires reusable tools to recommend content across a Sailor's career path and needs. At present, tools to create and manage this scale of adaptive content do not exist. Such tools must:

- Allow the development and maintenance of an adaptive content ecosystem for learning
- With minimal (or no) training, allow the incorporation of content into an adaptive system such that it is ready to deploy
- Support a team of content developers who can update and independently add resources

To reach this scale, we must broaden the range of personnel who can add content. Course developers without specialized AI training must be able to add instructional material to a resource library. A key step in enabling that is to develop tools that use AI to automate parts of the content management process. In many ways, registering adaptive content following best practices for course development: establishing learning objectives, then selecting lessons that teach and assess those objectives. However, adaptive content has unique considerations:

- Metadata: Resources must be tagged with competencies / knowledge components and other data such as duration and complexity, to enable tracking what learners know and to recommend resources.
- Consistency: Different authors must tag similar resources with similar metadata.
- Coverage: Content modules must ideally have sufficient resources for each competency to enable meaningful adaptation.

Research on these areas indicates that AI and machine learning (ML) offer capabilities to assist meeting these requirements. Automatic generation of metadata that describes resources based on their content is the subject of a growing body of research (e.g., Ambite et al., 2019; Bell et al., 2020; De Medio et al., 2016; Garten et al., 2019). Increasing quality of natural language processing tools offers the potential to perform such analyses in real-time, which would make them accessible to content managers. ML tags should also improve consistency and reduce creation of duplicate tags (e.g., an ontology could be used to enforce constraints). Coverage might be estimated through simulated student methodologies (Krauss, Merceron, & Arbanowski, 2019). In this approach, a simulated student could generate distributions of hypothetical resource completions, and a simplified recommender simulator could help determine which competencies have gaps in which sufficient remediation and assessment are unavailable.

The Rapid Adaptive Content Registry (RACR) project at the University of Southern California's Institute for Creative Technologies (ICT) investigates the feasibility of building an adaptive content management tool to quickly import and publish content into an adaptive learning ecosystem by non-specialist content experts. This paper describes the RACR prototype, which assists users registering adaptive content with an intelligent user interface. Before describing the system, we will review relevant background research and systems that inform this work. Next, we will present the user interface design principles for RACR, a comparison of different rapid content tagging techniques explored, and formative feedback from a small number of user testers.

BACKGROUND

Fundamentally, adaptive content management is a subset of content management systems (CMS) more generally. Content management varies widely based on use-cases, and early estimates for CMS overall were that as much as 70% of businesses used custom tools to manage content (Vitari, Ravarini, & Rodhain, 2006). For learning content, three main options exist: generic CMS systems, learning management systems (LMS), and custom-developed solutions. Use of tools to manage learning content varies by sector, with LMS very common in post-secondary education, but custom solutions often used in business contexts. In this research, we position RACR as a custom tool intended to complement more general tools (e.g., LMS, CMS). RACR is intended to be sufficient to describe a learning module (e.g., learning objectives, lesson names, metadata) and provide features to synchronize and exchange this data with systems that need to present content, such as LMS or recommender systems.

CMS research indicates that requirements change as a CMS moves from a small group of specialized authors to a larger set of less-experienced editors. Reddig, Karreman, & Van Der Geest (2008) found that preview capabilities increased the confidence of less-experienced authors. Based on general UI design principles, response time is also essential: a 1s delay typically does not interrupt the user's flow of thought, but users will actively wait only 10s for a response, even with reliable progress indicators (Miller, 1968). For a CMS, team collaboration on content development is also important, as "teacher collective efficacy" (collaboration between instructors) was the strongest factor in student outcomes found in the Hattie & Yates (2018) broad meta-analysis of educational research.

RACR is also informed by two earlier ICT projects developing adaptive learning systems: a) Personal Assistant for Life-Long Learning mobile adaptive learning system (PAL3; Swartout et al., 2016; Hampton et al., 2018) and b) Advancing Teachers' Understanding of Proportional Reasoning Project (ATProportion; Nye et al., 2021). PAL3 has developed a content production pipeline for mobile adaptive content based on Git tags and continuous integration processing. A key element is content testing: both automatic checks for basic integrity (e.g., unit tests), and different stages of quality-assurance testing and previews prior to deploying to learners. The ATProportion project used a similar process, with university-level educators as authors, and increased our understanding of the features and capabilities that less-technical content managers find valuable (e.g., immediate previews to test content as it appears on a real site, rollback).

Based on research on both these systems, non-specialized authors for AI-based learning systems benefit from:

a) Preview/Testing: rapid access to previews, to increase confidence that content is configured appropriately and to enable fast trial-and-error testing.
b) Auto-Save: allow fast or automatic saving of content, with a separate publishing process (to allow testing before learners see it) and an easy process to revert changes.
c) Effective Defaults: although AI systems typically have a large space of potential configuration options, they require only the bare minimum information from the content manager before allowing previews, to ensure working content to test with, even if it is not yet fully adaptive.
d) Familiar Workflow: leverage instructor's prior knowledge about defining the learning objectives for content modules, such that learning resources can be tagged and registered to address the key knowledge and skills.
e) Recognizing Gaps: import of external resources relevant to a target competency is an important feature of a content management system. However, any gaps must be identified in which insufficient assessment and remediation opportunities are available such that new content can be found or created to address these gaps.

Automated metadata tagging offers a way to provide effective defaults for adaptive content. The inputs to such tagging often include automated content analysis (e.g., text, structure), performance patterns (e.g., success on task 1 predicts success on task 2) and expert human authoring. In theory, human authoring is not necessary (i.e., a purely unsupervised ML approach such as topic modeling could be used), but this approach has not traditionally been applied to skill

modeling, because the tag names are not intuitive, and the stability of tag assignments can be hard to track. Typically, two types of expert human authoring data are provided: tag labels (ontology) and resource labels for supervised ML.

Three tagging processes have shown promise to scale up effectively. A traditional approach is to manually create an ontology of competency tags and have experts label the content collection. After sufficient usage by learners, unsupervised learning is applied to discover patterns in performance suggesting new tags, which are used to update the ontology and improve content labels. This approach has been demonstrated to scale up, with ALEKS and the Cognitive Tutor both using this model in widely used commercial systems (Ritter, 2015), but it relies on specialized authors when adding content to the system. Content analysis techniques are typically used when performance data is not yet available (De Medio et al., 2016; Garten et al., 2018). The bottom-up approach is to start with a large space of content (e.g., unsupervised clustering), then assign or "clean up" labels manually. In the "classifier" approach, expert authors manually create an ontology of competencies, and tag a small number of resources; developers then train classifiers to assign labels to remaining content (Bell et al., 2020). If the goal is content discovery (e.g., searching large-scale content for good quality resources), analytics about authors and sources of content are also important features for determining both the quality and relevance of a topic (Ambite et al., 2019). Our work in this research is closest to the "classifier" approach in requiring up-front expert human authoring in the form of an ontology of competency labels. However, by taking a semi-supervised ML approach we do not require content developers adding new resources to have specialized expertise about AI or adaptive learning.

RACR DESIGN

The research questions that drive the Rapid Adaptive Content Registry (RACR) system prototype are:

1) **User Experience**: What features and SME factors impact the usability of an adaptive content registry?
2) **AI for Content Metadata:** How can AI/ML be used to semi-automate metadata labeling when adding content incrementally (e.g., creating a course and tagging content)?
3) **AI for Content Gaps:** What approaches can estimate and explain content coverage/gaps that will affect recommender behavior to a content SME (e.g., what are the use-cases? how to present gaps to content developers?)

For the RACR user experience, the central design principle was to maximize familiarity of the RACR interface with its anticipated users (e.g., instructional designers, instructors). Since the dominant interface type for managing content is the learning management system (LMS), these systems were reviewed as the primary comparison cases ("comps"). This is because nearly every manager of training materials has used an LMS. As a result, RACR's first design guideline was to "walk and talk" like an LMS content manager UI.

This was a natural fit: in many ways, RACR expands on certain core capabilities of a traditional LMS. Modules that contain groups of lessons must be created, organized, and maintained. On the other hand, there are also notable differences. Some elements, such as shared competency labels must be created and managed globally, so they can be used across all modules. Additionally, unlike an LMS, the content registered is not necessarily equivalent to what users see: resources may be recommended in a different order or even not recommended at all. This is because adaptive learning systems traditionally aim for mastery learning, where concepts and skills are studied until they can be applied reliably.

The primary use case that RACR was designed for was to create adaptive course modules based on pre-existing content, such as a non-adaptive content module or by combining resources from many different sources to create a module. Prior to using RACR, a content manager should review the pre-existing training content to identify the main learning objectives that a student should master to complete the content modules. Although such a review is generally a good idea before adopting training content, solid understanding of the content is a critical prerequisite before registering an adaptive module. The steps a content manager then takes to register content are:

1) *Add Competencies*: Review existing competencies and add new ones as needed for the learning objectives for the new module. Competencies are tags for the types of knowledge or skills that are required to master a certain content module. This review does not need to be comprehensive, as duplicate competency tags can be merged later.
2) *Create Module*: Add a new module to RACR, which has a name, description, and associated competencies that are required to master that module.

3) *Add Lessons*: For each lesson in the module, add them in a reasonable default order with their display name, a location (web link), and optional description. This process is similar to popular LMSs such as Canvas or Moodle, but streamlined for RACR's data needs.
4) *Analyze Lessons*: For each lesson added, analyze the lesson to determine the appropriate competency tags, expected duration, and complexity level. This process includes an automated analysis (trigged by an "Analyze" button) and a manual review and edit of tags as needed.
5) *Preview Lessons*: Open the lesson in the Preview tool for review and testing; in the case of a dynamic web page, this step is needed for automated analysis of the text of the lesson.
6) *Simulate Module*: After all lessons are added simulate running the module inside an adaptive system. Select a recommender type and simulated student settings which represent different student archetypes.

Add Competencies (Figure 1). The Competencies Panel is a table of competencies showing the available competency metadata tags for content modules and lessons. Tags are simple to define, consisting of a name and a 1-3 sentence description. As these descriptions are used to help bootstrap automated analyses, they are mandatory and should be written carefully. The system is currently configured to support both "ad hoc" tags (created within RACR), as well as a tag sync operation which will update a set of tags drawn from CaSS (Competency and Skills System; Robson et al, 2020) which is dedicated to managing competencies. Competency tags can be "merged" to reduce duplication and allow ad hoc tags to be replaced with standards-based CaSS tags as needed.

Create Module (Figure 2). The list of modules is based on modern content management interfaces, with a tile list and a simple button (lower right) to create a new module at any time. Module edits are saved automatically with no separate action required. A "publish" function is available for each module, to allow pushing its updates to a system such as the live LMS being used by students; currently the RACR prototype is not connected to such a system. This approach is based on CMS systems (e.g., Contentful Headless CMS), which simplify editing by saving all changes immediately as a "draft" that can be "published" (read by a live system) or "reverted" (return to last published state). While this feature is not as common among LMS, it allows for greater confidence in editing.

Edit Module (Figure 3). After creating a module, the name and description are given. Then, the content module is manually tagged with the competencies that will be the learning objectives for that module.

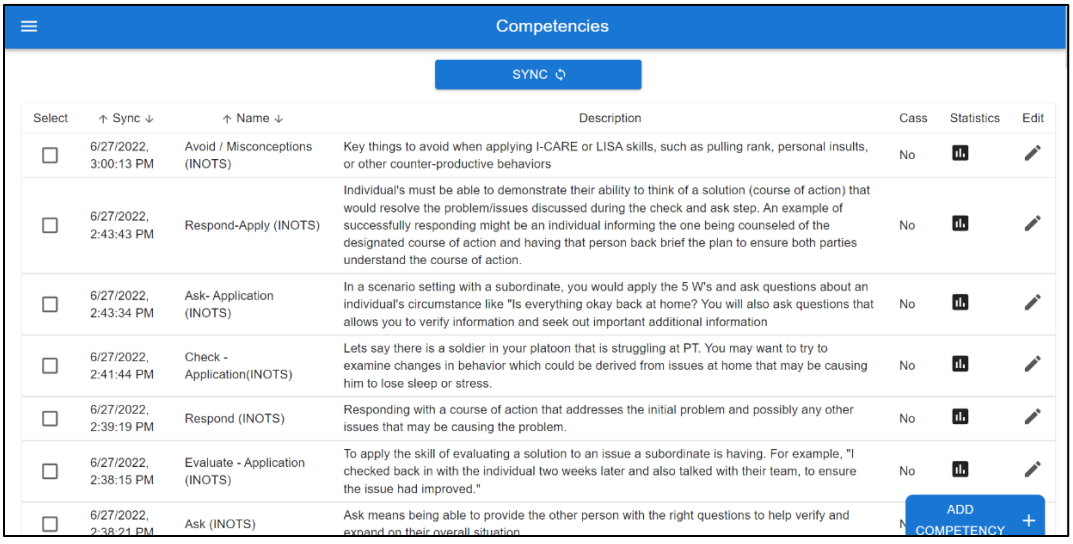

Figure 1. Competencies Panel – Managing and Adding Competencies

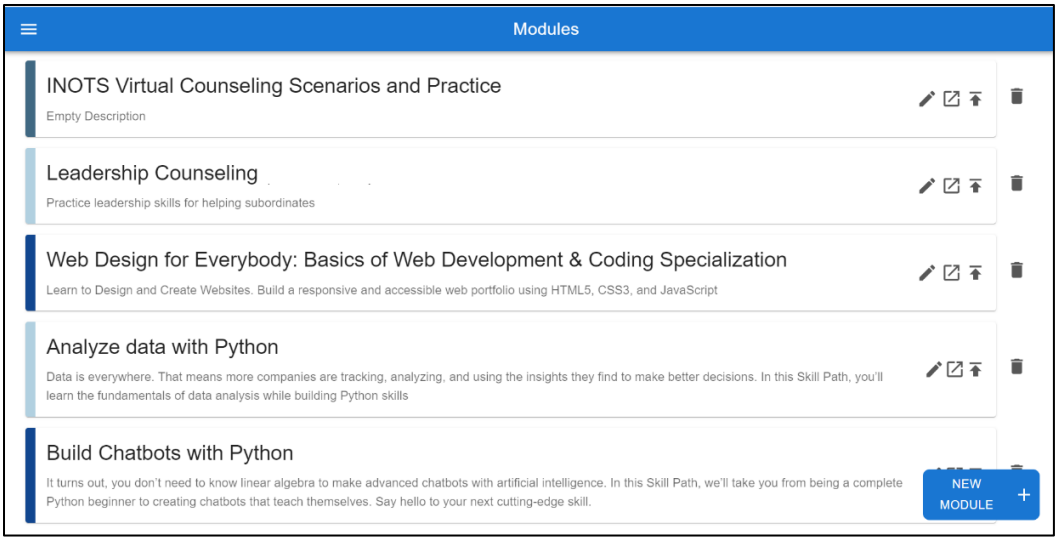

Figure 2. Module List Panel – Adding a Module

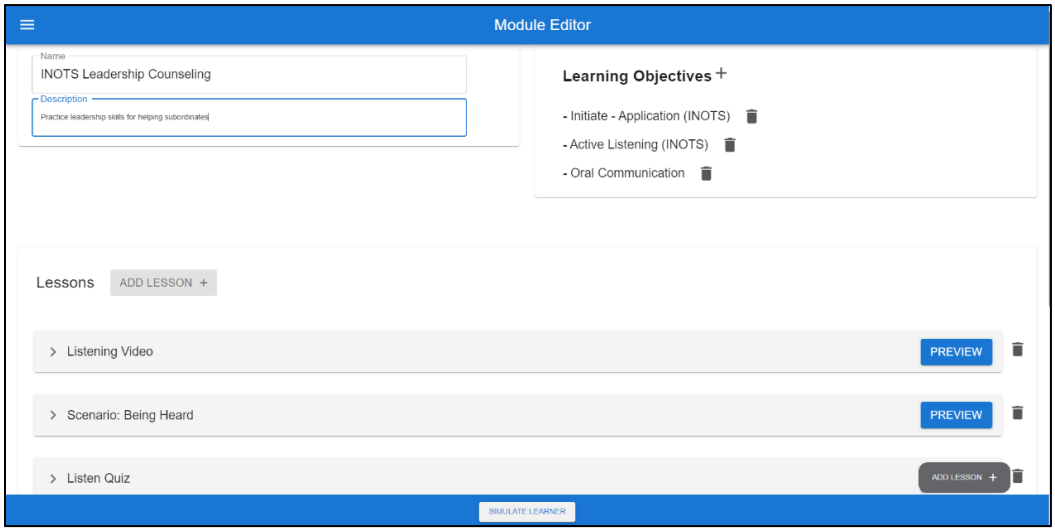

Figure 3. Module Editor Panel – Editing Module Information

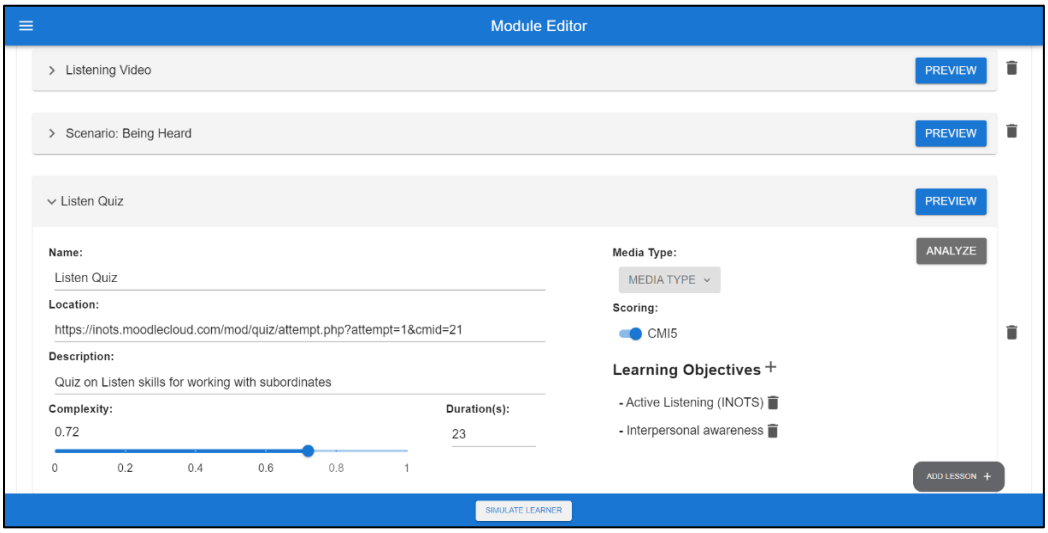

Figure 4. Adding Lessons in the Module Editor Panel

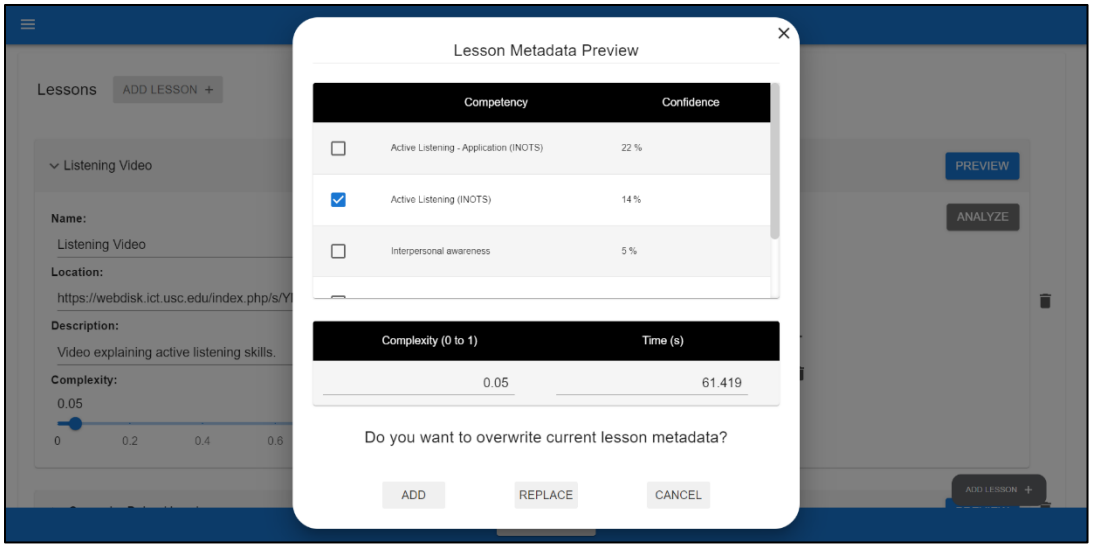

Figure 5: Content Analysis – Semi-Automated Tagging Process

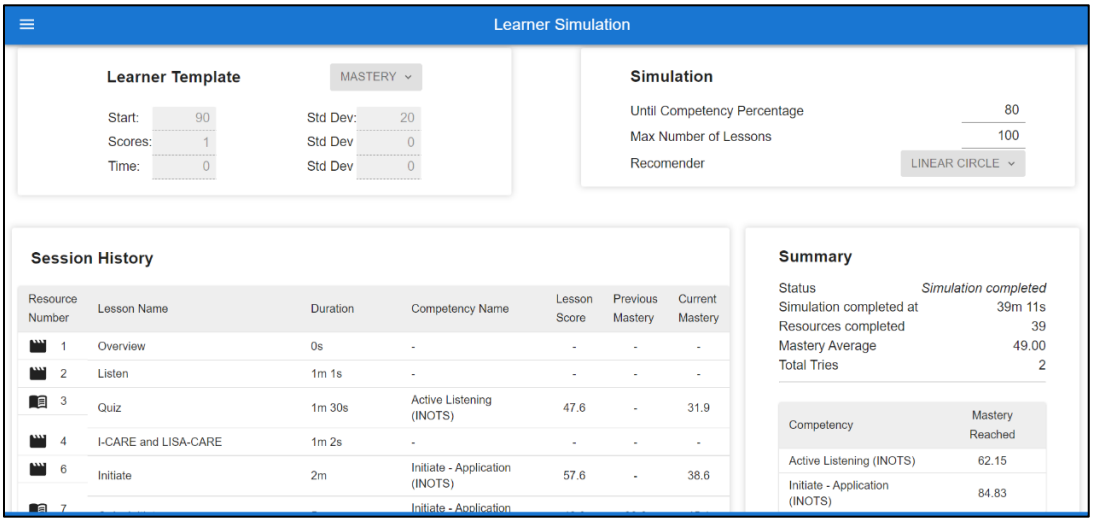

Figure 6: Module Simulation – Results of a Simulation Run

Add Lessons (Figure 4). Creating a lesson is intended to be simple, in that lessons can be either registered with all data up-front or incrementally (e.g., add basic lesson data, then analyze the lesson later). Sorting lessons is done through a simple drag-and-drop operation. Lessons are associated with a module but can be re-used by adding to other modules as well. Thus, lessons are global resources; changing the property of a lesson in one module means that it will be changed in all modules. However, if a lesson is deleted from one module it will persist if it belongs to another unless selected to delete globally. Currently, it is unclear whether this model adds value over the common approach of requiring copies to be made of lessons if they are to be reused.

Analyze Lessons (Figure 5). The semi-automated process for metadata tagging can be initiated as soon as a lesson has a Location (e.g., URL) and a Media Type. Currently, RACR is designed to analyze the following types of media: standard web pages, PDF's, videos, and text blobs (e.g., imported or manually entered). To analyze media requires the ability to extract meaningful text, such as through a web-scraper or transcription of a video. RACR can also be used to analyze dynamic web pages, by using the Preview panel to copy-paste text from a web page into a text blob to analyze. Assuming text content is available, an analysis can be done within a second. To analyze videos requires transcription of the audio slowing the process (can take a minute or more).

Clicking the Analyze button (upper right) extracts the text, generates metadata and then pops up the "Lesson Metadata Preview" window shown in Figure 5. The Preview shows the best-matching competency tags, associated confidence scores, an estimated complexity of the language involved and an estimated duration (e.g., video length, reading time). These generated values can be edited before continuing. At the bottom of the Preview window, the user can Cancel (no changes), Replace (metadata values replaced with the values shown), or Add (competency tags added to the existing set of competencies). The "Add" option allows a resource to be re-analyzed to quickly add new tags.

Simulate Module (Figure 6). After a content module is sufficiently developed, it can be evaluated with simulated students. This simulator assumes a student has a certain performance (with some random variation) as they complete resources. When the simulation's stopping rule is triggered, the simulation concludes and statistics are shown (e.g., one possible stopping rule sets a maximum number of lessons). These simulations are based entirely on the lesson metadata, and are intended to help find gaps indicating the need for additional resources. The upper-left "Learner Template" is where the user type is defined, in terms of distributions for initial knowledge, performance on lessons, and time spent relative to the expected time per resource. The upper-right "Simulation" allows selecting different recommender simulators and stopping rules (e.g., maximum lessons, minimum mastery to pass the module). The "Session History" shows the current simulated run of a learner completing lessons and receiving recommendations. The "Summary" area gives statistics on the simulation run, such as the final average mastery achieved and the estimated mastery of the module's learning objectives.

Compared to the other components of RACR, the module simulation remains the most exploratory. Research on how to evaluate adaptive systems is still evolving, even among specialized groups focused on this area (e.g., Krauss et al., 2019). Moreover, the majority of research for evaluating adaptive systems is focused on the problem of how to best-sequence an existing set of resources rather than how a set of resources might be expanded. However, analysis of adaptive systems shows a number of counter-productive behaviors that may be addressed with content. These include wheel spinning, where a learner attempts problems but cannot master a topic (Beck & Gong, 2013), and lack of content that aligns to the ability level of a learner. Although further exploration is needed, based on this related work, the simulator gives three types of feedback (in the "Summary" area): gaps (too few lessons assess a module learning objective), duration (completing a module takes a large number of hours, such as 12h), and failure to master (simulation found a learner type could not reach mastery, even after a large number of lessons).

SEMI-AUTOMATED TAGGING APPROACHES

Generating candidate metadata tags is a key part of this process, as it can help content managers quickly identify and add multiple relevant competency tags to new lessons. Building this capability required three stages: text extraction from media, content analysis, and learning from prior tags. Each stage will be described briefly, followed by a comparison of different techniques used to tag competencies.

Metadata Tagging Stages

Text Extraction. Since RACR is built to handle many different types of content, multiple tools are used to extract text. Web pages are scraped using Extractnet (Joulin et al., 2016), which can provide both an overall text equivalent of a web page and also content associated with different page sections. This tool is effective on pages that render when viewed by a bot, but is not effective when a page blocks bots or has dynamic elements that require browser-based rendering. Videos are processed to extract their transcript, either by extracting subtitles (FFMPEG) or by transcribing the video (e.g., external web services or Sphinx for on-server transcription; cmusphinx.github.io). Given the now vast advantages of modern web services for transcription (Këpuska & Bohouta, 2017), these should be strongly preferred unless firewall or content security needs prevent using commodity speech-to-text.

Content Analysis. Analysis of content was designed to be fast, as the real-time performance was a key requirement. The analysis currently produces competency metadata tags, text complexity, and an expected duration. Competency labels are suggested based on their confidence values in a multiple logistic classifier model using S-BERT sentence embeddings (Reimers & Gurevych, 2019). Text complexity and expected reading time were generated based on TextStat metrics which consider reading difficulty level (Bansal, 2021). For videos, duration was based on the video length instead. So far, complexity tagging needs further research because existing tools consider only the text but not the task context (e.g., a task to answer questions about an article is qualitatively harder than only reading an article).

Learning From Prior Tags. After competency tags are accepted and assigned to lessons, these are stored in a database that contains the tags and their associated extracted text (i.e., text blobs). These stored tags can be used in supervised ML to retrain the classifier model. This training process is currently started through a button in the user interface, and for the current amount of data, it is relatively quick (e.g., minutes).

Comparing Metadata Tagging Approaches.

To explore metadata tagging approaches, an example data set was built based on Wikipedia AI articles (478 wiki pages; 12 subtopics with at least 5-page examples). The goal of this exploration was to primarily investigate cold-start performance, where the number of known examples starts very small, since RACR must be able to suggest tags even if they have no actual examples yet (e.g., only the label name and description). Initial testing and earlier research indicated that an S-BERT transformer model (Sentence Bidirectional Encoder Representations from Transformers; Reimers & Gurevych, 2019) could be used to produce useful label suggestions. S-BERT is a neural network transformer model trained to take a sentence as an input and produce a vector of numbers (e.g., 768 float numbers), where the cosine similarity between the vectors for two sentences gives a high-quality measure of their semantic similarity.

Multiple techniques can be used in combination with S-BERT outputs to potentially enhance its accuracy in detecting relevant competency tags. However, for a real-time system, performance must also be balanced against complexity and time to retrain and store models. Three techniques were compared, which each used the S-BERT embeddings to generate features for a logistic regression model:

1) Raw Embeddings [*Label ~ S-BERT(input)*]: This model trained a multiple logistic classifier where the training data consisted of the pair of a tag on a document (label) and 768 numeric features (one for each dimension of the embedding for an input document). The primary disadvantage of this model is that it ideally needs multiple tagged examples for each category, which is not always feasible.
2) Competency-Only [*hasLabel ~ Cosine(S-BERT(competency), S-BERT(input))*]: This model trained a single classifier model which attempted to predict a binary outcome (hasLabel) based on a given cosine similarity between the competency description and the sentence embedding of the input document. This model does not learn from new labeled data but can be used even with no tagged examples.
3) Competency Plus Tagged Documents [*hasLabel ~ Cosine(S-BERT(competency+tagged), S-BERT(input))*]: This model is similar to the Competency-Only model, except that it can be updated by replacing the competency S-BERT embedding with an average of that and documents tagged so far. This enables learning from data while still keeping a very simple model.

Table 1. Comparison of Model Accuracy for Recommending Competencies for a Wiki Document (Leave-One-Out Cross-Validation, whole documents left out)

Model	Accuracy (Top-1)	Accuracy (Top-3)	Accuracy (Top-5)
Raw Embeddings	39%	**66%**	**77%**
Competency-Only	38%	63%	74%
Competency Plus Tagged	**42%**	62%	74%

As shown in Table 1, comparing these techniques found that they were qualitatively similar in accuracy for suggesting recommendations. Leave-one-out cross-validation found that all three models tended to provide a given recommendation within the top 5 most-confident labels. This analysis shows a small advantage to the Raw Embeddings on the Top-N accuracy to ensure a correct label is recommended. However, the advantage was small and it requires maintaining a substantially more complex model. By comparison, the Competency Plus Tagged lessons model performs nearly as well and in some cases, better (top suggestion accuracy). Exploratory research was conducted to build an ensemble model between Raw Embeddings and Competency Plus Tagged models, where the model would steadily favor the Raw Embeddings as more examples were tagged. However, this added complexity and did not substantially improve performance. As a result, the simpler Competency Plus Tagged model was selected for the current RACR prototype. However, it is straightforward to extend RACR to use additional models or external API's to suggest competency tags. Based on internal testing as part of the RACR content management UI,

this model has worked reasonably, where the best two competency tags tend to be in the top 3 or 4 tags recommended based on classifier confidence.

USABILITY – FORMATIVE FEEDBACK

To investigate the usability of the system, RACR was applied to a course based on the Immersive Navy Officer Training System (INOTS) scenario-based intelligent tutoring system (Georgila et al., 2019; Perez, Skinner, & Chatelier, 2016). INOTS was designed for junior officers-in-training to learn about and practice basic counseling skills to address personal issues or performance problems of subordinates. The INOTS web-based module has three types of media: videos, multiple-choice quizzes with feedback, and dynamic interactive training scenarios.

A usability session was conducted to gather formative feedback from three junior officers-in-training with no experience with RACR or INOTS. In this session, the formative testers were given the INOTS content as a series of links and lesson titles. They were instructed to follow the steps described in Figures 1 through 6, starting with a review of the content links and materials, adding four competencies to each, and registering their own INOTS module with 31 lessons to add and analyze. Testers logged their time to complete each activity phase and also provided survey feedback after each phase, for a period of 3h (including time for surveys and two 10 minute breaks). Usability was rated on a 6-point scale from 1=Completely Disagree, and 6=Completely Agree with statements in the form "RACR was clear and easy to use" and "Using RACR is a good idea." In the time allotted the testers did not completely register and analyze the whole module, with each tester completing about 20 lessons in the allotted time.

As shown in Table 2, formative testers were generally positive about the system but hit substantial issues with the Analysis step. Early steps, including reviewing and adding competencies, ran smoothly for testers. The primary feedback for these steps were that more guidance was needed in how to format competency names and instructions/examples on good descriptions for competencies and modules. The analysis stage had intermittent errors, partly due to more concurrent users than had previously been tested. Two additional issues were encountered. First, some users did not set the correct media type (or set no media type), which meant that the analysis of videos failed with an unhelpful error message. Second, video transcription resulted in long delays, and these were compounded when multiple users tried to analyze videos at the same time. While web pages typically produced analysis results within a second, videos always took over 10s and sometimes timed out with no result. This resulted in a significant split between high ratings of RACR Content Analysis overall (a solid "Agree" for "Good Idea") versus lower ratings for the Analyze Lessons stage (negative ratings for both "Good Idea" and "Clear and Easy to Use"). Open response feedback and high ratings of RACR Content Analysis indicate that automated analysis is a desirable and mostly intuitive feature, but that the specific implementation must be improved to be more robust against unexpected content types or related issues the required waiting or retrying analyses.

The simulation step was rated lowest overall and hit blocking bugs for some testers (e.g., would not complete), which appeared to be caused by unexpected values when testers registered certain lessons. As a result, the numerical ratings are not necessarily a clear indicator of the usability for that panel because testers were not always considering only their own UI (e.g., at least one tester was looking at another tester's screen to look at outputs due to bugs on their side). With that said, a larger issue was that all testers (regardless of positive/negative ratings) were not clear on how to interpret the simulation numbers or what to do with them, as a conceptual issue rather than due to bugs. This raises questions about the overall utility of a user-facing simulation panel, at least for a typical instructor or content manager.

Table 2. Formative Testing Feedback (N=3 testers)

	Overall	Content Analysis	1. Add Compet.	2. Create Module	3. Edit Module	4. Add Lessons	5. Analyze Lessons	6. Simul-ation
Time (min)	116	59	9	4	7	36	59	2
Clear and Easy to Use	3.7	4.0	5.0	5.7	4.3	5.3	1.7	2.3
Good Idea	4.3	5.0	5.0	5.3	5.7	5.3	3.3	2.3

DISCUSSION

Considering the tester's open response comments and usability ratings, the consensus opinion was that "the overall concept of RACR is useful" but regarding the analysis (particularly video analysis) we needed to "fix the bugs and the time it takes to analyze each lesson." As this was the first session with multiple simultaneous testers, it was anticipated that new bugs would be encountered and the majority of issues with the analysis function have either been fixed or are straightforward to resolve, such as using a commercial transcription engine for videos. Most importantly, the overall flow for registering content was well-understood, and testers appeared to have a clear understanding of how competency tags could be created and used to tag both content modules and lessons.

The semi-automated analysis process was understood by testers and they indicated it was significantly faster than manual tagging. However, the reliability must be improved significantly. An obvious improvement would be to auto-detect the media type when possible, rather than displaying warnings and requiring users to fix it when it is wrong. One tester also strongly requested that RACR "make 'analyze' automatic" when initially adding resources, so the tags could be added while adding more lessons and then re-visited later. This might be a good approach to speed up tagging because analysis could occur as a background task. However, the fully automated analysis raises risks that some content managers might blindly accept tags and not take the effort to validate or improve them. Additional features to encourage verifying tags would probably be required (e.g., an "unverified" flag for lessons where tags were generated but never approved/edited). A complementary piece of user feedback was that for manual tagging, it would help to be able to select multiple lessons to "paint" or "fill" lessons with a certain set of competencies since it took non-trivial time to add the same tag to multiple lessons.

Larger numbers of testers will be required to fully understand how well the automated analysis assists tag quality, particularly given that some testers had more trouble getting automated tags to generate reliably. Overall, lesson tags were relevant to the resources, though for early resources, the tagging task for INOTS is relatively simple (e.g., an "Initiate" skill has an associated "Initiate" video and quiz, in addition to being used in scenarios). A qualitative review of tags versus a set of expert-assigned tags indicates that some users tend to provide minimalist tags (e.g., a small number that is most-aligned to the lesson) while another user type tends to provide all tags that were somewhat relevant after reviewing a resource manually. Unfortunately, it is not clear which approach should be encouraged: different adaptive systems work better with sparser or denser tag sets. This might indicate that competency tags should have an additional setting for "primary" or a relevance level so that content managers who prefer dense tags can better align with those with sparser tags. Finally, sparse-tagging content managers are likely to only tag lessons with competencies that they registered for the current content module, even if other competencies might be relevant. This would mean that secondary skills or knowledge might not be tagged or tracked unless a process exists to re-evaluate tags for the whole corpus of lessons at some point.

The simulation testing showed both difficulties understanding this step and less enthusiasm over how useful it would be (e.g., "I do not understand where the scores and numbers are coming from"). Overall, this feedback suggests that the module simulator is not appropriate for general content managers. Alternate designs might help content managers better understand how their content will behave in an adaptive system. While this was a small number of testers, they seemed to understand the "Session History" (the list of lessons a simulated student would have completed) and the concept that it could identify gaps that needed more lessons. One alternative would be to use simulated students exclusively in the background, to suggest content or changes to the module when inside the module editor (e.g., no separate "Simulation" panel). Another approach might be to have a "simulate" panel that shows a representative session history for three standard student archetypes (e.g., High-Performing, Partial Knowledge, Struggling) so that these can be qualitatively reviewed rather than giving summaries of quantitative metrics. So far, little research has investigated this area, so different techniques will need to be explored.

CONCLUSIONS AND FUTURE DIRECTIONS

The potential value of RACR and tools like it for adaptive systems is significant: prior to developing RACR, the time for an expert to register and verify a module like INOTS for an adaptive system manually (e.g., using markup files for metadata) was on the order of 1.5 to 2 weeks at 20% to 50% effort (e.g., 12h-40h). By comparison, testing with RACR showed that the initial registration process could be done within a half-day, with non-specialist authors. Even if it took

an expert a full day to adjust tags and review content (an upper-bound estimate), tools such as RACR will substantially accelerate building adaptive content modules where sufficient content already exists.

For testing adaptive content, fundamental research is still required on how to help content managers anticipate how their registered resources will be recommended in an adaptive learning system. This research found that a simulation panel could be reviewed quickly, but it was not intuitive to testers. Future research should probably investigate two directions: black box or interactive. Black box recommendations could suggest ways to improve an adaptive module while editing the module (e.g., hints, tooltips, warnings before publishing). On the other hand, an interactive testbed might allow an instructor to "play through" a set of resources as a student, which is a functionality that some standard LMS platforms offer. Particularly if resources could be skipped, an instructor might be able to check for issues fairly quickly and in a way that is more familiar to them.

Automated competency tag generation also shows strong benefits for the process, making it easy to add some or all tags that are relevant to a learning resource from a suggested list rather than manually selecting them. However, the current tagging approaches are not yet high-enough quality to allow fully automated tagging: the ideal tags are reliably suggested, but irrelevant or less-relevant tags also tend to be included in the top-5 (weak precision). Additionally, user feedback indicates that content managers do not want to wait until longer analyses complete (e.g., for videos). Together, these factors mean that further design and user testing is needed to find ways that analyses can happen in the background while a content manager is still directed to verify and fix tags before publishing an adaptive module. As one benefit, fully-automated tags could be re-tagged and updated at any point prior to verification/manual edits to take advantage of new supervised labels and models. Future research on RACR is expected to look at this further because it offers a path to build large-scale adaptive content: automatic tagging done in the background could enable registering large amounts of content while deferring expert verification and editing to when content is being included in published modules.

ACKNOWLEDGEMENTS

This research was sponsored by U.S. Office of Naval Research (ONR) through the USC ICT University Affiliated Research Center (W911NF-14D0005). However, all statements in this work are the work of the authors alone and do not necessarily reflect the views of sponsors, and no official endorsement should be inferred.

REFERENCES

Ambite, J. L., Fierro, L., Gordon, J., Burns, G. A., Geigl, F., Lerman, K., & Van Horn, J. D. (2019). Bd2k training coordinating center's erudite: the educational resource discovery index for data science. *IEEE Transactions on Emerging Topics in Computing*, *9*(1), 316-328.

Bansal, S. (2021). TextStat. Retrieved from: https://github.com/shivam5992/textstat.

Barr, A., & Robson, R. (2019). Missing pieces: infrastructure requirements for adaptive instructional systems. In *International Conference on Human-Computer Interaction* (pp. 169-178). Springer, Cham.

Beck, J. E., & Gong, Y. (2013). Wheel-spinning: Students who fail to master a skill. In *International conference on artificial intelligence in education* (pp. 431-440). Springer, Berlin, Heidelberg.

Bell, B., Brawner, K., Brown, D., & Kelsey, E. (2020). Matching Content to Competencies with Machine-Learning: A Service-Oriented Content Alignment Tool for Authoring in GIFT and Beyond. In *Proceedings of the 8th Annual Generalized Intelligent Framework for Users Symposium (GIFTSym8)* (Vol. 101). US Army Combat Capabilities Development Command–Soldier Center.

De Medio, C., Gasparetti, F., Limongelli, C., Sciarrone, F., & Temperini, M. (2016). Automatic extraction of prerequisites among learning objects using wikipedia-based content analysis. In *International conference on intelligent tutoring systems* (pp. 375-381). Springer, Cham.

Garten, J., Hoover, J., Johnson, K. M., Boghrati, R., Iskiwitch, C., & Dehghani, M. (2018). Dictionaries and distributions: Combining expert knowledge and large scale textual data content analysis. *Behavior research methods*, *50*(1), 344-361.

Georgila, K., Core, M. G., Nye, B. D., Karumbaiah, S., Auerbach, D., & Ram, M. (2019). Using reinforcement learning to optimize the policies of an intelligent tutoring system for interpersonal skills training. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (pp. 737-745).

Hattie, J., & Yates, G. C. (2018). *Visible learning and the science of how we learn*. Routledge.

Hampton, A. J., Nye, B. D., Pavlik, P. I., Swartout, W. R., Graesser, A. C., & Gunderson, J. (2018). Mitigating knowledge decay from instruction with voluntary use of an adaptive learning system. In *International Conference on Artificial Intelligence in Education* (pp. 119-133). Springer, Cham.

Huang, Y., Yudelson, M., Han, S., He, D., & Brusilovsky, P. (2016). A framework for dynamic knowledge modeling in textbook-based learning. In Proceedings of the 2016 conference on user modeling adaptation and personalization (pp. 141-150).

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Këpuska, V., & Bohouta, G. (2017). Comparing speech recognition systems (Microsoft API, Google API and CMU Sphinx). *Int. J. Eng. Res. Appl*, *7*(03), 20-24.

Krauss, C., Merceron, A., & Arbanowski, S. (2019). The timeliness deviation: A novel approach to evaluate educational recommender systems for closed-courses. In *Proceedings of the 9th International Conference on Learning Analytics & Knowledge* (pp. 195-204).

Kulik, J. A., & Fletcher, J. D. (2016). Effectiveness of intelligent tutoring systems: a meta-analytic review. *Review of educational research*, *86*(1), 42-78.

Miller, R. B. (1968). Response time in man-computer conversational transactions. *Proc. AFIPS Fall Joint Computer Conference* Vol. 33, 267-277.

Nye, B.D., Shiel, A., Olmez, B., Mittal, A., Latta, J., Auerbach, D., & Copur-Gencturk, Y. (2021). Virtual Agents for Real Teachers: Applying AI to Support Professional Development of Proportional Reasoning. Florida AI Research Society (FLAIRS) Conference 2021.

Perez, R. S., Skinner, A., & Chatelier, P. (2016). Lessons Learned from Intelligent Tutoring Research for Simulation. *Using Games and Simulations for Teaching and Assessment: Key Issues*, 99.

Ritter, S. (2015). Authoring for the product lifecycle. *Design Recommendations for Intelligent Tutoring Systems*, *Vol. 3: Authoring Tools*, 137-144.

Robson, R., Havas, K., Ray, R., Schatz, S., Stafford, M., Robson, E., ... & Gordon, J. (2020). *Competency Framework Development Process Report*. Eduworks Corporation Corvallis United States.

Reddig, D., Karreman, J., & Van Der Geest, T. (2008). Watch out for the preview: The effects of a preview on the usability of a Content Management System and on the users' confidence level. In *2008 IEEE International Professional Communication Conference* (pp. 1-7). IEEE.

Schatz, S., & DADLAC (2022). *DADLAC (Defense Advanced Distributed Learning Advisory Committee) 2021 Annual Report*. Joint Advanced Distributed Learning Co-Lab (JADLL), Orlando, FL.

Swartout, W., Nye, B. D., Hartholt, A., Reilly, A., Graesser, A. C., VanLehn, K., ... & Rosenberg, M. (2016). Designing a personal assistant for life-long learning (PAL3). In *29th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2016* (pp. 491-496). AAAI Press.

Vitari, C., Ravarini, A., & Rodhain, F. (2006). An Analysis Framework for The Evaluation of Content Management Systems (CMS). *Communications of the Association for Information Systems*, *18*, 782-804.