VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

Analysis and Development of Artificial Intelligence Applications for Generating Educational Material

Análisis y desarrollo de aplicaciones de inteligencia artificial para generación de materiales didácticos

Isidro Salvador Aguirre-Cervantes[1], Karla Ximena Ávalos-García[1], Emily Orozco-Espinoza[1], Rosa Atenea Hedoné Rangel-Quiroz[1], Silvia Alejandra López-Juárez[1], Luis Carlos Padierna-García[1]
[1]División de Ciencia e Ingenierías, Universidad de Guanajuato, Campus León
is.aguirrecervantes@ugto.mx, kx.avalosgarcia@ugto.mx, e.orozcoespinoza@gmail.com, rah.rangelquiroz@ugto.mx, lopez.sa@ugto.mx , lc.padierna@ugto.mx

Abstract

The deployment of generative artificial intelligence systems accessible to everyone is driving extensive reforms in the educational environment, implying a continuous improvement in the development of tools that facilitate their integration into various learning processes. The following article presents the development of software designed for the automated creation of teaching materials. The system is composed of a graphical interface built with CustomTkinter and a multi-agent system based on the router pattern strategy using the GPT-4o-mini model, a text-to-speech conversion module using the open-source library Coqui TTS, and the analysis of lip-sync models such as Wav2Lip and SadTalker. The results obtained demonstrate the operational viability of the system, achieving an efficient delegation of tasks and responses for the user. Furthermore, the viability of the process was validated through a case study on control structures in the C language, yielding multimedia teaching material. It is concluded that the development of software coupling multi-agent systems and open-source models can be used to democratize the production of educational content, potentially overcoming commercial cost barriers associated with the generation of teaching materials

Key words: Generative Artificial Intelligence, Multi-Agent System, Speech Synthesis, Lip-Sync, Teaching Materials, Education

Introduction

The world is undergoing significant changes driven by the exponential development of artificial intelligence in recent years, an impact that has deeply resonated in education. According to the National Survey on the Uses and Perceptions of Generative Artificial Intelligence (ENIAG 2025) in Mexico, 66% of students and 60% of teachers report using generative AI (GAI) daily to generate text, images, code, video, and audio (Secretaría de Educación Pública, 2026). Furthermore, 7 out of 10 students and teachers perceive improvements in their performance when using GAI. Despite these benefits, none of the surveyed groups reported mastering these tools (Secretaría de Educación Pública, 2026). Considering these advancements is essential for guiding and designing modern education, allowing for regulations and integration strategies that truly benefit teaching and learning processes without exacerbating existing challenges. We focus on this endeavor through the research and development of software that allows users to create both textual and audiovisual academic content—such as short videos explaining concepts from various fields, curriculum design, among others.

To achieve the generation of educational videos, we proposed a workflow with the following components:

1. A graphical user interface (GUI).
2. Large Language Models (LLMs) and agents to retrieve the required intellectual content based on user requests.
3. Text-to-Speech (TTS) software to generate audio from the text provided by the LLM.
4. Software for avatar production and lip-syncing between the generated audio and the avatar.

VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

Regarding LLMs, ChatGPT, Gemini, Grok, and Claude are based on Transformer architecture, well-known and robust in reasoning across various fields of knowledge, including exact and experimental sciences, as well as engineering (Ngomseu Mambou & Esenogho, 2026). However, access to them is charged based on token usage, and their free plans are often insufficient for extensive or complex queries, making them inaccessible for this project. Nonetheless, efficient commercial options also exist, such as OpenAI's GPT-4o-mini model, which can be integrated into a multi-agent system utilizing a router pattern strategy for resource and cost optimization (OpenAI, 2024).

Similarly, for the TTS task, various software options are available. There are considerably complete and useful TTS conversion APIs available, such as Google Cloud TTS and Azure TTS, however, once again, their price and restrictions make them unappealing to the project. In contrast, Coqui TTS is an open-source library for advanced text-to-speech generation in numerous languages, featuring extensive documentation, which makes it a valuable software tool for this project (Coqui AI, n.d.-a).

Finally, for avatar production and lip-syncing, tools such as D-ID exist, specializing in generating videos from images, audio, different configurations of these, or prompts (D-ID, n.d.). All these options are heavily restricted regarding external use and API integration, and their costs are based on video duration and features; therefore, they are not useful for the pipeline. SadTalker and Wav2Lip are open-source alternatives that generate realistic videos from an audio file and an image of an avatar or person, performing lip-syncing and adding facial expressions (Rudrabha, n.d.; Zhang, n.d.).

Materials and methods

For video generation and overall educational content, agents, LLM, the GUI, TTS and lip-syncing software were used in a specific order. A schematic workflow of the process is presented in Figure 1. Relevantly, the first two components of said workflow suffice for the creation of non-audiovisual educational content (lesson plans, explanations, etcetera).

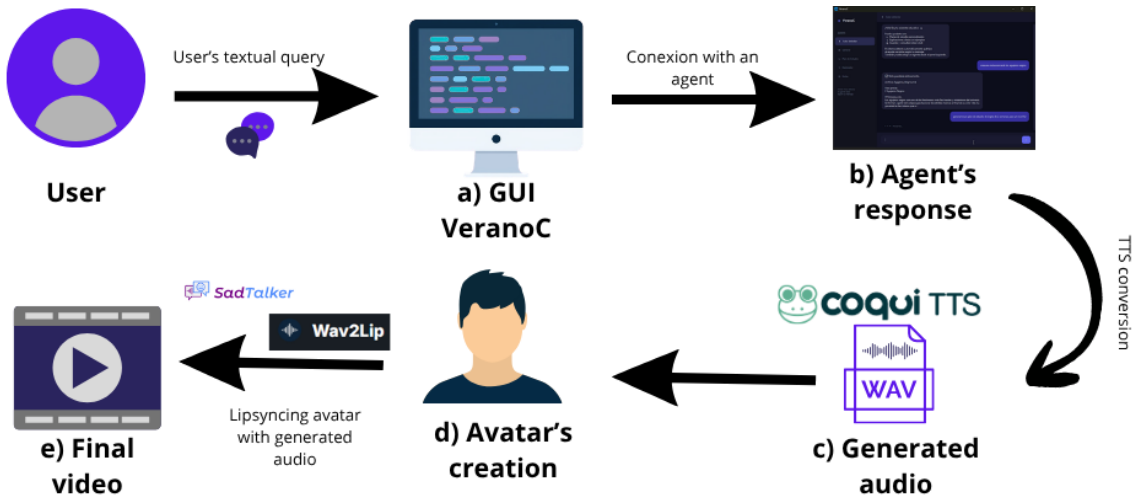


Figure 1. Workflow for creation of educational videos and other content in VeranoC.

a) Graphical user interface VeranoC

The graphical user interface (GUI), named VeranoC, is built using customTkinter, a library that extends the Tkinter graphics engine to generate modern interfaces with minimalist themes, advanced tools, and customizable color palettes. Tkinter by itself is the Python interface oriented toward the Tk graphics engine. It can create an operating system window and manage all events related to peripherals, such as the keyboard, mouse, etc. On the other hand, CustomTkinter is essentially a layer that runs on top of Tkinter, replacing the

system's native widgets with widgets rendered on Tk canvases providing support for rounded corners and RGBA colors (CustomTkinter, 2024).

The main block of the interface is AppWindow, which acts as the root container of the application. Through this component, the title, initial dimensions, and background color are configured. It instantiates the Sidebar and ChatPanel widgets, which contain the chat and the side panel, respectively. The side panel or Sidebar is a side panel that occupies the left side of the window and consists of three blocks: the logo or title, the buttons to select the agents and finally, an informative note. The ChatPanel is a component that is divided into three sections. The first acts as the top bar displaying the active agent. The second block contains a scrollable central area to read the conversation history. The final block contains the user input bar, which is a text field that receives the user's text and a button to send the message. Furthermore, while the thought process generated by the orchestrator or any specific agent is searching for a response, the interface generates a label to display the 'thinking' legend.

The development of this interface was supported by several libraries, specifically utilizing versions such as OpenAI 1.30, CustomTkinter 5.2, and Python-dotenv 1.0 or higher.

b) Large language model

The LLM utilized for the chat component is the GPT-4o-mini model, belonging to OpenAI's GPT-4 family. This model was selected because it is optimized for reasoning and response tasks while maintaining low computational costs and reduced latency, which enables rapid dialogue (OpenAI, 2024a). The interaction between the model and Python was implemented via the OpenAI Chat Completions API. By protocol, this API defines three roles within the conversation: *System*, which provides the initial instruction or prompt that defines the agent's behavior; *User*, representing the message sent by the user; and *Assistant*, which is the response generated by the model (OpenAI, 2024b).

The software architecture is primarily constituted by a Multi-Agent System, an AI paradigm wherein multiple autonomous entities, known as agents, reason and act in an organized manner to achieve a specific objective (Wooldridge, 2009). An AI agent is defined as a module that possesses an identity, such as a name, description, etc., has a specific goal determined by its initial prompt, and can execute actions such as reading and writing files, thereby extending beyond simple text generation.

The router pattern is a strategy in which an orchestrator or central component receives user-generated requests to subsequently delegate them to a specialized agent tailored to the task. The software features a multi-agent system based on the router pattern (also known as an orchestrator-agents architecture). Through this strategy, stability in the executed tasks is maintained, allowing for a higher degree of methodological repeatability and a conversation history bank. This enables agents to share the conversation context, and furthermore, ensures that adding new agents to the architecture does not modify the logic of existing ones.

Once the user sends a message, it is received by the orchestrator, which constructs a response using the LLM, the user's message, and the previous message history. It was decided to limit the message history to 20 messages, as a larger context generates higher computational costs in the system. The response generated by the orchestrator agent is constrained to return only the name of one of the 4 agents in the system; this restriction is enforced through the agent's initial prompt within its system configuration. The selected agent is then invoked along with the history and the message, enabling it to generate a response based on its own specialized prompt.

c) Coqui TTS

For the text-to-audio conversion, the open-source Python library Coqui TTS was employed. Within this library, various audio synthesis models are available, all of which can be instantiated via the Coqui TTS API and feature simple commands to convert short text fragments into audio files of a few seconds. The VITS model was selected for this project due to its ease of use. VITS, or 'Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech', is a TTS model that converts text directly into audio waveforms in a single step (Coqui AI, n.d.-b). It is composed of five main neural networks: 1) a text encoder that transforms text into phonemes, 2) a stochastic duration predictor that computes the duration of each syllable, 3) a Variational Autoencoder (VAE) that defines rhythm, pauses, and intonation, 4) a normalizing flow

that introduces voice variations, and 5) a discriminator to implement adversarial learning (Kim et al., 2021). Specifically, the version of VITS trained on the CSS10 dataset was utilized, which consists of numerous short fragments extracted from hours of audiobooks narrated by a single Spanish-speaking speaker (Park, n.d.).

The model implementation was done in Google Colab to gain access to computing resources that accelerate execution; however, TTS conversion can also be performed locally, resulting only in variations in execution times depending on the equipment used. The steps followed were:

a) Installation of the TTS library and dependency resolution, given that Coqui models are supported by Python versions greater than or equal to 3.9 and less than 3.12.
b) Definition of the path for the input text to be converted and the path for the resulting output audio file.
c) Instantiation of the model as a TTS object, specifying the model type and its version (the dataset utilized during training).
d) Synthesis of the audio using the tts_to_file method.

In this manner, an audio file is generated containing the provided text, featuring the narrator's voice corresponding to the dataset specified during model instantiation. Furthermore, the execution time using the CPU provided by the free tier of Colab was within a range of dozens of seconds, including library installation and dependency updates.

d, e) Avatar creation and lip-syncing

For the current phase of this project, avatar creation and lip-syncing tests were not conducted; nonetheless, the models considered for this task, which would complete the proposed workflow for educational video generation, are highlighted. The objective is to implement a module that generates an avatar image and animates it using dedicated lip-sync models, driven by the audio generated with Coqui TTS from the text retrieved as responses from the agents. In this regard, Wav2Lip and SadTalker are contemplated. The former is a model specialized in lip-syncing on pre-existing faces; that is, it must be provided with the face image to be used, and it exclusively modifies the mouth region to synchronize it with the given audio (Prajwal et al., 2020). The second model is noteworthy for its capability to perform not only lip-syncing based on the provided audio but also to generate facial movements driven by the audio from a static image; this feature adds to the realism of the videos (Zhang, n.d.). It is relevant to mention that both are open-source projects that can be executed locally and have public repositories on GitHub (Rudrabha, n.d.; Zhang, n.d.), giving them an advantage over commercial options currently available on the market, which incur high costs and do not offer flexibility for integration into custom workflows. Its implementation will be similar to the one discussed for the TTS conversion: it will require providing the paths for the base image to be used and the audio generated by Coqui TTS, followed by the instantiation of the models, and finally the generation of the video, which will be returned in MP4 format.

Results

Graphical user interface VeranoC, LLM and agents

The graphical user interface VeranoC was developed using the tools described in the previous section. It was constructed with five main sections arranged within a menu bar: *Auto-detect*, *General*, *Curriculum*, *Explainer*, and *Notes*. Within each section, user interaction with the orchestrator and agents is conducted through text prompts in which the user specifies their request. In the *Auto-detect* mode, the ideal agent is automatically assigned based on the request. The names of the other four sections or modes explicitly indicate the orientation of the task performed by their underlying agent. Specific base styles or prompts were assigned to each mode, ensuring that the responses provided by the respective agents maintain a consistent and organized structure for similar requests; for instance, in the *Explainer* section, the agent is designed to consistently deliver a definition, an analogy, and a practical example tailored to the concept.

VeranoC features a modern and sober appearance, employing blue and purple tones, and comprises only two main visual sections: 1) a vertical sidebar containing the available modes or agents alongside the application's name, and 2) a panel occupying the remainder of the screen where the interaction with the agent takes place in a chat format. This structure and some examples of interaction within the *Auto-detect* mode can be observed

in Figure 2. Overall, the interface and system operations, up to the incorporation of the agents, are simple and intuitive, making them desirable for educational applications targeted at a general audience not necessarily specialized in AI or software.

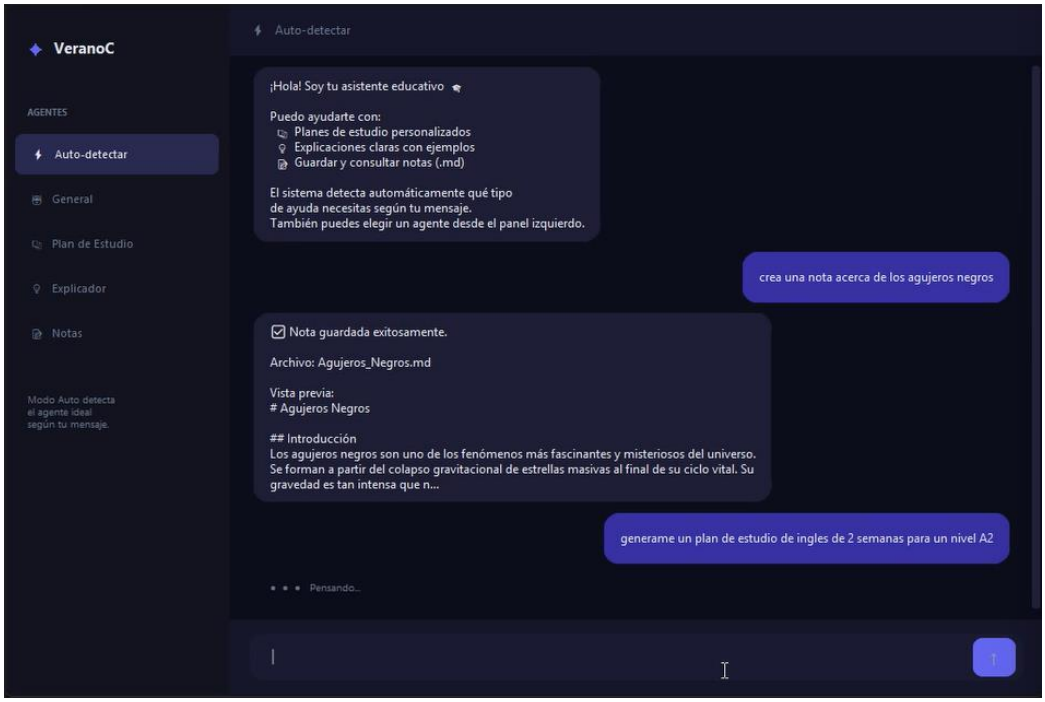


Figure 2. Graphic interface VeranoC.

Text-to-audio conversion tests

By utilizing the Coqui TTS library in Google Colab and following the implementation described in the Materials and Methods section, it was possible to generate a 6-second audio file. The text employed was *"Hola, esta es una prueba de material didáctico para la materia de Programación Básica."*, whose length was deliberately chosen to be short to prevent model hallucinations. Qualitatively, the audio corresponds to the voice of an adult male with a noticeable tone of tranquility and seriousness, featuring natural pauses, pronunciation, and intonation. This confirmed that the VITS model from Coqui TTS, trained on the Spanish CSS10 dataset, is viable for short audio synthesis, thereby allowing the project to bypass paid APIs. Furthermore, based on the qualitative evaluation of the audio, the default voice and overall tones produced by VITS are appropriate for creating educational content. Another advantage is the execution time of just a few seconds if the dependencies are already installed, preventing excessively long waiting times for the user. Additionally, Coqui TTS features a voice-cloning option, meaning that the capability to use audio voluntarily provided by professors for TTS conversion can be included in the future. Finally, for the future utilization of the model to generate long audios corresponding to more extensive texts, the text can simply be manipulated via Python code to fragment it into short, manageable portions for the model; this would involve feeding one text fragment at a time into the model and concatenating the resulting short audio files.

Regarding the lip-sync module, it should be noted that, although the implementation process is straightforward in principle, the complexity lies in the hardware requirements, the necessary software versions that are generally incompatible with those demanded for Coqui TTS, and the final pipeline integration of all elements (GUI, agents, TTS, and lip-syncing). Models such as SadTalker require a considerable amount of VRAM; therefore, it is worth considering the acquisition of equipment with sufficient GPU VRAM or the deployment of Google Colab virtual machines. Furthermore, it is necessary to equip the system's orchestrator with the ability to perform tasks asynchronously and to add waiting messages during video generation, given that execution

and generation times will be considerable. Thus, for subsequent advancements and improvements to the project, the lip-sync module must be integrated into the pipeline, and the connection of all managed APIs and models must be established. The ultimate goal is to generate software through which the user only needs to provide prompts in the GUI to obtain a variety of educational content within an appropriate timeframe.

Implementation and Practical Demonstration (Didactic Sample)

To validate the viability of the audiovisual pipeline focused on teaching exact sciences and engineering, a practical case study was conducted centering on the *for* loop control structure in the C programming language. For this validation, commercial tools were utilized, which can be deployed provisionally while the open-source software-based project is finalized. Due to quota optimization criteria and the fixed-frame design required for the animation of digital avatars, the sample processing was successfully resolved through the following development phases:

- **Isolation and Preparation of the Base Material:** A conceptual explanatory video generated via Google's NotebookLM tool was used. The soundtrack was extracted in MP3 format with optimal fidelity using the *Movavi* tool, while the original video was completely muted using the *InShot* mobile application to prevent audio duplication during the editing phase.
- **Segmentation and Structural Synchronization:** The video script was divided to match the animation precisely, so that each static visual frame (containing the avatar positions and the graphical aids for the *for* loop) was linked to a specific acoustic segment or script. To achieve this, the general audio file was meticulously fragmented using *Online MP3 Cutter*, and the video was edited in *InShot* respecting the timing of each block. The D-ID avatar generation platform automatically adjusted the video clipping based solely on the duration of the last audio track belonging to the final frame, ignoring the length of the final video fragment.
- **Transfer Optimization:** To meet the required storage standards without degrading the signal, the processed audio was compressed using *Free Compress*, successfully reducing the file size below the critical 15 MB limit while preserving the acoustic clarity essential for lip-sync algorithms.
- **Final Sample Results:** The integration of these components yielded a fully functional audiovisual educational material with an optimized final duration of 2 minutes and 45 seconds. This sample demonstrated that intellectual content generated by language models can be clearly transferred into interactive multimedia formats, facilitating the comprehension of complex logical programming concepts through high-fidelity virtual avatars.

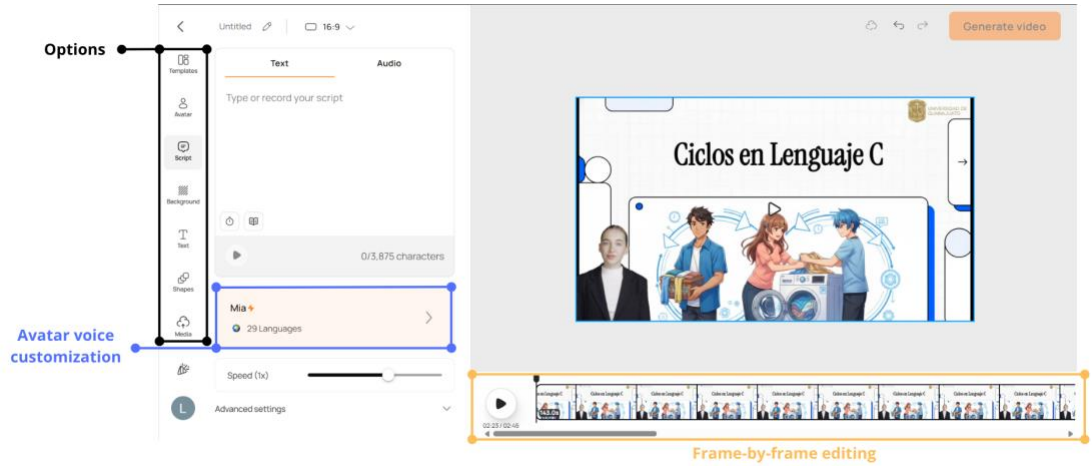


Figure 3. Components of the D-ID interface used for avatar video generation.

Conclusions

The VeranoC user interface was fully produced and connected to a multi-agent system, featuring essential visual and interactive characteristics for its use in generating educational material. The interface is aesthetically appealing while prioritizing simplicity and accessibility, avoiding an excessive number of sections or the inclusion of unintuitive options. Furthermore, the functions currently provided cover important aspects of educational needs for both teachers and students, ranging from the illustrative explanation of concepts to the generation of personalized study plans. Likewise, execution tests validated the viability and solidity of the software. The use of the multi-agent architecture made the software more efficient by coordinating the appropriate agent with a given task. Similarly, the connection to the GPT-4o-mini model demonstrated low latency in obtaining responses, allowing for a short wait time for the user. Additionally, restricting the system memory had a beneficial effect on maintaining coherence in the educational texts and reducing computational cost.

On the other hand, the tests conducted regarding TTS conversion are relevant for future advancements and enhancements of this project, making it clear that the open-source library Coqui TTS is viable and valuable to implement for adding explanatory video generation to VeranoC. In addition, there is room for audio modifications and customizations, ranging from the production of long audio files to the use of teachers' voices, creating a sense of familiarity for students that aids in learning. Similarly, the research and exploration carried out on tools for avatar creation and lip-syncing will serve as the foundation for the final elements in audiovisual content production, featuring varied software with different approaches through which realistic and engaging videos can be achieved for the user. In this way, the path is cleared for the future development of this project with the addition of more options to assist and complement modern learning.

Finally, the realization, testing, and detailed documentation of the creation of short and long educational videos expanded upon in this article is considered useful in the current technological and academic landscape, fostering the beneficial incorporation of available digital tools and contributing to the illustration of a potentially entertaining and free way to unite technology, AI, and education.

Study limitations

Developing and integrating the audiovisual educational production pipeline faced critical methodological, operational, and technical constraints due to tool interoperability and platform usage policies (D-ID, n.d.). In this section, these difficulties are discussed as well as the actions taken to work around them and be able to produce the final video.

1. **Credit and quota restrictions in D-ID:** The deployment was constrained by the monetization model of the D-ID API during its free trial. Charging one credit per 15-second interval (rounded up), generating a 3-minute video requires 13 credits; however, free accounts offer only 12. This discrepancy forced a methodological adjustment, capping the final video length at 2 minutes and 45 seconds. Additionally, audio files were restricted to a maximum size of 15 MB.
2. **Preprocessing and fragmentation complexity:** Token barriers and accessibility limits prevented native end-to-end API automation, requiring a fragmented manual workflow to process the base material (a NotebookLM-generated video on C-language *for* loops):
    - **Extraction and Acoustic Isolation:** To prevent audio duplication, the original MP3 audio was extracted using *Movavi*, while the video track was stripped using *InShot*.
    - **Frame Segmentation and Synchronization:** Editing a digital avatar requires its position and any added elements to remain fixed in specific frames, requiring an exact correspondence of one frame for each script or audio segment. This forced the manual splitting of the audio using the *Online MP3 Cutter* web tool and cutting the base video in *InShot* to match the exact duration of the videos with their respective audio fragments.

Although D-ID automatically crops the video if it exceeds the duration of the audio track in the last segment, the lack of prior automated synchronization represented a considerable technical workload.

- **Quality Loss in Audio Compression:** Complying with the 15 MB limit revealed that standard online audio compressors introduce digital noise and distortion, degrading lip-sync accuracy. Only Free Compress preserved sufficient acoustic fidelity.

Supplementary Material

The resulting audiovisual educational material can be accessed at the following link:

Figure 4. QR code linking to the practical demonstration of the generated educational material

References

Coqui AI. (s.f.-a). *Inference* [Documentación de software]. Coqui TTS Docs. https://docs.coqui.ai/en/latest/inference.html

Coqui AI. (s.f.-b). *VITS* [Documentación de software]. Coqui TTS Docs. https://docs.coqui.ai/en/latest/models/vits.html

CustomTkinter. (2024). Documentation. https://customtkinter.tomschimansky.com/

D-ID. (s.f.). *D-ID: Digital human & AI video generation*. https://www.d-id.com/

D-ID. (s.f.). Frequently asked questions. D-ID. https://www.d-id.com/faqs/

D-ID. (s.f.). *Pricing plans*. D-ID. https://www.d-id.com/pricing/studio/

Google Cloud. (s.f.). *Text-to-Speech: Conversión de texto a voz con IA*. https://cloud.google.com/text-to-speech

Kim, J., Kong, J., & Son, J. (2021). *Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech* (arXiv:2106.06103). arXiv. https://arxiv.org/pdf/2106.06103

Ngomseu Mambou, E., & Esenogho, E. (2026). *A comparative analysis of leading AI chatbots: Benchmarking ChatGPT, DeepSeek, Gemini, Grok, and Claude AI*. University of South Africa. https://www.researchgate.net/publication/400490339

OpenAI. (2024a). *GPT-4o mini: Advancing cost-efficient intelligence*. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

OpenAI. (2024b). Completions API. https://platform.openai.com/docs/api-reference/chat

Park, K. (s.f.). *Spanish single speaker speech dataset* [Conjunto de datos]. Kaggle. https://www.kaggle.com/datasets/bryanpark/spanish-single-speaker-speech-dataset

Prajwal, K. R., Mukhopadhyay, R., Namboodiri, V. P., & Jawahar, C. V. (2020). *A lip sync expert is all you need for speech to lip generation in the wild* (arXiv:2008.10010). arXiv. https://arxiv.org/abs/2008.10010

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022). *Robust speech recognition via large-scale weak supervision* (arXiv:2212.04356). arXiv. https://arxiv.org/abs/2212.04356

Rudrabha. (s.f.). *Wav2Lip* [Repositorio de software]. GitHub. https://github.com/Rudrabha/Wav2Lip

Secretaría de Educación Pública. (2026). *Dossier de resultados: Encuesta Nacional sobre Usos y Percepciones de la Inteligencia Artificial Generativa en la educación superior en México (ENIAG 2025)*. Gobierno de México. https://www.gob.mx/cms/uploads/attachment/file/1071411/Dossier_Resultados_ENIAG2025_SEP_.pdf

Wooldridge, M. (2009). An Introduction to MultiAgent Systems (2nd ed.). Wiley.

Zhang, W. (s.f.). *SadTalker* [Repositorio de software]. GitHub. https://github.com/OpenTalker/SadTalker