OpenOmni: An Open-Source Multimodal Systems

Abstract

Multimodal conversational systems are increasingly sought after for their ability to facilitate natural and human-like interactions. However, comprehensive, collaborative development and benchmarking solutions remain scarce. Proprietary models like GPT-4o and Gemini have showcased impressive integration of audio, visual, and textual data, achieving response times between 200-250 milliseconds. Nonetheless, challenges persist in managing the trade-offs between latency, precision, financial cost, and data confidentiality. To address these complexities, we introduce OpenOmni, an open-source, end-to-end pipeline benchmarking platform. OpenOmni incorporates advanced technologies such as Speech-to-Text, Emotion Detection, Retrieval Augmented Generation, and Large Language Models, while also offering the capability to integrate custom models. It supports both local and cloud deployment, thereby guaranteeing data privacy and providing latency and accuracy benchmarking capabilities. This adaptable architecture allows researchers to tailor the pipeline to pinpoint performance bottlenecks and expedite the development of proof-of-concept solutions. OpenOmni holds significant potential to improve applications, including indoor assistance for individuals with visual impairments, thereby advancing human-computer interaction.

1 Introduction

Large Language Models (LLMs) have shown remarkable proficiency in interpreting user intent and adhering to instructions. However, text-based human-computer interaction (HCI) is often inadequate. The recent introduction of models that process audio, video, and text in real-time highlights the progress towards multimodal interaction. The impressive performance, characterized by response times of 200-250 milliseconds, makes these models suitable for large-scale applications. This marks a trend towards multimodal generative models and applications. One of the early publicly available solutions for multimodal large models that integrate text and images is available, but an open-source, end-to-end conversational agent implementation has not yet been made publicly accessible online.

The preferred mode of multimodal HCI should replicate human interaction, incorporating visual and auditory inputs alongside audio outputs. Despite the existence of various modular components, a comprehensive, integrated, open-source implementation that fosters research and development in this domain is lacking. The integration of existing models, such as audio speech recognition (Speech2Text), multimodal large models (MLMs), and text-to-speech synthesis (TTS), into a multimodal conversation framework reveals substantial difficulties in managing latency and ensuring accuracy. Traditionally, accuracy has posed a significant challenge. However, progress in large language models (LLMs) has significantly enhanced contextual relevance. The primary challenge now lies in minimizing end-to-end latency while maintaining high accuracy. Although it has been shown that this is feasible, the open-source community has not yet replicated these results.

Data privacy is another concern. The closed-source nature of certain solutions raises issues related to cost and data confidentiality. Since these models are not open-source, users are required to upload their data to servers via paid APIs, leading to privacy concerns. The privacy policy indicates that various types of personal information are collected when users create accounts to access services, such as account details, user-generated content, communication data, and social media information.

To facilitate the swift and responsible development of this new form of HCI, it is crucial to establish robust evaluation and benchmarking protocols. For instance, if a user initiates a conversation with a sad and urgent tone, the system should respond appropriately and with patience. Evaluating these interactions is both crucial and difficult for widespread adoption. This project aims to bridge these gaps by:

- Creating an open-source framework to facilitate the development of customizable, end-to-end conversational agents.
- Offering a fully local or controllable end-to-end multimodal conversation solution to address privacy concerns.
- Establishing tools for annotating and benchmarking latency and accuracy, allowing for rapid proof-of-concept development and research.

To accomplish this, we propose the OpenOmni framework, an open-source, end-to-end multimodal pipeline that integrates advanced technologies such as Speech-to-Text (Speech2Text), Emotion Detection, Retrieval Augmented Generation (RAG), Large Language Models (LLMs), and Text-to-Speech (TTS). This framework collects video and audio data via cameras and microphones, processes the data through a customizable agent pipeline, and responds using a speaker. OpenOmni can be deployed on a local server, ensuring secure data management and addressing privacy concerns.

For research purposes, OpenOmni includes tools for straightforward annotation and benchmarking, offering real-time monitoring and performance evaluation of latency. Users can annotate individual components and entire conversations, generating comprehensive benchmark reports to identify bottlenecks. The open-source nature of OpenOmni allows for adaptation across various application domains, such as aged care and personal assistants. Each pipeline component can be enabled or disabled based on specific use cases, facilitating flexible and efficient deployment. Moreover, the framework supports the easy addition of new models, enabling comparisons and further experimentation. The OpenOmni framework allows researchers to focus on solving critical bottlenecks without reinventing the wheel, fostering innovation in multimodal conversational agents. It enables rapid proof-of-concept development, such as indoor conversational robots assisting visually impaired individuals.

2 Related Work

Traditional end-to-end multimodal conversation systems typically employ a divide-and-conquer approach, separating the process into sub-tasks: speech-to-text (automatic speech recognition), image-to-text, text generation, and text-to-speech. Speech-to-text transforms spoken language into written text, while image-to-text produces textual descriptions of images. Text generation, often driven by large language models, generates contextually appropriate responses, and text-to-speech converts these responses back into spoken form. These core components constitute the fundamental structure of the conversational pipeline. The inclusion of image-to-text provides essential context, enhancing natural human-computer interaction, and additional functions like emotion detection adjust responses based on the user's emotional state. An optional safeguard module can be integrated to guarantee that responses are suitable, non-harmful, and controlled, maintaining interaction integrity, particularly in delicate situations. Although this modular design enables the optimization of individual components, the cumulative latency and accuracy errors can make the complete system impractical for real-world use.

While certain models are presented as fully end-to-end solutions, capable of handling video, audio, or text inputs and producing audio, image, or text outputs, their technical specifics remain undisclosed. It is postulated that audio and video frames are processed by modules that generate text, audio, and image outputs. Demonstrations suggest that these models possess memory capabilities, though the details and limitations are not fully understood. Whether the system can directly incorporate external private data is also unknown.

Unlike the divide-and-conquer method, a fully end-to-end neural network can integrate more contextual information, such as tone, the presence of multiple speakers, and background noises, leading to more adaptable outputs. Theoretically, this method can decrease latency by removing orchestration bottlenecks. Nonetheless, both methods face substantial challenges because of the extensive data input and output, especially from video. The large size of video files puts a strain on servers and models, raising computational costs and introducing latency from data transfer and model inference. Real-time conversation necessitates streaming processing, posing additional latency challenges. It was highlighted that a stable internet connection is needed to ensure smooth operation, underscoring these challenges.

A technology company has introduced a planned open-source, fully end-to-end multimodal conversational AI, which supports text and audio modalities but excludes images. This model claims to achieve an end-to-end latency of 200 milliseconds. Integrating video modality through an Image2Text module into this model is possible, creating a hybrid solution that combines divide-and-conquer and fully end-to-end approaches. Another viable hybrid solution involves using speech-to-text to convert audio into text, then feeding this text along with video (processed into image sequences) to a vision language model, which generates text responses. These responses can subsequently be processed through text-to-speech. Multimodal end-to-end conversational agents show promise, yet large-scale implementation is challenging due to the need to balance latency, accuracy, and cost. Generating real-time responses within 200-400 milliseconds is difficult. The primary objective is to decrease latency and cost while enhancing accuracy, thereby improving the real-world applicability of conversational agents.

2.1 Evaluation Metrics

To ensure productive and effective collaboration, it is crucial to have consistent and comparable evaluation metrics. For speech-to-text, the Word Error Rate (WER) is used to assess transcription accuracy, where a lower WER signifies better performance. Evaluating text-to-speech involves objective metrics like the Mean Opinion Score (MOS) for naturalness and intelligibility, and the Signal-to-Noise Ratio (SNR) for clarity, along with subjective human ratings. Text generation is the most difficult to evaluate, using metrics such as BLEU, ROUGE, and METEOR, which compare generated text to reference texts but may not completely capture the quality and relevance of responses. Assessing text generation often necessitates large-scale datasets, which are not always accessible. These metrics are widely adopted by the research community. Nevertheless, real-world applications require evaluation in production environments, taking into account various factors beyond these metrics. For instance, a conversational agent designed for aged care should steer clear of sensitive topics that may be specific to each individual. Subjective opinions differ by region, emphasizing the necessity for adaptable and innovative automatic or semi-automatic evaluation methods for conversational agents.

3 System Design

3.1 Requirement Analysis

The system is designed to accept audio and video inputs and produce audio as output. Initially, two modules are required: one for gathering audio and video data from the microphone and camera, and another for emitting audio through a speaker. These Client modules must be compatible with a variety of devices, such as smartphones, laptops, or Raspberry Pi. The data collected will be transmitted to a server.

The server, known as the API, should handle audio and video data along with associated metadata. It should have access to a storage layer that includes a relational database, file management, and a graph database for potential GraphRAG integration. Although the API can be located on the same device as the Client module, it is preferable to keep them separate for enhanced adaptability. This separation introduces the difficulty of transferring large volumes of data between modules. If the API is cloud-based, audio and video data must be uploaded to the cloud, for instance, using AWS S3, Azure Blob Storage, or Google Cloud Storage. However, the upload process can introduce a bottleneck, making data transfer time-intensive. If the server is local, within the same network as the Client, transfer latency will be reduced. Nevertheless, this configuration necessitates running the large language model locally, which addresses data ownership and privacy issues but may increase model inference latency and reduce accuracy due to limited computational resources. Another approach is edge computing, where video data is pre-processed on edge devices and summarized for the API. Although this could be a research direction, data compression might result in information loss and decrease overall performance.

The pipeline components will require adjustments if developers intend to adopt the framework and integrate it with their work. To maintain flexibility, this part should be an independent module capable of running locally or in the cloud. Researchers and developers should be able to easily incorporate new components into this Agent module, further complicating the sharing of large datasets between modules.

Finally, benchmarks are needed to comprehend the latency and accuracy performance of the entire pipeline. For tasks that are challenging to evaluate automatically, such as assessing the appropriateness of the LLM response, we propose and develop an annotation module to allow human annotators to easily evaluate results and generate benchmark reports.

3.2 System Architecture

Based on these requirements, the system architecture was designed as depicted in Figure 1. The system is divided into five modules: Client, API, Storage, User Interface, and Agent, all primarily developed in Python. The Client module includes two submodules: the Listener for collecting video and audio data, and the Responder for playing audio. The Storage module consists of file storage for media, a relational database (PostgreSQL) for metadata, and a graph database (Neo4j) for potential GraphRAG integration. The API module, built with the Django framework, extends Django's admin interface and permission control system to develop the benchmark and annotation interface. Django's maturity and large support community make it ideal for production development. The Agent module, also in Python, includes all agent-related submodules, allowing deployment on suitable compute nodes without altering the architecture. Communication between the Client, API, and Agent modules will be via RESTful endpoints. For sharing large data between modules, local deployments (e.g., Client on Raspberry Pi, API and Agent on local servers) will use FTP for file synchronization. In cloud solutions (e.g., AWS), files will be uploaded to AWS S3, triggering a Lambda function to download files to an AWS Elastic File Storage (EFS) shared by the API and Agent modules. Docker and Docker Compose are used to manage all modules, allowing easy setup with a single docker compose up command.

4 Demonstration

4.1 Datasets

Most multimodal question-answering datasets concentrate on multiple-choice questions rather than open-ended conversations. Some datasets involve multimodal conversations with images as additional input, but the output is often limited to multiple-choice or text. A significant challenge in developing multimodal conversational agents is the scarcity of suitable datasets.

Although there is an abundance of data from human-human interactions or data extracted from movies and YouTube videos, efficient methods to organize this data into structured datasets are lacking. For specific domain applications, collecting data from human interactions and extracting datasets to train systems would be advantageous, enabling the agents to mimic human behavior. The OpenOmni Framework offers both capabilities: extracting conversational datasets from videos and testing them through the pipeline to assess agents' responses, or gathering data from real-world scenarios to create datasets for further research.

4.2 Can "AI" be your president?

One intensive conversational scenario is a debate. Segments were extracted from a US Presidential Debate, focusing on a candidate addressing the public and handling questions. After downloading the videos, a prepared script in our codebase can be used to split them into segments. This script allows for the specification of the start and end times of each conversation, enabling the creation of a conversational dataset from the videos. These segments were fed into our pipeline to evaluate its performance under different configurations: one using a commercial speech-to-text model, a vision model, and text-to-speech (Configuration A); a locally deployed quantization LLM with a speech-to-text model, text-to-speech, and our emotion detection model for video input (Configuration B); a version using a different LLM for inference (Configuration C); and a version using only a speech-to-text model, a language model, and text-to-speech, ignoring the video modality (Configuration D). The Agent modules were run on a specific GPU with 12GB memory.

The latency benchmark statistics are automatically generated. For example, Configuration A has an average latency of 45 seconds, with the vision model accounting for 31 seconds. The fastest configuration is Configuration D, averaging around 15 seconds, with most of the time consumed by the text-to-speech part, because the generated content is quite long and comprehensive. The slowest configuration is Configuration C, taking around 189 seconds, with the LLM model inference step taking the longest time. Configuration B takes an average of 60 seconds, with the LLM model inference averaging 28 seconds and our emotion detection model averaging around 10 seconds.

Table 1: Accuracy: Overall Conversation Quality

TRACK ID	USER ID	OVERALL COMMENT
f1	1	As the question is quite subjective, the answer is good and in context
f2	2	The answer is quite general, while the candidate is doing much better work with supported eviden
f3	1	Failed to generate proper in-context response; the response is talking about how to respond, not ac
f4	1	Generate some general comments without strong support evidence
f5	1	General response, however, no good evidence to support.

After annotation with our interface, accuracy statistics are automatically generated. The accuracy metrics here include evaluation metrics like WER, CER for the speech-to-text task, and overall scores given by the annotators. As shown in Table 1, the average score for each conversation is 2.4. Text-to-speech can be improved with more natural emotion or personality. The generated content is often too general and sometimes inappropriate. The candidate's responses are more in-context and evidence-supported. The pipeline excelled only in answering a subjective question about the candidate's age, where Configuration A performed well. Configuration D had the best overall accuracy, but its responses were often in-context yet pompous. Thus, the candidate still outperforms AI. In conclusion, "AI cannot be the President of the US just yet, considering both latency and accuracy."

4.3 Assist the Visually Impaired

While latency and the need for external information currently prevent AI from undertaking mission-critical tasks, conversational agents can be production-ready and useful for non-latency-critical areas that do not require extensive external knowledge. Assisting indoor activities for the visually impaired is one such application, where high-speed internet can be utilized, or data transfer can be limited to local exchanges. These types of applications can benefit from maintaining high input/output rates, helping to mitigate latency issues. Questions were prepared for the visually impaired, including locating objects, navigating indoors, and inquiries about the surroundings. Six questions were sampled and fed to the Configuration A pipeline. One scenario demonstration is included in our provided video. In this scenario, video and audio data stream from the client side and are saved to storage along with exportable metadata accessible via the admin portal. This setup allows for the exportation of annotated datasets, including raw video and audio data, for developing new models. The latency statistics show responses within approximately 30 seconds.

Annotated results show a 4.7/5 accuracy, but the agent lacks specific skills for assisting the visually impaired. For example, ideally, it should provide step-by-step instructions on grabbing a coffee cup rather than just a general description. This indicates that while conversational agents are nearly ready for assisting the visually impaired with indoor activities, improvements in latency and response quality are still needed.

5 Conclusion

Multimodal conversational agents offer a more natural form of human-computer interaction, as demonstrated by models like GPT-4o. However, real-world constraints require a balance between cost, latency, and accuracy, which may explain why the full capabilities of such models are not yet accessible.

Several technical options exist to achieve this balance, including traditional divide-and-conquer methods, fully end-to-end models, and hybrid approaches. The fully end-to-end approach inherently allows for lower latency, while the divide-and-conquer method faces latency issues when coordinating multiple components. Both approaches must address the challenge of handling large data I/O. If models are deployed locally, local network I/O issues can be more manageable. However, some models are closed-source, making local deployment impractical. While deploying other vision models locally is feasible, achieving high accuracy may be limited by local computational resources. Hybrid solutions provide alternative approaches: pre-processing or compressing large data locally and then utilizing cloud-based models, or converting video to text and integrating it into the end-to-end voice model.

We developed the OpenOmni framework to enable researchers to integrate their work into an end-to-end pipeline. The framework supports various solutions, allows for pipeline customization, generates latency performance reports, and provides an annotation interface for accuracy review. These features facilitate the creation of benchmark reports to identify and address key issues.

Testing with the US Presidential debate scenario highlighted latency as a critical issue, particularly with large video data. Integrating external knowledge remains a challenge, emphasizing the need for efficient Retrieval-Augmented Generation (RAG). For applications like indoor assistance for the visually impaired, latency improvements and model adaptation are both essential.

The OpenOmni framework can significantly benefit the research community by facilitating the collection and management of new datasets, integrating various conversational agents approaches, and generating automatic latency benchmarks. Its annotation interface aids in accuracy performance review, making OpenOmni production-ready for suitable application scenarios and fostering further development in multimodal conversational agents.