Supplementing Instructor-Led Training with AI: Lessons Learned Developing a Virtual SME

Nelson Hurst, John Thornton, Hung Le
Integration Innovation Inc. (i3)
Huntsville, AL
nelson.hurst@i3-corps.com, john.thornton@i3-corps.com, hung.le@i3-corps.com

ABSTRACT

Studies have shown that digital training and interactive multimedia instruction (IMI) products improve learner outcomes through increased interactivity and immersion. Instructor-led IMI is a specific type of training where instructors guide students through a digital course, providing invaluable feedback and guidance. However, an instructor's time is finite, and teachable moments only scale linearly. Qualified instructors are 'aging out' of training programs, which could limit the number of students allowed to take specific training. What if we could use AI to augment an instructor? Or even create a virtual surrogate instructor to allow real instructors to teach and reach more students?

This paper describes a study funded by US Army PEO Aviation that leverages large language models (LLMs), text-to-speech (TTS), and other AI tools to create a high-fidelity intelligent agent (IA). The agent can interact with the user and the virtual environment to emulate a subject matter expert for instructor-led training. Users can interact with the agent freely through voice or text during self-paced learning to fill in any gaps in the learning content. The paper outlines a prototype application's capabilities and technical limitations, what tools and technologies were considered, and the tests and results to select those technologies integrated into the final prototype. The prototype architecture, which includes the interactive agent's 3D avatar, voice, perception and context of the virtual environment, and course material is also discussed as well as a custom-trained LLM and knowledge base that serves as the "brain" of the agent. Finally, the paper covers the agent's current hardware requirements and usage limits, lessons learned during development, and future steps for continued development of the prototype.

ABOUT THE AUTHORS

Nelson Hurst is a Sr. Principal Software Engineer at Integration Innovation, Inc. (i3) with over two decades of professional experience in multimedia and software development. He specializes in creating proprietary software, pioneering productivity tools, and streamlining workflows. Nelson's expertise spans programming, web development, full-stack engineering, mobile game and app development, and cross-platform work. Nelson also has competencies in machine learning, prompt engineering, fine-tuning, and working with large language models.

John Thornton is a Department Manager and software engineer at Integration Innovation Inc. (i3) with a background in full-stack web and cross-platform application development. His passion is interactive computing and incorporating new and innovative technologies into experiences that improve or redefine how humans interact with each other and the world around them.

Hung Le is a software engineer at Integration Innovation Inc. (i3), where he contributes to cutting-edge training and simulation projects. He holds a Master's degree in computer engineering from the University of Central Florida. His passion lies in AI, machine learning, and automation tools, areas where he continually seeks to innovate and make a positive impact.

Supplementing Instructor-Led Training with AI: Lessons Learned Developing a Virtual SME

Nelson Hurst, John Thornton, Hung Le
Integration Innovation Inc. (i3)
Huntsville, AL
nelson.hurst@i3-corps.com, john.thornton@i3-corps.com, hung.le@i3-corps.com

INTRODUCTION

Digital training and interactive multimedia instruction (IMI) have revolutionized educational methodologies by enhancing learner outcomes through increased interactivity and immersion. IMI leverages various digital tools and platforms to create engaging and effective learning environments, which have been shown to improve knowledge retention and skill acquisition across diverse fields (Graves, Blankenbeckler, Wampler, & Roberts, 1996). When coupled with immersive 3D training content, students can visualize and understand complex systems and tasks with nothing more than a desktop computer, mobile device, or extended reality (XR) headset. For example, take a maintainer of a UH-60 helicopter. The maintainer must learn how the fuel system operates before performing a complex maintenance procedure. With immersive digital training and IMI, the maintainer can view the fuel flow through the system in full 3D and perform the task virtually multiple times for familiarization. The need for dedicated time on a part-task trainer or hands-on time with the UH-60 is greatly diminished due to the ability to perform complex learning procedures virtually.

IMI integrates text, 2D and 3D graphics, animation, sound, and video to create a rich, user-engaged learning experience. It comprises four levels of increasing complexity and interactivity: Level I provides basic instruction through linear presentation and simple practice exercises. Level II introduces limited branching, allowing learners to make choices that affect the instructional sequence. Level III allows for complex branching and simulated environments, fostering problem-solving skills. Finally, Level IV offers immersive, fully interactive simulations and games that adapt to individual learner responses (US Department of Defense, 2001). The Department of Defense employs IMI across these levels to deliver comprehensive training and education to military personnel, optimizing learning efficiency and tailoring content to service members' diverse needs and learning styles.

Instructor-led IMI represents a specialized form of this training, where instructors guide students through digital courses, providing critical feedback and personalized guidance. This approach combines the benefits of traditional teaching with the advantages of digital interactivity, creating a hybrid model that maximizes learning efficiency (Nelson, 2021). However, the availability of qualified instructors limits the scalability of instructor-led IMI. As experienced educators retire and service branches face instructor shortages, the capacity to deliver high-quality training to a growing number of students diminishes, posing a significant challenge to institutions and training programs (Novelly, 2023).

Integrating artificial intelligence (AI) into IMI presents a promising solution to address this issue. AI can augment the capabilities of human instructors by automating routine tasks, providing real-time feedback, and personalizing learning experiences based on individual student needs. Furthermore, AI-driven virtual surrogate instructors could potentially replicate the role of human instructors, enabling the delivery of high-quality education to a larger audience without human resource limitations. This paper describes developing and evaluating an AI-driven virtual subject matter expert (SME) that allows students to ask questions during training and receive human instructor-like contextual information for training tasks. This study evaluated various large language models (LLMs), speech recognition, and text to speech libraries.

This paper describes an approach for standing up a local, on-premises LLM that is incorporated into an IMI training application. The LLM is augmented with additional data, such as technical manuals, to provide SME-like expertise to the user. While this research focused on IMI, the approach and lessons learned can be applied to any domain where a virtual SME is needed. Multiple LLMs were evaluated using a set of metrics and their results are discussed. User input

mechanisms, such as keyword and speech recognition, were also evaluated and incorporated into the approach. Future improvements and research areas are discussed so that readers understand the current gaps in the approach.

The Evolution of Large Language Models

The development of LLMs is advancing rapidly, pushed by increased computational power, vast amounts of training data, and continuous innovations in model architectures and training techniques. This rapid evolution has led to significant breakthroughs in LLM capabilities, enabling them to tackle increasingly complex tasks with impressive accuracy and fluency. However, the swift pace of progress also presents challenges in evaluation and benchmarking, as the metrics and standards must constantly evolve to keep up with these models' ever-expanding capabilities.

Recent Advances in LLMs: LLMs have recently demonstrated remarkable capabilities in natural language processing tasks and beyond. These works encompass diverse topics such as architectural innovations, better training strategies, unsupervised pre-training, context length improvements, fine-tuning, multi-modal LLMs, robotics, datasets, benchmarking, efficiency, and more (Naveed, et al., 2024).

Evaluation of LLMs: The evaluation of LLMs is not just a task-level concern; it also has significant societal implications. Understanding the potential risks of LLMs is increasingly critical. This responsibility highlights the need for considerable efforts to evaluate LLMs from multiple angles, including what to evaluate, where to evaluate, and how to evaluate (Chang, et al., 2023).

Challenges in LLMs: Despite their impressive capabilities, LLMs have associated challenges and concerns. These potential issues, including biases in the training data, inaccurate or inappropriate content generation, and ethical considerations regarding their use, need to be addressed. Over time, the availability of inexpensive computational power and large datasets has improved LLMs' capabilities and raised new challenges, but with determination and focus, these challenges can be overcome (Patil & Gudivada, 2024).

Applications of LLMs: LLMs have found diverse applications, from instruction controllable text summarization to biomedical tasks and process mining. However, each of these applications brings its own set of unique challenges, further highlighting the complexity of the field (Liu, et al., 2023).

Cloud-Based vs. Local LLMs

Organizations must consider various factors when deciding between cloud-based and local deployment of large language models (LLMs). Cloud-based LLMs offer scalability, providing on-demand access to substantial computational resources. They also offer cost efficiency by avoiding upfront investments in expensive hardware. Furthermore, cloud platforms provide APIs, tools, and managed services for streamlined deployment and access to pre-trained models that can be fine-tuned and easily deployed.

On the other hand, local LLMs provide control, enabling natural language processing without reliance on cloud services. They require only upfront hardware and software costs, with no ongoing subscription fees. Moreover, local deployment ensures data privacy, as sensitive information remains within the organization's environment.

We used local LLMs to keep the data private and offline using Retrieval-Augmented Generation (RAG). This approach allows us to control our data entirely and ensures that sensitive information is not exposed to external parties. Additionally, reliance on cloud-based infrastructure introduces risks such as service disruptions, network vulnerabilities, or data breaches, which can be mitigated by deploying LLMs locally.

Retrieval-Augmented Generation and Quantization

Retrieval-Augmented Generation (RAG) combines LLMs with external knowledge sources to generate more accurate and informative responses. We decided that by leveraging the vast amount of information available through external sources, RAG would help our local LLMs provide more relevant and up-to-date answers without requiring extensive local fine-tuning. Using a collection of PDFs from our training manuals and converting it to vector representation provided the LLM with the data necessary to answer and respond to information that is not publicly available. Using

Facebook AI Similarity Search (FAISS), we could quickly perform a similarity search on large amounts of data and retrieve relevant information based on the user's requests.

Quantization techniques reduce the precision of the model's weights and activations, leading to significant savings in memory and computational requirements. Using the llama.cpp software library allowed us to leverage quantization to enable efficient local inference of LLMs across a wide range of hardware platforms. The library provides a plain C/C++ implementation without any external dependencies and is actively updated frequently.

TECHNICAL APPROACH

In order to implement our virtual SME we chose to use the Unity Engine as the frontend and rendering engine as it is an industry standard for high level IMI development allowing us to have a 3D avatar and environment for our SME to interact with. Its support for multiple deployment platforms, availability of plugins, and multiple inputs support allowed us to focus more time on the technical pipeline for our virtual SME. As described in Figure 1, the Unity application allows the user to interact with the SME through either text or audio inputs. This is performed by using the chat interface within the application to type their query or by speaking a keyword followed by their query using the embedded keyword recognition implementation. The query from the user and any necessary context from the Unity application such as location within the course instruction or game state is then sent using a web request to the LLM interface, either on the local machine or hosted at a remote endpoint, where a response is generated with input from the RAG server. This response is then converted to speech using a TTS implementation and sent back to the Unity application where the user can see and hear the SME's response completing the communication loop. Portions of the response can contain information or metadata just for the Unity application not to include in the response to the user, such as a list of 3D components related to the response, that the Unity application could then leverage to have our SME's 3D avatar navigate toward, point at, and/or highlight while the user is listening to the response. This sets up a pattern for expanding our SME's interactions with the environment as the app context and LLM's knowledge set grows. While our approach was successful in running on a single high-powered machine with substantial RAM and graphics power, the LLM and RAG server benefit significantly from being hosted on another machine from the frontend to allow the dedication of more resources for response generation and therefore faster response generation or increased context for the response resulting in higher quality or more accurate answers.

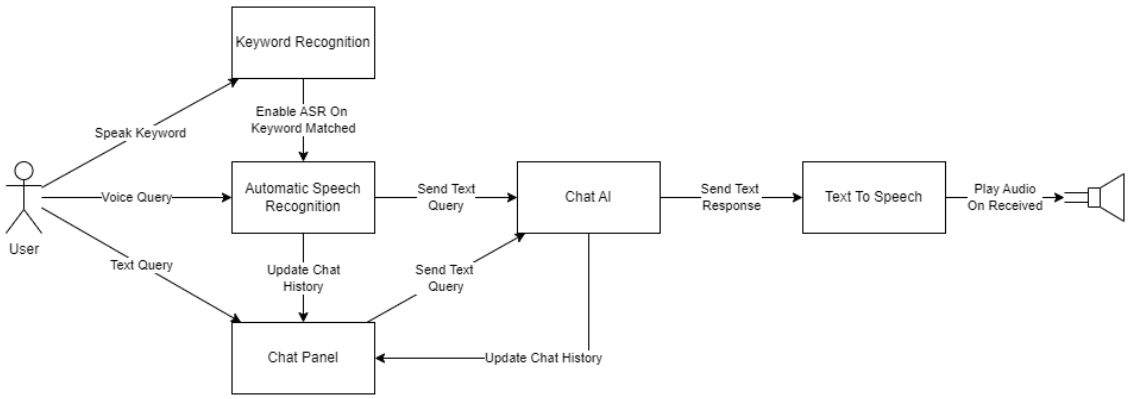

Figure 1: Data Flow Between System Components

Keyword Recognition

Keyword recognition, or Wake-word Recognition, is a small system often used in embedded devices to control short commands or activate other functions. It is present in most modern smartphones, smart TVs, and home assistant devices like Amazon Echo and Google Nest.

Unlike speech recognition, keyword recognition aims to detect predefined keywords. The input audio is often limited to isolated audio, which is expected to contain only the keywords; therefore, mid-sentence detection is not supported. By constraining the input audio and providing predefined keywords, the problem is changed from "parsing audio to massive corpus of vocabulary" to the far simpler problem of "similarity comparison."

Because of their reduced complexity, keyword recognition systems are usually small and computationally inexpensive. This allows the system to be run on lower-end devices or often as a subsystem of the primary application, as in our approach. In comparison, speech recognition services are more complex and computationally expensive and often require remote hosting to ensure quality without compromising the performance of the front-end application.

Many free products and libraries, such as Catalyst or OpenWakeWord, are available for keyword recognition. However, the Unity game engine comes with a decent implementation of keyword recognition integration within the engine. From our experience, there was no significant difference in quality between the Unity built-in implementation and the other libraries we tested, so it was chosen to save development time and effort.

Automatic Speech Recognition

After researching and experimenting with various options, we chose the Whisper Large v3 of the OpenAI automatic speech recognition model (Radford, 2022). During our testing, the model was highly accurate at converting speech to text and supported multiple languages, expanding the user base that could use our application without swapping models each time. The model was also easy to implement, allowing us to focus on getting it to work with other application modules.

With keyword recognition, it is easy to start recording the user's speech when a keyword is detected, but determining when to stop recording is more challenging. The basic solution is to check the streaming audio during recording for a period of low sound input lasting longer than a certain threshold (0.5-1secs). This method and our configured threshold worked relatively well for our application because our users typically interact with the LLM using short and clear queries that expect a response instead of longer communications that might have varied silence periods based on pacing or speech cadence.

This method breaks down however when there is loud ambient noise around the user or with certain microphones that automatically gain or lower sensitivity to continually detect sound resulting in a non-consistent "silence" threshold when analyzing the audio stream. To accommodate this some basic audio processing was needed. Since we expected human speech, the input audio was put through a bandpass filter to reduce the electronic noise of sensitive or faulty equipment or background noise frequencies that hide the range of human speech. We also created an algorithm to handle audio calibration every minute to monitor the ambient background noise level to help determine the threshold for what could be considered as silence. This approach works under the assumption that the user will always be closest to the mic and therefore louder than the background noise when using this feature.

While the bandpass filter and calibration is sufficient for our limited use case, it is relatively primitive and unlikely to account for all potential edge cases when deployed to other environments in production. In the future, we plan to evaluate more dedicated processes and implementations for noise suppression, such as those from Microsoft's Deep Noise Suppression Challenge, for inclusion in our application. The noise suppression implementations will be evaluated based on the ability to accurately separate the user's voice from the environment while not adding too much processing overhead or too much delay in recognition of user queries breaking the conversational user experience of the SME.

Text To Speech

There are many available models for performing text to speech, such as Bark (Suno, Inc., 2023), Tortoise (Betker, 2023), OpenVoice (Zengyi Qin, 2023), SpeechT5 (Ao, 2022), XTTS-v2 (Coqui.ai Corp., 2023). Some requirements that we were looking for in this feature are:

- Free Commercial license: The model should be commercially available; thus, MIT and Apache2.0 are preferred.
- Speed: Since the goal is to provide an interactive AI agent, the model cannot take too long to generate audio.
- Voice Cloning: For potential reuse for other applications, it is important to be able to do voice cloning on the model. The complexity and data demand of the cloning process could affect the reusability.

In order to retain the ability to distribute our application to as many users and organizations as possible, models with custom licenses were excluded, such as XTTS-v2 and OpenVoice (which only recently changed to an MIT license).

The Bark model is a generative model that can produce speech resembling reasonably realistic conversations and allows special expressions such as laughing within the prompts. However, this often results in less clear and undesirable audio, such as “ah” or “uh,” not in the original prompt. The model is relatively slow to generate audio.

On the other hand, the speech generated by the Tortoise model is generally clean and sounds like a professional reader reading a textbook. The model is relatively slow by default but comes with pre-packaged code optimized for speed and streaming; therefore, it can start returning audio with very little wait time.

The SpeechT5 model generated audio also sounds like reading but is drier than Tortoise. The model lacks the option for streaming, but it is relatively fast.

Both Bark and SpeechT5 models have a somewhat complex process for cloning voice compared to Tortoise. Based on our experiments with the models, the Tortoise model was the best fit for our purpose. It strikes a balance between quality, speed, and ease of cloning voices, allowing for different or custom-voiced SMEs in the future.

Most text to speech models have difficulty generating long sentences. The most common practice to mitigate this problem is to break the sentence into shorter sentences based on punctuation or semantics. We use the Punkt Sentence Tokenizer from the Natural Language Toolkit to break sentences based on trained semantics, such as distinguishing between a period in “Dr.” and one at the end of a sentence. Combined with some heuristics for breaking sentences based on a comma for longer sentences, this reduced most of the long sentences in our target domain, which are the responses from chat AI.

One problem that cannot be solved with a text to speech model itself is the abbreviations, such as “GPS”, “UH-60M” or “FM”. These abbreviations are context dependent and were not included in most training data. The solution is to convert any word with all uppercase to separated characters so that “GPS” would become “G P S”. Since text to speech does not care about special characters it can be stripped from output. However, the number must retain the continuity, so that “UH-60M” would become “U H 60 M”. While this simple heuristic would handle most of the abbreviations, some abbreviations that include lowercase, such as “No.” for “Number,” would require hard replacement. We used a simple dictionary replacement for this.

BENCHMARKING AND EVALUATION

When evaluating the performance of large language models (LLMs), it was crucial to consider the limitations of public benchmarks and the benefits of using local Knowledge and Skills Assessment (KSA) benchmarks. While public benchmarks provided a standardized way to compare models, they did not always reflect an individual organization’s specific needs, requirements, or use cases.

One issue with public benchmarks is that many have been marred by model developers' over-optimization or gaming. (Deng, Zhao, Tang, Gerstein, & Cohan, 2023) This can lead to inflated performance scores that do not accurately represent a model's true capabilities in real-world applications. As a result, relying solely on public benchmarks may not provide a clear picture of how well an LLM will perform on specific tasks.

To overcome these limitations, using local KSA benchmarks for testing provided better results aligned with our organization's needs. By designing tests that focus on the knowledge and skills relevant to our use cases, we could better understand a model's performance in the context of our specific requirements.

KSA Testing and Results

When developing local KSA benchmarks, it is essential to consider the types of questions and tasks used to evaluate the models. Using elementary questions that require basic reasoning, problem-solving skills and common-sense reasoning can help determine whether a model has the fundamental capabilities needed to be a good fit for your application. These questions should test the model's ability to understand and respond to basic concepts, rather than focusing on highly specialized or complex tasks. Below is a set of five sample questions, from 25 total questions, that were used to assess the performance of several LLMs:

1. My trophy cannot fit into my suitcase because it is too small. From that sentence, which item is too small?
2. House A is to the east of House B, and House B is to the east of House C. Is House C to the east of House A?
3. Sam is older than Bob, and Bob is younger than Tina. Who is the youngest?
4. At a pet shop, there are 12 cats, 5 dogs, and 3 hamsters. If a person buys one dog, how many animals are left in the pet shop?
5. What is the answer? 9/3(2+1)

While seemingly simple, these questions require the models to understand the provided context, extract relevant information, and apply logical reasoning to get the correct answer. Surprisingly, many smaller models struggled to answer these elementary questions correctly, highlighting the limitations of their language understanding and reasoning capabilities.

We conducted two sets of tests: presenting each question individually and presenting all 25 questions simultaneously. While some cloud-based models performed well in answering individual questions, nearly all models struggled or failed when all questions were asked simultaneously. These findings highlight the need for more robust and comprehensive evaluation methods to assess the models' ability to handle complex, multi-step reasoning and maintain consistency across different contexts.

During the testing process, we discovered that priming the models with related messages and using prompts that encourage intelligent, logical, and reasoned responses significantly improve test results. By providing the models with relevant context and guiding them toward a more structured and analytical approach, we were able to produce more accurate and coherent answers, which helped mitigate some of LLMs' limitations, such as their tendency to generate irrelevant or inconsistent responses and can enhance their ability to handle complex reasoning tasks.

LANGUAGE MODEL DETAILS				TEST 1						TEST 2
Context	**TPS**	**Params**	**Name**	**Logic**	**Math A**	**Math B**	**Instruct**	**Reflect**	**Total**	**Total**
16k/32k	50	46.7B	Mixtral-8x7B-Instruct-v0.1.IQ3_XXS	13	6	2	3	y	**24**	16
4k	45	10.7B	Nous-Hermes-2-SOLAR-10.7B.Q8_0	12	6	2	3	y	**23**	11
16k/32k	45	46.7B	Nous-Hermes-2-Mixtral-8x7B-DPO.Q3_K_S	12	6	2	3	y	**23**	14
16k/32k	45	46.7B	dolphin-2.7-mixtral-8x7b.Q3_K_S	12	6	1	3	y	**22**	13
8K/200k	21	34B	dolphin-2.2-yi-34b-200k.Q4_K_M	13	6	2	1	y	**22**	10
16k/200k	25	34B	Smaug-Yi-34B.Q3_K_M	13	6	1	2	y	**22**	10

4k	29	34B	Nous-Hermes-2-Yi-34B.Q4_K_M	12	6	2	2	n	**22**	11
32k	67	7B	NeuralBeagle14-7B.Q8_0	10	6	2	2	y	**20**	6
8K/32k	22	70B	miqu-1-70b-sf.IQ2_XXS	13	3	2	2	y	**20**	14
8K/200k	30	34B	bagel-34b-v0.2.Q4_K_S	12	4	2	1	n	**19**	12
4k	38	28B	laserxtral.Q6_K	8	6	2	3	n	**19**	11
32k	67	7B	CapybaraHermes-2.5-Mistral-7B.Q8_0	11	3	2	3	n	**19**	8
32k	67	7B	WestLake-7B-v2.Q8_0	12	5	2	3	n	**22**	6
8k/200K	20	60B	Mixtral_34Bx2_MoE_60B.IQ2_XS	13	6	0	2	n	**21**	12
32k	67	7B	OmniBeagle-7B.Q8_0	11	6	2	2	n	**21**	8
32k	67	7B	OpenHermes-2.5-Mistral-7B.Q8_0	8	3	1	2	y	**14**	9
32k	67	7B	Mistral-7B-Instruct-v0.2.Q8_0	13	0	0	2	n	**15**	10
8K/200k	30	34B	Nous-Capybara-34B.Q4_K_S	12	3	1	0	n	**16**	5
			Maximum Scores	**14**	**6**	**2**	**3**	**y/n**	**25**	**25**

Table 1: LLM Performance Metrics Across Testing Scenarios

Legend for LLM Performance Comparison Table

- **Context:** Represents the context window size in used/total tokens, indicating the model's input capacity.
- **TPS** (Tokens Per Second): Measures the model's processing speed.
- **Params:** Denotes the number of trainable parameters in the model, typically expressed in billions (B).
- **Name:** Identifies the specific language model variant under evaluation.
- **Logic:** Quantifies performance on cognitive reasoning and logical thinking tasks.
- **Math A:** Assesses capability in basic arithmetic operations.
- **Math B:** Evaluates proficiency in advanced arithmetic problems.
- **Instruct:** Measures the model's ability to accurately follow complex instructions.
- **Reflect:** Indicates the model's capacity for error recognition and self-correction (y = yes, n = no).
- **TEST 1:** Results from individual questions, one question per prompt.
- **TEST 2:** Results from all questions asked in a single prompt.

Limitations of LLMs

Computational Requirements: LLMs are complex models requiring significant computational power for development, training, and deployment. This is because these models need to process and learn from vast amounts of data, which can be computationally intensive. The cost of the necessary hardware and the energy required to run these computations can be substantial. Furthermore, the availability of computational resources can limit the size and complexity of the models that can be developed, which in turn can impact their performance in real-world applications.

Counting Limitation: Despite their advanced capabilities, one fundamental limitation of LLMs is their inability to count accurately. This limitation becomes evident when LLMs are tasked with precise enumeration or ordering items based on specific criteria. For instance, if an LLM is asked to "List all 50 States from the USA, ordered by the number of letters in each state, ignoring the spaces," it would struggle to provide a correct answer. LLMs typically do not have a built-in mechanism for counting or understanding numerical relationships like humans do. This limitation is not exclusive to any LLM, whether cloud-based, open-sourced, or local, and is independent of their size or training methodology.

Contextual Understanding: While LLMs have made significant strides in processing and generating human-like text, they may still need a greater understanding of the context and deeper meaning of the language they process. This limitation can manifest in various ways, such as misinterpreting sarcasm, failing to grasp cultural nuances, or misunderstanding complex metaphors. As a result, LLMs may sometimes provide technically correct responses but contextually inappropriate or irrelevant. This limitation highlights the gap between processing language at a surface level and truly comprehending its underlying meaning and intent, which humans naturally excel at.

Generating Misinformation: One of the more concerning limitations of LLMs is their potential to generate incorrect or misleading information. This issue stems from the fact that LLMs are trained on large amounts of data from the internet, which can include inaccurate or biased information. When generating responses, LLMs may confidently present this misinformation as fact, potentially spreading false or harmful information. This limitation underscores the importance of using LLMs responsibly and implementing safeguards to verify the accuracy of generated content. RAG techniques can help mitigate this issue by grounding the model's responses in verified, up-to-date information from curated knowledge bases.

Lack of Creativity: While LLMs can generate text that appears creative and original, it is important to recognize that they do not possess true creativity in the human sense. These models recombine and repurpose patterns and information from their training data in novel ways, but they cannot innovate or create entirely new concepts. LLMs do not have personal experiences, emotions, or consciousness, which are often the wellspring of human creativity. As a result, while LLMs can be powerful tools for assisting in creative tasks, they cannot replace human creativity and originality. This limitation becomes particularly evident in tasks requiring deep emotional resonance, genuinely novel ideas, or solutions that require thinking far outside the box of existing knowledge.

Lack of Common-Sense Reasoning

Despite their impressive ability to process and generate human-like text, large language models (LLMs) often struggle with common-sense reasoning that humans find intuitive. This limitation becomes particularly evident when LLMs face scenarios that require a basic understanding of the real world.

Common sense reasoning involves making logical inferences based on general knowledge about the world and everyday experiences. Humans develop this ability naturally through their interactions with the environment and society. However, LLMs, trained primarily on textual data, lack the embodied experience and real-world interactions that form the basis of human common sense.

This limitation can manifest in various ways:

1. Logical Inconsistencies: LLMs may generate responses that, while grammatically correct, are logically absurd or physically impossible. For example, an LLM might suggest that a person could "drive a car to the moon" or "boil water to make it colder."
2. Misunderstanding of Basic Physics: LLMs often struggle with simple physical concepts humans intuitively understand. They might fail to recognize that objects fall due to gravity or that liquids take the shape of their containers.
3. Temporal and Causal Reasoning: LLMs can have difficulty understanding the order of events or cause-and-effect relationships. They might suggest impossible sequences of actions or fail to recognize that certain events must precede others.
4. Lack of Understanding of Human Needs and Behaviors: While LLMs can process text about human experiences, they do not truly understand human needs, emotions, or typical behaviors. This can lead to suggestions or responses that are inappropriate or unrealistic in real-world social contexts.
5. Difficulty with Analogical Reasoning: LLMs may need help to apply knowledge from one domain to another in the way humans naturally do. This limits their ability to solve problems creatively or understand metaphors and analogies.
6. Inability to Distinguish Between Common and Rare Events: LLMs may treat unlikely or rare events with the same probability as common occurrences, leading to unrealistic or improbable output scenarios.

This lack of common-sense reasoning highlights the gap between current AI capabilities and human-like intelligence. While LLMs can process and generate text based on patterns in their training data, they lack the fundamental understanding of the world that humans develop through lived experience. Addressing this limitation remains a significant challenge in artificial intelligence and is crucial for developing more robust and reliable AI systems that can interact meaningfully in real-world scenarios.

Local Deployment with Optimized Resources

In our exploration of local LLM deployment, we utilized the llama.cpp library to create a server endpoint for RAG and chatbot communication. This allowed us to quickly develop and test solutions within our local area network, providing a seamless and efficient development environment. One key finding is that local models can work effectively in most cases using a machine with 64-128GB of memory and a high-performance GPU, such as the NVIDIA GeForce RTX 4090. This configuration provides sufficient computational resources to handle the demands of LLMs while still being accessible to a broader range of users and organizations compared to large-scale, cloud-based infrastructure. The combination of adequate memory and a powerful GPU enables efficient training and inference of LLMs, allowing for faster iteration and experimentation. The RTX 4090, with its 24GB of GDDR6X memory and 10,752 CUDA cores, offers significant performance improvements over previous generations, making it well-suited for handling the complex computations required by LLMs.

By leveraging llama.cpp, we were able to harness the power of LLMs and RAG without relying on external cloud-based services, ensuring data privacy and reducing latency. The library's efficient use of CUDA cores made the responses extremely fast, further enhancing the performance and responsiveness of our local deployment. This localized approach streamlined our development process and enabled us to iterate rapidly, ultimately leading to more effective and tailored solutions for our specific use cases.

FUTURE RESEARCH

Improving the knowledge of our interactive agents is likely to be a long-term process involving cultivating and refining the data sets that the RAG server and LLM have access to, followed by reinforcement training as we interact with the agents to correct responses deemed low quality or incorrect. We also would like to implement a way for actual subject matter experts to correct responses while testing our agents to continually train the LLM and perhaps add a library of known questions and correct responses that their students commonly ask during live instruction after deployment. One way to enhance the agent's capabilities is to improve its interaction with the environment and the on-screen courseware. This could be achieved by better modeling the virtual environment. The improved model would then give more detailed information to the Language Learning Model (LLM) when prompted. Additionally, it is important to develop a method that allows the LLM to give commands directly to the agent. These commands should enable the agent to interact with the course content as it interacts with 3D model components.

Additional high priority tasks exist for the LLM. One task is for the LLM to understand images and documents shown to the user during training. This is crucial in case a student asks a question about the information on screen that is not text-based. Another task is for the agent to guide the user to relevant past or future content. This helps when a user asks a question, enabling the agent to supplement the response with appropriate course material. All these improvements will require updating the interface to pass application context to the agent in a generic way suitable for many virtual environments and instruction types.

The user experience of the virtual SME is another area that could be refined. UX studies and data collection should be performed to make data-driven improvements to the interaction with the virtual SME. Additionally, we want to provide several ways to integrate our virtual SME into training content. For example, including a 2D avatar that can optionally replace our 3D avatar when content does not have a 3D environment or where an instructor is present in the environment would take away from the instruction. Quality of life and fidelity improvements could be made to the avatars to improve immersion. For example, high-quality animations for avatars can be included, or integrating new technologies for AI-driven procedural character animations like those in development by NVIDIA **(Burnes, 2023)**. Improvements are needed in speaking to our interactive agents as outlined in the text to speech and speech recognition sections above to ensure our keyword and speech recognition solutions hold up under most environments and hardware configurations to reduce failure to capture or erroneous capture outcomes when processing user speech.

We want to improve the fidelity of our agents' voice output to more closely resemble natural speech and better handle long-form, highly technical responses. We plan to continue evaluating new models and technologies that might be

better suited to our use case than our currently selected implementations. We plan to refine our test and evaluation processes as the experience our end users desire becomes more defined to ensure our choices remain the best suited.

Lastly, additional testing remains to push the prototype's capabilities and scalability for deployment environments by using different hardware infrastructure. The original study was confined to staying fully local hosted and only high-end consumer gaming hardware, which is how we arrived at RTX 4090 graphics cards. We want to investigate how far we can push our virtual SMEs' capabilities, interactivity, and intelligence by leveraging cloud hardware infrastructure to see what is in the realm of possible when our current technical approach is not hardware constrained and to better investigate optimization options. We also would like to move towards specialized local hardware that could be provided as a local server to a network when deploying the SME allowing us to still be a secure and local deployment with less reliance on consumer hardware and a single machine. This will enable us to improve the resource and context size our agents can access giving our agents the ability to have a broader knowledge set and therefore hopefully the capability of supplementing much larger courses, in addition to interacting with multiple users simultaneously. The agents would be required to stay within the local deployment limits as required by customer requirements. The ability to connect to sensitive data sets not available over a public network is still possible with this approach.

CONCLUSION

Following the technical approach described in this paper, we created an immersive interactive SME that could be inserted within a virtual IMI environment to supplement self-paced IMI. The virtual SME can respond to prompts from the user and interact with the virtual environment in a limited capacity, increasing user immersion and making virtual training of any IMI level more interactive and potentially less linear, regardless of the lesson structure and content. The technical approach is still a proof of concept at this stage and further research and work are needed to improve the capabilities and knowledge of the interactive agents, validate and improve the user experience of interacting with the virtual SME, and identify what kind of optimizations and hardware configurations would be needed in order to provide the experience in a production environment with a sufficient number of users.

In summary, the described approach is a firm starting point for incorporating a virtual SME into an immersive IMI lesson, especially when internet-required connections to an LLM are not possible. LLMs continue to evolve and improve, and the use of RAG allows an LLM to recall specific information (e.g., technical manuals, training slides) not available as part of the trained model itself. When coupled with user input mechanisms such as speech recognition, users are allowed to naturally ask questions to the LLM like they would a real SME. The approach described in this paper was performed by a very small team with limited resources. The requirements for adopting the described approach are expertise in machine learning and software development and a modest hardware budget. The authors acknowledge a virtual SME will never replace the quality and experiences of a real-life SME. We envision a virtual SME as an augmentation to training curriculums for when real life SMEs are in short supply or unavailable.

REFERENCES

Ao, J. a. (2022). *SpeechT5: Unified-Modal Encoder-Decoder Pre-Training for Spoken Language Processing.* arvix. doi:10.48550/arXiv.2110.07205

Berti, A., Kourani, H., Hafke, H., Li, C., & Schuster, D. (2024). Evaluating Large Language Models in Process Mining: Capabilities, Benchmarks, Evaluation Strategies, and Future Challenges. *ArXiv*.

Betker, J. (2023). *Better Speech Synthesis Through Scaling.* arvix. doi:10.48550/arXiv.2305.07243

Burnes, A. (2023, 05 28). *Introducing NVIDIA ACE For Games - Spark Life Into Virtual Characters With Generative AI*. Retrieved from NVIDIA: https://www.nvidia.com/en-us/geforce/news/nvidia-ace-for-games-generative-ai-npcs/

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., . . . Xie, X. (2023). A Survey on Evaluation of Large Language Models. *ArXiv. /abs/2307.03109*.

Coqui.ai Corp. (2023). *Coqui/XTTS-2*. Retrieved from Hugging Face: https://huggingface.co/coqui/XTTS-v2

Deng, C., Zhao, Y., Tang, X., Gerstein, M., & Cohan, A. (2023). *Investigating Data Contamination in Modern Benchmarks for Large Language Models.* ArXiv. doi:10.48550/arXiv.2311.09783

Graves, T. R., Blankenbeckler, P. N., Wampler, R. L., & Roberts, A. (1996). *A Comparison of Interactive Multimedia Instruction.* United States Army Research Institute.

Jahan, I., Laskar, M. T., Peng, C., & Huang, J. (2024). A Comprehensive Evaluation of Large Language Models on Benchmark Biomedical Text Processing Tasks. *arXiv*.

Liu, Y., Fabbri, R., A., Chen, J., . . . A. (2023). Benchmarking Generation and Evaluation Capabilities of Large Language Models for Instruction Controllable Summarization. *arXiv*.

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., . . . Mian, A. (2024). A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

Nelson, T. (2021, 3 12). *Interactive Multimedia Instruction: state-of-the-art approach to training the next generation of Ordnance Soldiers*. Retrieved from Army.mil: https://www.army.mil/article/242994/interactive_multimedia_instruction_state_of_the_art_approach_to_training_the_next_generation_of_ordnance_soldiers

Novelly, T. (2023, 5 12). *Air Force Scrambles to Address Officer Training School Instructor Shortage*. Retrieved from Military.com: https://www.military.com/daily-news/2023/05/12/air-force-scrambles-address-officer-training-school-instructor-shortage.html

Patil, R., & Gudivada, V. (2024). A Review of Current Trends, Techniques, and Challenges in Large Language Models (LLMs. *Applied Sciences*.

Radford, A. a. (2022). *Robust Speech Recognition via Large-Scale Weak Supervision.* arXiv. doi:10.48550/ARXIV.2212.04356

Suno, Inc. (2023). *Suno/Bark*. Retrieved from Hugging Face: https://huggingface.co/suno/bark

US Department of Defense. (2001). *MIL-HDBK-29612-3: DEVELOPMENT OF INTERACTIVE MULTIMEDIA INSTRUCTION (IMI)* .

Zengyi Qin, W. Z. (2023). *OpenVoice: Versatile Instant Voice Cloning.* arvix. doi:10.48550/arXiv.2312.01479