

Sink or SWIM: Tackling Real-Time ASR at Scale

Federico Bruzzone , Walter Cazzola , Matteo Brancaleoni and Dario Pellegrino

Abstract—Real-time automatic speech recognition systems are increasingly integrated into interactive applications, from voice assistants to live transcription services. However, scaling these systems to support multiple concurrent clients while maintaining low latency and high accuracy remains a major challenge. In this work, we present **SWIM**, a novel real-time ASR system built on top of OpenAI’s **Whisper** model that enables true model-level parallelization for scalable, multilingual transcription. **SWIM** supports multiple concurrent audio streams without modifying the underlying model. It introduces a buffer merging strategy that maintains transcription fidelity while ensuring efficient resource usage. We evaluate **SWIM** in multi-client settings—scaling up to 20 concurrent users—and show that it delivers accurate real-time transcriptions in English, Italian, and Spanish, while maintaining low latency and high throughput. While **Whisper-Streaming** achieves a word error rate of approximately 8.2% with an average delay of approximately 3.4s in a single-client, English-only setting, **SWIM** extends this capability to multilingual, multi-client environments. It maintains comparable accuracy with significantly lower delay—around 2.4s with 5 clients—and continues to scale effectively up to 20 concurrent clients without degrading transcription quality and increasing overall throughput. Our approach advances scalable ASR by improving robustness and efficiency in dynamic, multi-user environments.

Index Terms—Automatic Speech Recognition, **Whisper**, Multi-client ASR Systems, Real-time ASR Systems.

I. INTRODUCTION

AUTOMATIC speech recognition (ASR) systems¹ are now ubiquitous in everyday applications [1, 2, 3, 4], ranging from voice assistants for human-computer interaction [5, 6] to transcription services [7, 8]. In recent years, ASR has advanced significantly through deep learning [9, 10, 11, 12]. The field has shifted from traditional *cascaded* architectures [13, 14], which separate the recognition pipeline into distinct stages, to *end-to-end* models [15], where a single neural network jointly learns the entire task [16, 17]. **Whisper**, developed by OpenAI, is a notable transformer-based end-to-end ASR model [18]. It represents a well-balanced end-to-end solution in the ASR landscape, offering a trade-off between computational cost, speed, and transcription accuracy, as demonstrated by Radford et al. [18].

ASR Systems and Their Challenges. Contemporary ASR systems, particularly those based on conversational AI, are

F. Bruzzone is with the Department of Computer Science, Università degli Studi di Milano, Milan, Italy. E-mail: federico.bruzzone@unimi.it.

W. Cazzola (corresponding author) is with the Department of Computer Science, Università degli Studi di Milano, Milan, Italy. E-mail: cazzola@di.unimi.it.

M. Brancaleoni is with the Computer Science Division, VoiSmart, Milan, Italy. E-mail: matteo.brancaleoni@voismart.it.

D. Pellegrino is with the Computer Science Division, VoiSmart, Milan, Italy. E-mail: dario.pellegrino@voismart.it.

¹From here on, we use the term *system* to refer specifically to a software system, typically in a client-server setup, where the client is the user and the server hosts the ASR model.

generally characterized along two dimensions: *single-client* systems [9, 19], which serve a single user at a time, and *multi-client* systems, which concurrently serve multiple users [20, 21]. These architectures are further classified based on deployment, into *local* systems, which operate on the user’s device, and *remote* systems, which are hosted on servers and accessed via network connections. Recent trends in ASR research have increasingly focused on real-time applications, where both low latency and high throughput are critical [1, 22, 23, 24]. In these settings, the system must generate responses within a few seconds to meet users’ expectations for immediate feedback. Although local ASR systems offer enhanced privacy [25, 26], their performance is inherently constrained by the limited computational power and memory of user devices [27], as well as their susceptibility to adverse environmental conditions. For example, in audio-only communication channels such as *voice over internet protocol* (VoIP) and telephone calls, audio signals are typically routed to centralized remote ASR systems that benefit from greater computational resources and cloud-based services [28, 29].

Single-client vs. Multi-client Systems. Furthermore, single-client systems are not without limitations [30]. E.g., **Whispy** [31] and **Whisper-Streaming** [32] lack model-level parallelization, limiting their scalability in multi-audio streams scenarios. In contrast, remote ASR systems can leverage greater computational resources and memory, enabling efficient scaling to support multiple clients concurrently [33]. Such systems benefit from optimized resource allocation, reduced latency via parallel processing, and consistent performance across diverse user populations [34, 35].

Nevertheless, scaling real-time ASR to handle many concurrent clients remains a major challenge. This is largely due to the computational demands of real-time models, which must deliver low-latency transcriptions without losing accuracy—a critical requirement in applications like voice assistants [36], live captioning [37], and safety-critical domains such as healthcare [38, 39, 40] or legal proceedings [41, 42].

Contributions. In this paper, we introduce **SWIM** (Serve **Whisper** In Multi-client), a novel real-time ASR system built on top of the **Whisper** model, designed to serve multiple clients simultaneously while maintaining low latency and high throughput. **SWIM** prioritizes scalability, enabling efficient handling of multilingual audio streams under varying workloads. Unlike **Whisper-Streaming** [32] and **Whispy** [31], **SWIM** achieves true model-level parallelization, ensuring stable performance even under high-load conditions. Through empirical evaluation, we demonstrate that a single **Whisper** instance running **SWIM** can transcribe multiple multi-lingual audio streams in parallel, maintaining optimal accuracy in real-time scenarios. Our evaluation spans various multi-client

settings (e.g., 5, 10, and 20 concurrent users) and includes: (i) assessment of real-time playback performance, in terms of delay with respect to wall-clock time, (ii) measurement of transcription quality using *word error rate* and accuracy metrics, and (iii) evaluation of the system’s handling of multilingual inputs, including English, Italian, and Spanish. This work tackles scalability challenges in real-time ASR, advancing the state of the art in multi-client speech recognition for dynamic environments.

Our main contributions are:

- 1) a scalable real-time ASR system enabling model-level parallelization for concurrent multilingual transcription;
- 2) a buffer merging strategy that maintains transcription accuracy across clients without added latency; and
- 3) a demonstration of **Whisper**’s unmodified ability to handle multiple parallel audio streams from diverse sources effectively.

The remainder of the paper is organized as follows. Sect. II provides background. Sect. III reviews real-time ASR systems, focusing on **Whisper** and its limitations; Sect. IV details the design of **SWIM**, and Sect. V presents our evaluation, and Sect. VII concludes the paper.

II. BACKGROUND

We provide background on speech-to-text ASR models, the **Whisper** Model, and the **Whisper**-Streaming system.

Speech-to-Text ASR Models. ASR converts spoken language into written text using machine learning. Traditionally, ASR systems have relied on hidden Markov models [43, 14] and Gaussian mixture models [13], which dominated the field for many years. ASR models vary from high-performance, *heavy* systems designed for offline processing [16, 44] to *lightweight* versions optimized for real-time use on limited hardware [45, 46]. These differences shape key architectural and optimization decisions.

Traditional ASR models often used a *cascaded* architecture [13, 14], where separately trained components process speech sequentially. In contrast, *end-to-end* models [15] typically integrate these stages into a single neural network trained jointly [16, 17], simplifying the pipeline and often improving performance, especially with large datasets [16, 17]. A prominent example is the *transformer* architecture [16, 47], which replaces recurrent and convolutional neural networks with self-attention mechanisms. Transformers enable high parallelization and achieve state-of-the-art results across many ASR tasks [16, 48, 49]. ASR models are commonly trained on large-scale datasets such as LibriSpeech [50]—available in variants like 1) LibriSpeech Long [51] and 2) Multilingual LibriSpeech [52]—as well as Fleurs [53]. These datasets provide thousands of hours of transcribed speech, supporting robust model training.

Performance is typically measured by *word error rate* (WER) [54, 55, 56], *character error rate* (CER) [57, 58, 59], and *sentence error rate* (SER) [60, 61], which quantify transcription accuracy using normalized Levenshtein distance [62, 63] between predicted and reference texts. Synchronized transcription of audio streams is crucial—especially for subtitles,

captions, or live transcription [64, 65]. To address this, ASR systems segment audio into temporal segments transcribed individually. This *timestamp-aligning*² process [66, 67] aligns transcriptions with the audio, enabling extraction of precise temporal segments.

Whisper Model. **Whisper** [18] is an open-source ASR model developed by OpenAI. It is trained on a large corpus of multilingual and multitask datasets, enabling it to perform multilingual speech recognition. **Whisper** is a prime example of an end-to-end ASR model based on the Transformer architecture. Unlike traditional ASR models, **Whisper** processes raw audio and uses a sequence-to-sequence transformer model to generate text. Being an end-to-end model, the **Whisper** consists of an *encoder* and a *decoder* architecture. The former processes the audio input into a dense representation, while the latter translates the representation into a sequence of text tokens using cross-attention mechanisms. Beyond speech-to-text transcription, **Whisper** demonstrates high proficiency in additional tasks, including: 1) speech translation, 2) language identification and 3) voice activity detection. At inference time, a list of start and end timestamps can be provided to specify audio segments, bypassing voice activity detection by assuming these intervals contain speech. **Whisper** is available in several sizes and variants—e.g., *tiny*, *small*, *medium*, *large-v3*. Recently, a *turbo* version of the pre-trained *large-v3*, pruned and fine-tuned, was introduced, offering a 5–8× speedup with minimal accuracy loss.

III. STATE-OF-THE-ART

Real-time Whisper. **Faster-whisper**³ is a reimplementation of **Whisper** built on top of **CTranslate2**.⁴ It has been widely adopted by several real-time ASR systems, including **Whispy** [31] and **Whisper**-Streaming [32].

Whisper-Streaming enables real-time speech transcription and translation by using an incremental buffer and local agreement mechanism to minimize latency and maximize throughput [32]. **Whispy**, on the other hand, is designed to support real-time use of **Whisper** by employing Levenshtein distance to align overlapping transcriptions and select the most accurate segments [31]. Despite adopting different approaches, both projects rely on **faster-whisper** as the **Whisper** implementation, minimizing latency and offering a valuable contribution to the field. However, **Whisper**—including its variants—is GPU bound and introduce big bottlenecks when it comes to real-time scenarios and parallelization. Other projects have proposed solutions for these issues, such as **Whisper-T** [68]. **Whisper-T** aims to reduce **Whisper**’s GPU load and power consumption through lightweight decoding strategies, beam pruning, and CPU/GPU pipelining. Nevertheless, the project remains limited to a local environment, processing only one audio stream at a time. Achieving significant scalability and parallelization for **Whisper** remains an open challenge and is essential for building remote, multi-client ASR systems.

²Also known as *temporal*-, *forced*-, or *word-level alignment*.

³<https://github.com/guillaumekln/faster-whisper>

⁴<https://github.com/OpenNMT/CTranslate2>

Limitations of the State-of-the-art. The goal is to develop a system capable of handling multiple requests while running on relatively modest hardware (e.g., a few GPUs or even a single GPU). Achieving this presents several challenges, including: 1) most existing `Whisper` implementations (except `faster-whisper`) exhibit significant performance degradation in real-time scenarios, and 2) the absence of model-level parallelization support. Several `Whisper` implementations, such as: `insanely-fast-whisper`⁵ and `WhisperX` [69] (which builds upon `faster-whisper`), are not suitable for real-time applications due to latency issues. The former performs poorly on short audio clips (e.g., 15–30 seconds) and is not designed for processing raw audio buffers. While it includes a *diarization*⁶ feature, the latter inherits the limitations of `faster-whisper`, making it slower than its base implementation. Moreover, `Whisper` and its variants are not inherently designed to process multiple audio streams concurrently, primarily due to two reasons: 1) although available VRAM may support multiple model instances, GPU compute resources are quickly exhausted during inference, and 2) when multiple `Whisper` instances are launched, only one can utilize GPU computation at a time, forcing the others to wait—resulting in serialized audio processing. A high-cost solution, feasible for large companies, involves horizontal scaling with a dedicated GPU for each `Whisper` instance. A more affordable alternative is asynchronous processing on a single GPU, though this is poorly supported in `faster-whisper` and similar implementations.

IV. A DEEP DIVE INTO SWIM

SWIM is a multi-client, real-time ASR system built on top of the `Whisper` model. It is designed to: 1) concurrently process multiple multilingual real-time audio streams, 2) maintain low latency and high throughput, and 3) deliver accurate transcriptions for all clients.

A. Overview

The actors in **SWIM**’s architecture are illustrated in Fig. 1. The system consists of multiple clients **1**, each streaming audio to a dedicated service **2**. Each service is responsible for two main tasks: 1) building an *audio buffer* from incoming audio chunks and submitting it to the *shared ASR model* **3** for transcription, and 2) maintaining a *hypothesis buffer* **4**, which stores intermediate transcriptions and applies a *local agreement* strategy to ensure reliability. The *shared ASR model* assembles clips from all *audio buffers* **5**, passes them to a `faster-whisper` instance **6** for inference, and relays the transcriptions to the *result dispatcher* **7**.

B. Clients

SWIM is designed to be a multi-client ASR system. It can be deployed on a remote server or adapted for local use without the client-server setup—e.g., handling multiple audio

sources on a single machine. Clients, shown in Fig. 1-**1**, send audio streams to the system. These can be any audio-capturing devices, such as smartphones, computers, or single-board development kits like Raspberry Pi or Arduino. Client implementations emphasize polyglotism and platform agnosticism, supporting any programming language compatible with the gRPC protocol.⁷ Clients receive transcriptions of their audio streams via a *bidirectional* communication channel, allowing the server to return transcriptions and other related information. Clients must send audio streams sampled at 16 kHz, the rate required by `Whisper` and widely adopted in modern ASR systems. Knowing this sampling rate is essential for the system to effectively process audio buffers—such as chunking based on timestamps or synchronizing local transcriptions.

C. Services

As illustrated in Fig. 1-**2**, services handle the audio streams sent by clients. Each service is dedicated to a single gRPC client and constructs an *audio buffer* from the incoming chunks. These *audio buffers*—shown in and in Fig. 1-**2**—are continuously forwarded to the `Whisper` model for processing (indicated by arrows from **2** to **3**) to maintain real-time performance. To reduce processing overhead, the audio buffer is kept as short as possible (typically 10–15 seconds). If the buffer exceeds its maximum duration (e.g., 15 seconds), the oldest content is trimmed. This trimming applies only to the already transcribed portion of the buffer, which has been marked as *confirmed* by the *local agreement* mechanism within the *hypothesis buffer*⁸ (see Sect. IV-E). Thanks to the *word-level* timestamp alignment provided by `faster-whisper`, the buffer can be segmented at precise word boundaries, preventing word splits. Each segment—depicted in Fig. 1-**7**—is a continuous sequence of words with start/end timestamps, along with metadata such as full text and detected language. This design facilitates operations like overlap resolution, insertion, and deletion in the *hypothesis buffer* (**3**–Fig. 1). Specifically, the buffer tail (**4**–Fig. 1) is truncated at the timestamp of the first confirmed word that occurs before the midpoint of the buffer duration, i.e., $\ell < \frac{B}{2}$, where B is the total buffer length (e.g., 15 seconds). As shown in Fig. 1-**2** (Service 1), the confirmed portion of the *hypothesis buffer* is marked in *yellow* and *green* while the unconfirmed portion is shown in *red* . When a new text segment is confirmed (as in Service 1), it is marked *green* and sent back to the client, along with its start and end timestamps. Details on the local agreement policy can be found in Sect. IV-E.

D. Shared ASR Model

Shared Audio Buffer. As we have already mentioned, **SWIM** is designed to be a multi-client ASR system. However, each service does not have a dedicated instance of the `Whisper`

⁵<https://github.com/Vaibhav510/insanely-fast-whisper>

⁶*Diarization* refers to segmenting an audio stream by speaker identity, allowing the system to identify who spoke when.

⁷<https://grpc.io>

⁸The *hypothesis buffer* (Fig. 1-**3**) stores transcriptions from the shared model’s result dispatcher (Fig. 1-**7**). It functions by comparing the latest confirmed transcription with the current one.

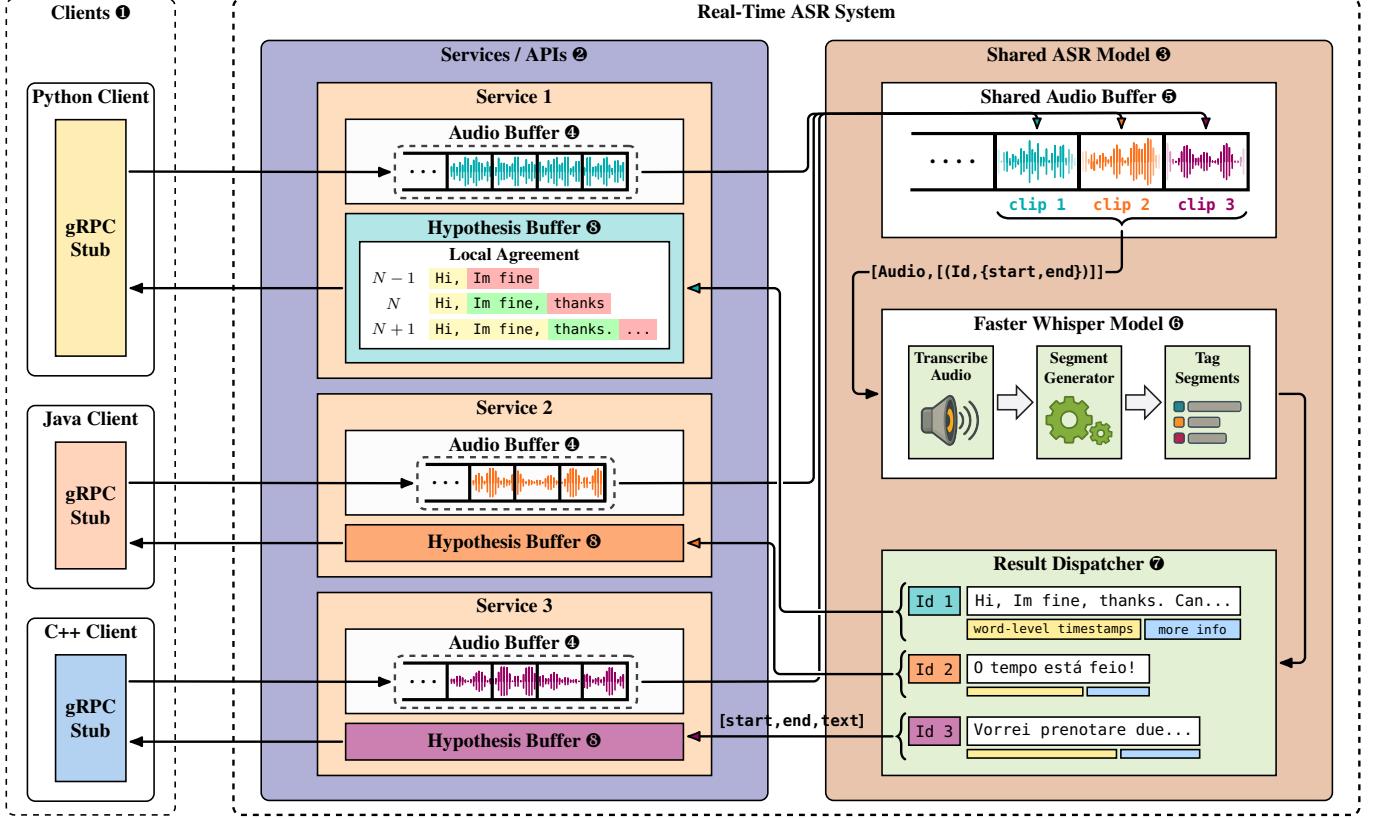


Figure 1: Architecture of **SWIM**. Multiple clients ❶ stream audio to dedicated services ❷, which forward data to a *shared ASR model* ❸ for parallel processing.

model. Instead, **SWIM** is based on a *shared ASR model*⁹ Fig. 1-❸, which, as the name suggests, is shared across all services ❷. To function properly and to enable parallel processing of multiple audio streams, the *shared model* is designed to operate using a single *shared audio buffer* Fig. 1-❹, which is likewise shared among all services. The *shared ASR model* is responsible of orchestrating the parallel progressing of the audio streams, by continuously: 1) gathering the services *audio buffers* into a *shared audio buffer*, 2) processing them in parallel using a *faster-whisper* instance Fig. 1-❺, and 3) tagging and sending the text results back to the services (see *Result Dispatcher* Fig. 1-❻).

The *shared audio buffer* is a monolithic raw buffer built on top of the individual *audio buffers* managed by each dedicated service. Its internal structure enables the system to track which portions of the buffer belong to each client. Alongside the *audio buffer*, the system maintains a list of pairs consisting of: 1) the unique identifier of the associated service, and 2) the clip timestamps marking the start and end of the corresponding service’s audio segment. The *shared audio buffer* is populated by the *shared ASR model* when all the services are ready to process their *audio buffer*—that is, a progressing request (e.g., a novel audio chunk) has been received from the client. Several policy mechanisms to avoid deadlocks could be easily

⁹When we refer to the *Shared ASR model* or *Shared model*, we mean an instance of a custom object that wraps the *faster-whisper* model and provides a set of methods to interact with it by allowing it to be used in a multi-client environment.

implemented to handle the situation when one or more services are not ready to process their *audio buffer*. For instance, the system could wait for a certain amount of time (e.g., range from 1 to 2 seconds) for a service to be ready. When unready, the system could skip it and proceed regardless with two behaviors: 1) skipping the service, but pausing or kicking it out from the system, and 2) skipping the service only for the current transcription, but keeping it active for the next ones.

Each service operates autonomously and processes its own audio stream independently, without interference from other services. As a result, synchronization issues are implicitly managed by the system architecture. A core assumption of **SWIM** is that the audio stream is persistent and transmitted continuously in fixed-duration chunks, with each chunk having a duration $D \in (0, 1]$. Consequently, as Fig. 2 illustrates, there always exist two consecutive audio chunks, denoted by (D_1, T_1) and (D_2, T_2) , where D_j represents the audio chunk received by client C_j at time T_j . Let $(T, i) \leftarrow (\max\{T_1, T_2\}, \arg \max_{i \in \{0,1\}} \{T_i\})$, then the following inequality holds:

$$T - T_{|i-1|} - \varepsilon \leq 1s$$

where:

- 1) $T_{|i-1|}$ is the timestamp of the last audio chunk received by client $C_{|i-1|}$, and
- 2) $\varepsilon = o(1)$ represent the network latency of T .

This architectural assumption, as Fig. 2 depicts, ensures that the maximum delay incurred while waiting for all services to

be ready is bounded by

$$\max(0, 1 - T_p) < 1s,$$

where T_p is the processing time of the transcriptions. Intuitively, this represents the worst-case scenario where one service has to wait for the other to finish processing and sending its audio chunk.

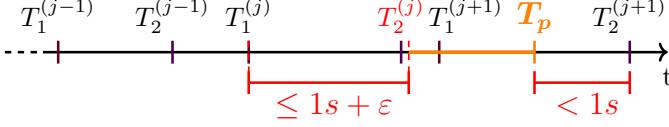


Figure 2: Audio chunk processing time diagram.

The outer maximum with zero ensures that we do not consider negative delays: if the *shared ASR* takes longer than 1 second to process all active services, the next chunk will already have arrived for every connected services, so there’s no extra waiting time.

Faster Whisper Model. In the adopted architecture, the entire *shared audio buffer* is passed to the model as a single, continuous waveform. Individual client contributions are identified through a list of uniquely associated identifiers and temporal intervals defined by a pair (`start`, `end`)—denoted as `Timestamp` in Fig. 1 (in the path from ❷ to ❸).

Inference is then performed on the entire *shared buffer*, based on the artificially defined clips—that is, the *shared audio buffer* divides the audio stream into clips, each associated with a unique client identifier. For each registered clip—denoted by ❶, ❷, ❸ colors in Fig. 1–❸—the system extracts a corresponding transcription segment. As briefly mentioned in Sect. IV-C, the length of the service *audio buffer* should range between 10 and 15 seconds. This design choice is motivated by multiple, complementary factors. First, it alleviates the computational load by limiting the input size processed at each inference step. Second, and more importantly, clips that are too long—exceeding approximately 30 seconds are cut off the excessive part of the clip.

Result Dispatcher. After inference, the text segments are returned in the exact order in which they were processed. With this design choice, the system can easily associate each segment with the corresponding client identifier. Before returning the results, the system adjusts the segment timestamps—including word-level timings—so they align with the relative time of the original service audio buffer. This adjustment is necessary because the timestamps are relative to the *shared audio buffer*, not to the service-specific *audio buffer*. Finally, the segment results are forwarded to the corresponding service, which processes the segment and delivers the transcription back to the client according to the previously described mechanism (on the arrow from ❷ to ❸).

E. Hypothesis Buffer with Local Agreement

SWIM uses an adapted *hypothesis buffer*, inspired by the mechanism introduced in *Whisper-Streaming* [32], which implements a *local agreement* strategy to ensure the reliability of the transcriptions produced by the *Whisper* model. Our implementation employs a *similarity metric* [31, 62, 70] rather

than strict token equality to govern the behavior of the *hypothesis buffer*. **SWIM** uses the *QRatio*,¹⁰ also known as normalized Indel distance, introduced by Bachmann, defined as:

$$QRatio(s1, s2) = \left(1 - \frac{d(s1, s2)}{|s1| + |s2|}\right) \times 100,$$

where $d(s1, s2)$ is the Indel distance [71, 72] between the two strings $s1$ and $s2$ —a variant of the Levenshtein distance [73] where substitutions have a cost of two—, and $|s1|$ and $|s2|$ are the lengths of the two strings.

The motivation behind these changes is twofold: 1) to account for punctuation, apostrophes, or minor discrepancies—whether introduced by *Whisper* itself or by individual services—and 2) to populate the local agreement list (a buffer of words along with timestamps (`start`, `end`, `text`)) and retain content that may have been slightly altered by the model.

When a new segment is received, the system replaces the previous segment with the new one, inserting the first n consecutive tokens whose similarity ratio exceeds or equals 98% compared to the corresponding n tokens from the previous segment. The purpose of this insertion mechanism is to find overlaps between the two segments, which helps hypothesis buffer with being more accurate and reliable. The local agreement mechanism enables token confirmation by comparing the new segment against the previous one, using a token-by-token equality check. By setting a confirmation threshold at a 95% similarity ratio, the mechanism accepts variations such as added or removed punctuation without wasting additional iterations, while maintaining high transcription accuracy. An illustrative example is shown in the local agreement box under the first ❸ of Fig. 1.

If no confirmation occurs via the local agreement policy, a fallback strategy is triggered before proceeding to the next iteration without any confirmation. The system takes the first half of the previous transcription—based on word count—and generates all possible ordered prefixes. These prefixes are then compared to those of the new transcription, but only among tokens whose end timestamps fall within the duration of the selected half of the previous segment. This constraint ensures that matching is confined to the relevant portion of the new transcription, preventing premature confirmation of content beyond the midpoint of the previous segment. The prefix with the highest similarity score is selected, and all its constituent tokens are confirmed individually, as confirmation is performed on a token-by-token basis, adopting the same similarity metric as in the local agreement. This fallback mechanism helps recover from shifts in phrasing or tokenization that can put the service in a never confirming state until the service audio buffer gets trimmed.

V. EVALUATION

The evaluation of **SWIM** utilizes four datasets: the *Google Fleurs dataset* [53], the *Multilingual LibriSpeech dataset* [52], the *LibriSpeech-Long dataset* [51], and the *Italian Parkinson’s Voice dataset* [74, 75, 76]. We use `large-v3-turbo`, a

¹⁰Later in the paper we refer to this *QRatio* metric also as *similarity metric*, *similarity ratio*, or *similarity score*.

pruned and fine-tuned version of the pre-trained `large-v3` model, which offers significant improvements in efficiency. A replication package for the experiments is available at:

<https://doi.org/10.5281/zenodo.15856828>

A. Datasets

Description. The *Google Fleurs* dataset offers a diverse multilingual benchmark with recordings from 102 languages, making it ideal for evaluating **SWIM**’s robustness across a wide range of speech types. The *Multilingual LibriSpeech* dataset consists of read audiobooks in 8 European languages, supporting evaluation in moderately resourced multilingual settings. The *LibriSpeech-Long* dataset extends LibriSpeech by including longer audio segments, testing the model’s ability for long-range temporal modeling. Finally, the *Italian Parkinson’s Voice* dataset contains Italian speech samples from Parkinson’s disease patients and control subjects, enabling assessment of **SWIM** in pathological speech recognition scenarios, which is critical for evaluating performance on impaired speech.

Preparation. We preprocess all datasets to align audio files precisely with their transcriptions. Using the `Whisper` `large-v3` model [18], we obtain word-level ASR alignments that map audio to text accurately. This precise word-level timestamping is essential for evaluating **SWIM** in real-time ASR scenarios requiring tight audio-text synchronization.

We chose the `Whisper` `large-v3` model for its balance of latency, accuracy, and speed, making it well suited for alignment. Its state-of-the-art performance also provides a strong baseline for evaluating **SWIM**, given our method’s constraints and the `large-v3` `turbo` variant’s limitations.

The prepared datasets support evaluation of **SWIM** across multiple scenarios: 1) the *Multilingual LibriSpeech* and *Google Fleurs* datasets assess multilingual concurrent performance, with average audio lengths of approximately 15 and 10 seconds, respectively; 2) the *LibriSpeech-Long* dataset evaluates long-form audio processing, with an average duration around 5 minutes; and 3) the *Italian Parkinson’s Voice* dataset offers an intermediate scenario, with audio segments around 1 minute long, featuring recordings used for disease diagnosis and including specific Italian regional accents.

B. Experimental Setup

We conduct our experiments on a server equipped with: 1) an NVIDIA RTX 6000 Ada Generation GPU with 48GB of VRAM, 2) Intel[®] Xeon[®] Gold 5412U CPU with 24 cores and 48 threads, 3) 128GB of RAM, and 4) the operating system is Ubuntu 22.04 LTS.

C. Experiments and Results

Delay Distribution Plots. Figures 3a and 3b show the delay distribution by language under different client loads, using chunk durations of 1.0s and 0.5s, respectively. Each box plot group is labeled by dataset on the x-axis, with delay (in seconds) on the y-axis. While delays generally increase linearly with more concurrent clients, the plots are

useful for identifying the threshold where performance starts to degrade. This insight helps determine when to switch to offline transcription or scale **SWIM** by allocating additional resources.

The delays are computed as the mean difference between the timestamp of each confirmed transcription segment and its expected delay from the start of the audio stream. In the plots, colors represent different numbers of concurrent clients. As expected, both plots show an approximately linear increase in delay as the number of clients grows, due to increased contention for model inference. Interestingly, with fewer clients, shorter chunks (0.5s) yield lower delays by enabling more frequent updates and faster reaction times. However, as concurrency increases, the 1.0s chunk configuration begins to outperform the 0.5s one, likely due to reduced overhead from less frequent processing.

These results highlight the importance of tuning the chunk duration parameter for optimal performance. Shorter chunks (0.5s) are more responsive in low-load scenarios (up to 15 clients), reducing delay by approximately 11% compared to 1.0s chunks. In contrast, for higher loads (more than 15 clients), 1.0s chunks yield about 8% lower delay, due to reduced processing overhead. Overall, this behavior allows **SWIM** to dynamically adapt to workload conditions, balancing throughput and latency as the number of clients changes.

WER Distribution Plots. Figures 3c and 3d present the WER distribution per language for different numbers of concurrent clients, using chunk durations of 1.0s and 0.5s, respectively. Each group of box plots is labeled with the corresponding dataset on the x-axis, while the y-axis shows the word error rate. As expected, the WER remains relatively stable across varying concurrency levels, with only minor fluctuations. This consistency suggests that transcription quality remains largely unaffected by the number of parallel audio streams. The WER is computed as:

$$\text{WER} = \frac{S + D + I}{N}$$

where S is the number of substitutions, D the number of deletions, I the number of insertions, and N the total number of words in the reference transcription. As expected, WER remains relatively stable across different levels of concurrency, with only minor fluctuations. This behavior suggests that system accuracy is more influenced by architectural choices—particularly the use of a shared buffer—than by the number of concurrent requests. Notably, **SWIM** consistently maintains low WER values in multilingual settings, highlighting one of its key strengths. While chunk duration has some effect on WER, the differences are marginal.

The most notable deviation occurs in the *Italian Parkinson’s Voice* dataset, where the WER is higher and more variable for 5 concurrent clients. This can be attributed to the presence of samples where speakers articulate isolated words with long pauses rather than continuous speech. In such cases, when processing latency is low, the limited context can lead to punctuation mistakes—such as incorrect or missing sentence boundaries—which in turn affect casing. These errors disproportionately inflate the WER metric.

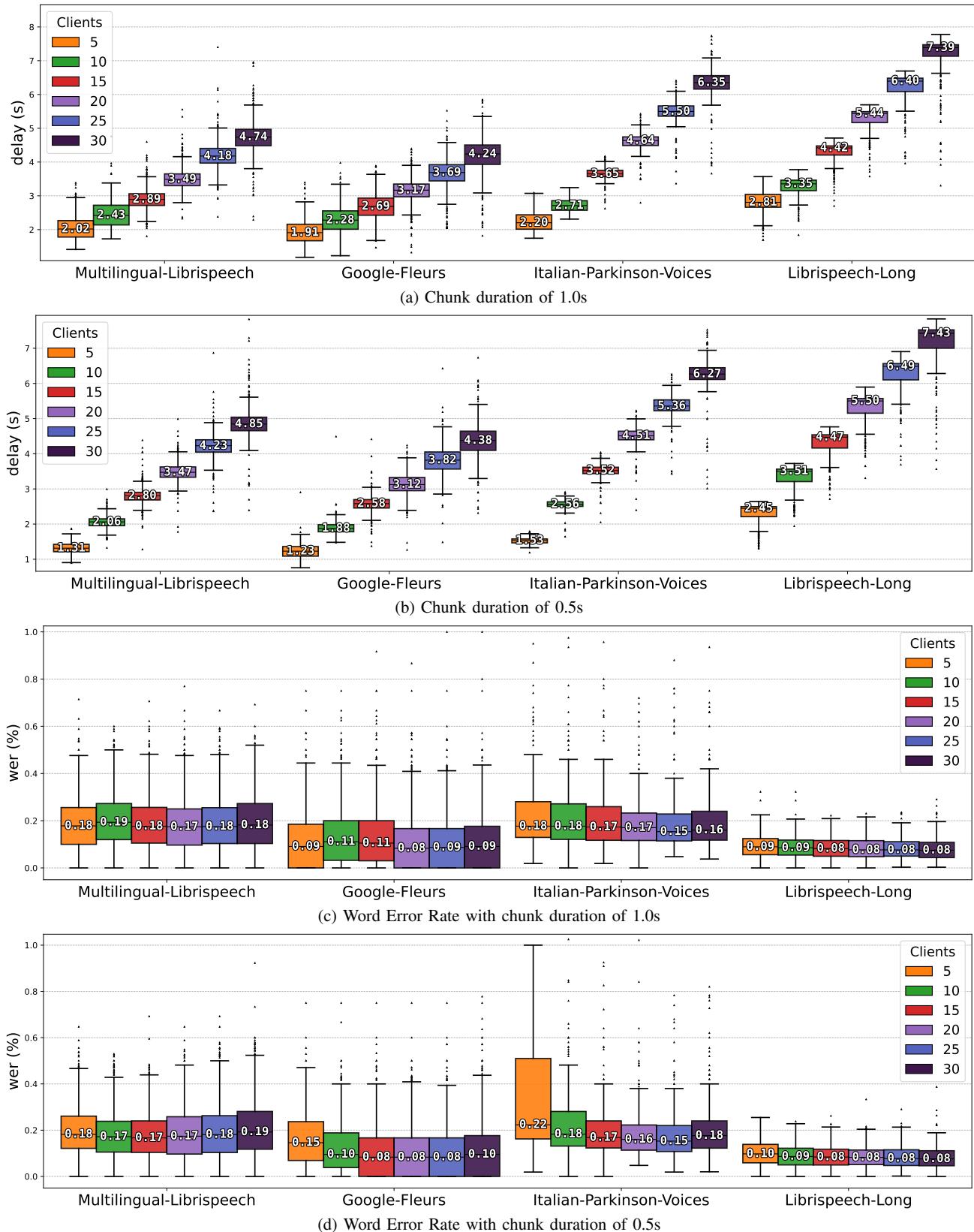


Figure 3: Evaluation results with different chunk durations using box plots.

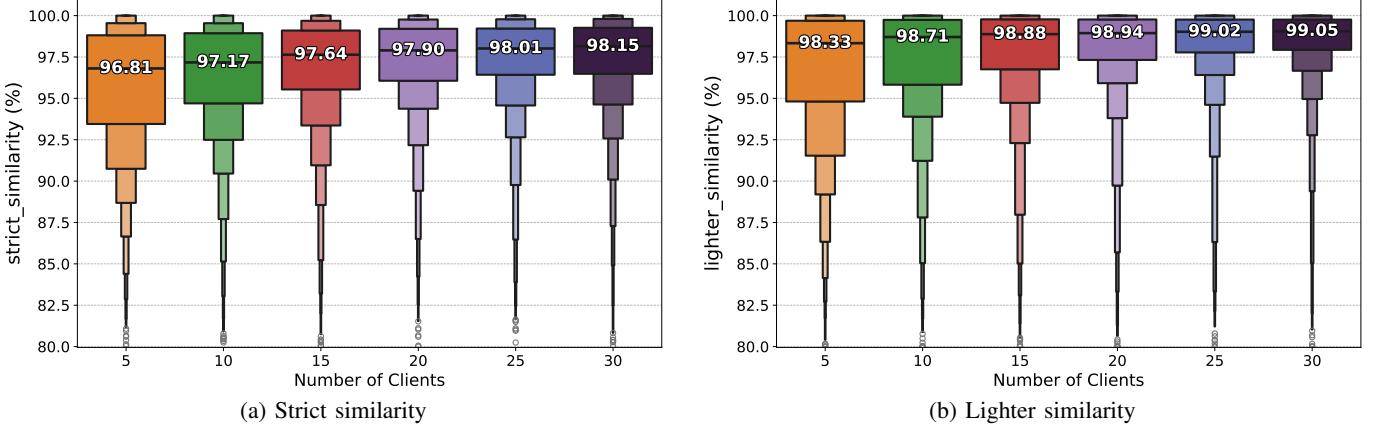


Figure 4: The similarity score distribution per client across all datasets using boxen plots.

Although WER is the *de facto* standard for evaluating ASR accuracy, it does not always reliably capture performance in real-time scenarios. In particular, it penalizes formatting issues—such as punctuation and casing—even when the transcription remains semantically accurate.

Nevertheless, **SWIM** maintains low WER across datasets and languages, demonstrating its effectiveness with multilingual inputs and high transcription accuracy.

Similarity Distribution Plots. Figures 4a and 4b display the similarity score distribution by number of concurrent clients across all datasets. Scores are shown as percentages and visualized using boxen plots,¹¹ which provide a more detailed view of the data distribution.

The expected outcome is that similarity scores—computed using the QRatio metric—will be higher than corresponding WER values, as WER penalizes formatting inconsistencies (e.g., punctuation and casing) that do not impact semantic correctness. Both similarity scores are computed using the previously introduced QRatio metric. For each response, we extract the corresponding segment of the reference transcript using its start and end timestamps. We then calculate similarity between this reference segment and the generated response text. The *strict similarity* score uses the standard QRatio, while the *lighter similarity* score ignores differences in punctuation and casing. As expected, the strict similarity is slightly lower than the lighter score, but the difference is small, indicating that the system maintains strong semantic accuracy even under stringent comparison criteria.

Interestingly, similar to the trends seen in Figures 3c and 3d, similarity scores tend to improve as the number of concurrent clients increases. This is expected because longer processing times allow more audio to accumulate in each dedicated service buffer, giving the model more context when inferring each clip within the shared buffer.

Finally, we observe that both similarity scores remain relatively stable across datasets, with only minor variations. As shown in Fig. 4, the median similarity scores exceed 96% for the strict metric and 98% for the lighter one, regardless of the number of concurrent clients. This indicates that **SWIM**

¹¹Boxen plots extend traditional box plots by adding additional boxes to better capture the distribution's shape.

consistently maintains high semantic accuracy across diverse datasets and languages—a crucial requirement for real-time ASR systems.

Linguistic Delay Distribution Plots. Figure 5a shows the delay distribution per language for the *Google Fleurs* dataset, while Figure 5b presents the same for the *Multilingual LibriSpeech* dataset. Both use violin plots, which combine box plots with kernel density estimation to visualize delay distributions. Thanks to **SWIM**'s client synchronization mechanism, delays are expected to remain consistent across languages, as they are primarily influenced by model processing time.

The most notable observation is that the distributions are similar across languages, with only minor variations in the median delay. Moreover, the relatively narrow spread of the distributions denotes stable and predictable latency behavior, which is crucial for real-time applications. These results highlight **SWIM** generalizes effectively across diverse linguistic contexts without sacrificing responsiveness, underscoring its potential utility in multilingual speech processing environments.

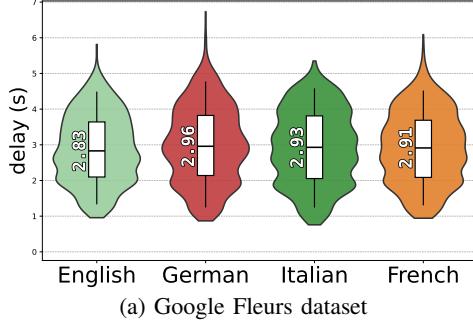
Linguistic WER Distribution Plots. Figures 6a and 6b show WER distribution across languages in the same settings.

As expected, WER is slightly influenced by the linguistic characteristics of each language. English, for instance, typically yields the lowest WER, while languages with diacritics or complex orthographic rules may show higher error rates.

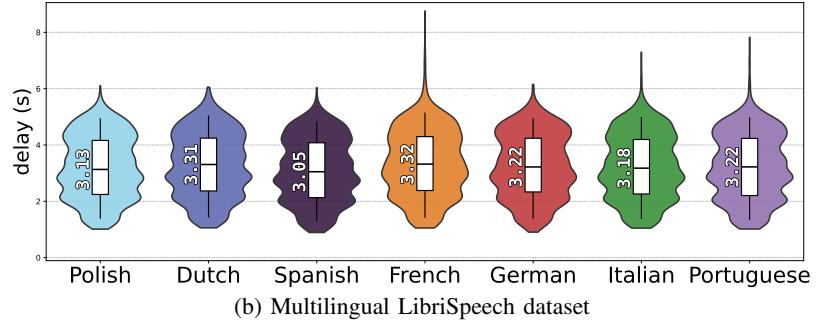
Nonetheless, the plots reveal that WER distributions remain consistent across languages, with only minor differences in median values. The small variations can be attributed to distribution tails and outliers. This indicates that languages rich in diacritics or with complex orthographies do not significantly degrade **SWIM**'s transcription accuracy.

Unlike whisper-streaming, which already demonstrates high accuracy, **SWIM** maintains comparable WER across languages—including those with complex phonetic structures—even as the number of concurrent clients increases.

Hypothesis vs. Confirmation Delay Plot. Figure 7 shows the mean delays—visualized as a bar plot—for both hypothesis and confirmation responses on the *Multilingual LibriSpeech* dataset, using a fixed chunk duration of 1.0s. The purpose of this plot is to illustrate that, beyond confirmed responses—which represent finalized, immutable

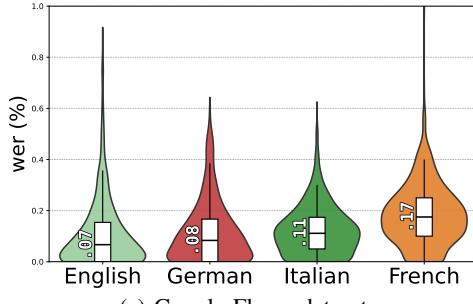


(a) Google Fleurs dataset

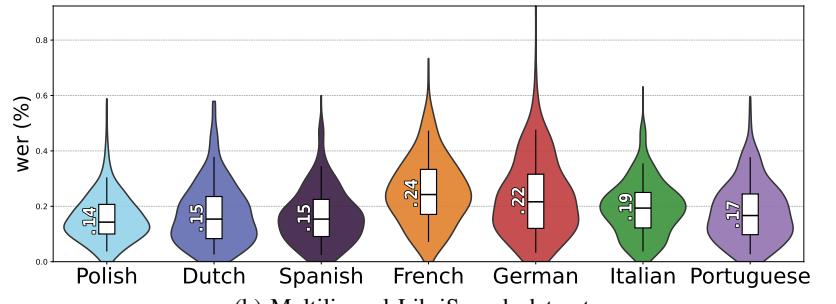


(b) Multilingual LibriSpeech dataset

Figure 5: Delay per language across the Multilingual LibriSpeech dataset and Google Fleurs dataset using violin plots.



(a) Google Fleurs dataset



(b) Multilingual LibriSpeech dataset

Figure 6: WER per language across the Multilingual LibriSpeech dataset and Google Fleurs dataset using violin plots.

transcriptions—hypothesis responses can provide preliminary transcriptions earlier in the pipeline. Thanks to the local agreement mechanism, hypothesis responses serve as early approximations of the final confirmed outputs. These intermediate results are often sufficiently accurate to be used for immediate feedback to users. This distinction is especially valuable in real-time applications where low-latency feedback is critical, such as live captioning or when supplying incremental input to *large language models* in interactive agent systems.

D. Concluding Remarks

Whisper-Streaming [32], in an English-only setting with a single fixed client, achieves a WER of approximately 8.2% and an average transcription delay of about 3.4 seconds. In comparison, **SWIM** scales to 5 concurrent clients while maintaining a similar WER (see Tables 3c and 3d on the *LibriSpeech-Long dataset*) and reducing average delay to roughly 2.4 seconds. Even at 15 concurrent clients, **SWIM** sustains stable latency and accuracy comparable to Whisper-Streaming. Furthermore, **SWIM** maintains low delay and high transcription accuracy—measured by WER and similarity scores—across diverse datasets and multilingual scenarios with up to 20 concurrent clients. These results demonstrate **SWIM**’s effective scalability, significantly increasing throughput without sacrificing transcription quality.

VI. THREATS TO VALIDITY

In our discussion, we follow Wohlin et al. [77]’s taxonomy.

A. Construct Validity

Transcription Quality Metrics. Our evaluation also relies on standard ASR metrics such as WER to assess transcription

quality. However, these metrics may not fully reflect the nuances of real-time multilingual transcription, especially in cases involving code-switching, domain-specific vocabulary, or varied audio conditions across clients.

Real-Time Performance Metrics. We use latency and throughput to evaluate real-time performance. However, the definition of “real-time” varies by application. Our processing thresholds may not meet the stricter requirements of scenarios like live broadcasting or emergency response systems, where sub-second latency is critical.

Scalability Assessment. Our evaluation measures scalability by the number of concurrent clients the system can support. This may not fully reflect real-world conditions, where clients vary in audio quality, language, and network stability.

Mitigation Strategies. To address these threats to construct validity, we adopted several strategies: 1) for transcription quality, we complement standard WER metrics with our QRatio-based local agreement mechanism and a fallback confirmation strategy within the adaptive hypothesis buffer. This enables more nuanced handling of multilingual and code-switched input, especially in noisy or ambiguous contexts; 2) for real-time performance, our latency threshold applies only to confirmed segments, while unconfirmed portions are updated dynamically using word-level indicators. This better reflects practical expectations for streaming ASR systems; and 3) for scalability, **SWIM** was evaluated on four datasets (including two multilingual) recorded in both studio and real-world conditions, covering diverse speakers and varying audio quality. This helps ensure that our results generalize to real-world deployments rather than idealized settings.

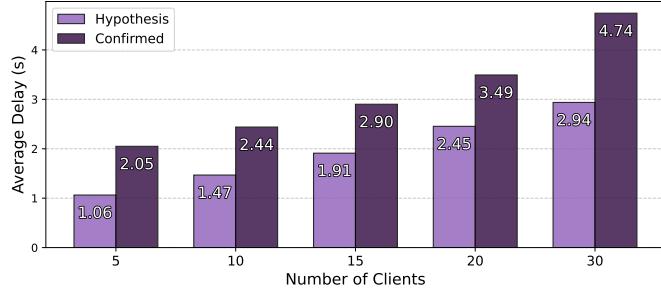


Figure 7: Hypothesis delay vs Confirmation delay with fixed chunk duration of 1.0s on Multilingual LibriSpeech dataset.

B. Internal Validity

Whisper Model Limitations. The underlying Whisper model has known limitations when handling silent audio segments, occasionally producing hallucinated or repeated phrases. These artifacts can inflate error rates and may not accurately reflect the performance of the **SWIM** system architecture.

Audio Quality Variations. Inconsistent audio quality across test samples may affect our evaluation. Background noise, microphone fidelity, speaker accents, and recording conditions can introduce biases, favoring certain client configurations or audio types.

Mitigation Strategies. To address these threats to internal validity, we adopted several strategies: 1) to mitigate limitations of the Whisper model, we support optional voice activity detection as a preprocessing step. This filters out non-speech segments, reducing hallucinations caused by silence; and 2) to handle audio quality variations, **SWIM** includes a fallback mechanism that activates after repeated local agreement failures, ensuring reliable output even under challenging acoustic conditions.

C. External Validity

Dataset Limitations. Our evaluation relies on selected datasets that may not capture the full diversity of real-world audio streams. If these datasets predominantly feature clean, studio-quality recordings or limited language combinations, the results may not generalize to noisy environments, telephone audio, or underrepresented languages in Whisper’s training data.

Client Configuration Constraints. Our experiments assume clients use uniform audio chunk durations (≤ 1 second) and fixed sampling rates (16kHz). However, real-world deployments may involve more diverse configurations, including varying chunk lengths and audio formats, which could impact system performance.

Deployment Environment. Our evaluation takes place in controlled server environments that may not capture the constraints of real-world deployments, such as limited computational resources, fluctuating network conditions, and varying system loads.

Mitigation Strategies. To address threats to external validity, we implemented several strategies: 1) to overcome dataset

limitations, our evaluation covers diverse audio conditions—including noisy environments, telephony audio, and multiple languages—beyond clean, studio-quality recordings to improve generalizability; 2) concerning client configuration constraints, **SWIM** supports variable chunk sizes, with some settings yielding better performance. Although a 16kHz sampling rate is currently required, we are extending support for automatic upsampling and downsampling to increase compatibility with different input formats; and 3) while testing occurred in controlled environments, **SWIM** is actively deployed in production by **REDACTED**, serving both internal and external clients, confirming its real-world effectiveness.

D. Conclusion Validity

Baseline Comparison Limitations. Comparisons with existing systems such as Whisper-Streaming and Whispy may be affected by differences in implementation, evaluation settings, or optimization goals, which can make direct performance comparisons potentially misleading.

Statistical Analysis Limitations. Our analysis is limited by sample size, which may affect the reliability of our conclusions. Small samples can yield unstable performance estimates and may not fully capture true variability in system behavior.

Mitigation Strategies. To address threats to conclusion validity, we adopted several strategies: 1) although implementation differences can impact baseline comparisons, our primary contribution is a novel architectural approach to multi-client ASR, which can be adopted or adapted by other systems regardless of their implementation. This highlights methodological innovation over direct performance advantage; and 2) to strengthen statistical validity, we evaluated **SWIM** on datasets with long audio samples (up to 5 minutes) and hundreds of entries, tested across varying numbers of concurrent clients. This reduces small-sample effects and offers a more comprehensive assessment of system behavior under diverse load conditions.

VII. CONCLUSION

We present **SWIM**, a novel multi-client real-time ASR system that enables a single Whisper model instance to process multiple concurrent multilingual streams with low latency and high accuracy. Unlike multi-instance setups, **SWIM** achieves model-level parallelization through a shared buffer merging strategy, optimizing resource utilization while preserving transcription quality. Key innovations include an adaptive hypothesis buffer with QRatio-based agreement and a fallback confirmation mechanism to manage linguistic variability. **SWIM** outperforms existing single-client methods and provides scalable, cost-effective ASR suitable for enterprise deployment.

REFERENCES

- [1] A. S. Sharma and R. Bhalley, “ASR—A Real-Time Speech Recognition on Portable Devices,” in *Proceedings of the 2nd International Conference on Advances in Computing, Communication, & Automation*

(ICACCA'16). Bareilly, India: IEEE, Sep. 2016, pp. 1–4.

[2] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, “Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning,” in *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*, M. Ranzato, Ed., Vancouver, BC, Canada, Apr./May 2018.

[3] S. Alharbi, M. Alrazgan, A. Alrashed, T. Alnomasi, R. Almojel, R. Alharbi, S. Alharbi, S. Alturki, F. Alshehri, and M. Almojil, “Automatic Speech Recognition: Systematic Literature Review,” *IEEE Access*, vol. 9, pp. 131 858–131 876, 2021.

[4] B. Lecouteux, M. Vacher, and F. Portet, “Distant Speech Recognition in a Smart Home: Comparison of Several Multisource ASRs in Realistic Conditions,” in *Proceedings of the 12th INTERSPEECH Conference (INTERSPEECH'11)*, R. Pieraccini, Ed. Firenze, Italy: ISCA, Aug. 2011, pp. 2273–2276.

[5] A. Ram, R. Prasad, C. Khatri, A. Venkatesh, R. Gabriel, Q. Liu, J. Nunn, B. Hedayatnia, M. Cheng, A. Nagar, E. King, K. Bland, A. Wartick, Y. Pan, H. Song, S. Jayadevan, G. Hwang, and A. Pettigrue, “Conversational AI: The Science Behind the Alexa Prize,” *arXiv e-prints*, vol. arXiv:1801.03604, pp. 1–18, Jan. 2018.

[6] M. Gerosa, D. Giuliani, S. Narayanan, and A. Potamianos, “A Review of ASR Technologies for Children’s Speech,” in *Proceedings of the 2nd Workshop on Child, Computer and Interaction (WOCCI'09)*. Cambridge, MA, USA: ACM, Nov. 2009, pp. 7:1–7:8.

[7] A. Tjandra, S. Sakti, and S. Nakamura, “Attention-Based wav2text with Feature Transfer Learning,” in *Proceedings of the Automatic Speech Recognition and Understanding Workshop (ASRU'17)*. Okinawa, Japan: IEEE, Jan. 2017, pp. 309–315.

[8] P. Dubey and B. Shah, “Deep Speech Based End-To-End Automated Speech Recognition (ASR) for Indian-English Accents,” *arXiv e-prints*, vol. arXiv:2204.00977, pp. 1–6, Apr. 2022.

[9] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, “Deep Speech: Scaling up End-to-End Speech Recognition,” *arXiv e-prints*, vol. arXiv:1412.5567, pp. 1–12, Dec. 2014.

[10] R. Prabhavalkar, T. Hori, T. N. Sainath, R. Schlüter, and S. Watanabe, “End-to-End Speech Recognition: A Survey,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 32, pp. 325–351, 2023.

[11] J. Li, “Recent Advances in End-to-End Automatic Speech Recognition,” *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, pp. 1–64, Apr. 2022.

[12] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “WaveNet: A Generative Model for Raw Audio,” in *Proceedings of the 9th Workshop on Speech Synthesis (SSW'16)*. Sunnyvale, USA: ISCA, Sep. 2016, p. 125.

[13] S. Furui, “50 Years of Progress in Speech and Speaker Recognition Research,” *Transactions on Computer and Information Technology*, vol. 1, no. 2, pp. 64–74, Nov. 2005.

[14] M. Gales and S. Young, “The Application of Hidden Markov Models in Speech Recognition,” *Foundations and Trends® in Signal Processing*, vol. 1, no. 3, pp. 195–304, Feb. 2008.

[15] H. Aldarmaki, A. Ullah, S. Ram, and N. Zaki, “Unsupervised Automatic Speech Recognition: A Review,” *Speech Communication*, vol. 139, pp. 76–91, Apr. 2022.

[16] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations,” in *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS'20)*, M. Ranzato, R. Hadsell, M. F. Balcan, and H.-T. Lin, Eds., vol. 33. Vancouver, BC, Canada: Curran Associates, Inc., Dec. 2020, pp. 1–12.

[17] G. Chen, S. Chai, G.-B. Wang, J. Du, W.-Q. Zhang, C. Weng, D. Su, D. Povey, J. Trmal, J. Zhang, M. Jin, S. Khudanpur, S. Watanabe, S. Zhao, W. Zou, X. Li, X. Yao, Y. Wang, Z. You, and Z. Yan, “GigaSpeech: An Evolving, Multi-Domain ASR Corpus with 10,000 Hours of Transcribed Audio,” in *Proceedings of the 22nd INTERSPEECH Conference (INTERSPEECH'21)*, L. Burget, L. Lamel, O. Scharenborg, and P. Motlicek, Eds. Brno, Czech Republic: ISCA, Aug. 2021, pp. 4376–4380.

[18] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavy, and I. Sutskever, “Robust Speech Recognition via Large-Scale Weak Supervision,” in *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*. Honolulu, Hawaii: PMLR, Jul. 2023, pp. 28 492–28 518.

[19] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu, “Deep Speech 2: End-to-End Speech Recognition in English and Mandarin,” in *Proceedings of the 33rd International Conference on Machine Learning (ICML'16)*, N. Balcan and K. Weinberger, Eds., vol. 48. New York, NY, USA: PMLR, Jun. 2016, pp. 173–182.

[20] A. Morris, A. Hagen, H. Glotin, and H. Bourlard, “Multi-Stream Adaptive Evidence Combination for Noise Robust ASR,” *Speech Communication*, vol. 34, no. 1-2, pp. 25–40, 2001.

[21] R. Li, X. Wang, S. H. Mallidi, S. Watanabe, T. Hori, and H. Hermansky, “Multi-Stream End-to-End Speech Recognition,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 6, pp. 1026–1039, Jun. 2019.

[22] P. N. Garner, J. Dines, T. Hain, A. El Hannani, M. Karafiat, D. Korchagin, M. Lincoln, V. Wan, and L. Zhang, “Real-Time ASR from Meetings,” in *Pro-*

ceedings of the 9th INTERSPEECH Conference (INTERSPEECH'09). Brighton, United Kingdom: ISCA, Sep. 2023, pp. 2119–2122.

[23] C. Arriaga, A. Pozo, J. Conde, and A. Alonso, “Evaluation of Real-Time Transcriptions Using End-to-End ASR Models,” *arXiv preprint arXiv:2409.05674*, 2024.

[24] A. V. Ivanov, P. L. Lange, D. Suendermann-Oeft, V. Ramanarayanan, Y. Qian, Z. Yu, and J. Tao, “Speed vs. Accuracy: Designing an Optimal ASR System for Spontaneous Non-Native Speech in a Real-Time Application,” in *Proceedings of the 7th International Workshop on Spoken Dialogue Systems (IWSDS'16)*, ser. Lecture Notes in Electrical Engineering 427. Saariselkä, Finland: Springer, Jan. 2016, pp. 1–12.

[25] S. Ahmed, A. R. Chowdhury, K. Fawaz, and P. Ramanathan, “Preech: A System for Privacy-Preserving Speech Transcription,” in *Proceedings of the 29th USENIX Security Symposium (USENIX Security'20)*, S. Capkun and F. Roesner, Eds. Boston, MA, USA: USENIX, Aug. 2020, pp. 2703–2720.

[26] H. Abdullah, K. Warren, V. Bindschaedler, N. Papernot, and P. Traynor, “SoK: The Faults in our ASRs,” in *Proceedings of the 42nd IEEE Symposium on Security and Privacy (SP'21)*, A. Oprea and T. Holz, Eds. San Francisco, CA, USA: IEEE, May 2021, pp. 730–747.

[27] A.-L. Georgescu, A. Pappalardo, H. Cucu, and M. Blott, “Performance vs. Hardware Requirements in State-of-the-Art Automatic Speech Recognition,” *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2021, no. 1, pp. 28:1–28:30, Jul. 2021.

[28] F. M. Ramirez, L. Chkhietiani, A. Ehrenberg, R. McHardy, R. Botros, Y. Khare, A. Vanzo, T. Peyash, G. Oexle, M. Liang *et al.*, “Anatomy of Industrial Scale Multilingual ASR,” *arXiv preprint arXiv:2404.09841*, 2024.

[29] H. Zhu, J. Wang, G. Cheng, P. Zhang, and Y. Yan, “Decoupled Federated Learning for ASR with Non-IID Data,” in *Proceedings of the 23rd INTERSPEECH Conference (INTERSPEECH'22)*, K. Lee, L. Lamel, M. Hasegawa-Johnson, K. Livescu, and O. Kang, Eds. Incheon, South Korea: ISCA, Aug. 2023, pp. 2628–2632.

[30] S. Basak, H. Agrawal, S. Jena, S. Gite, M. Bachute, B. Pradhan, and M. Assiri, “Challenges and Limitations in Speech Recognition Technology: A Critical Review of Speech Signal Processing Algorithms, Tools and Systems,” *Computer Modeling in Engineering and Sciences*, vol. 135, no. 2, pp. 1053–1089, Oct. 2022.

[31] A. Bevilacqua, P. Saviano, A. Amirante, and S. P. Romano, “Whispy: Adapting STT Whisper Models to Real-Time Environments,” *arXiv preprint arXiv:2405.03484*, 2024.

[32] D. Macháček, R. Dabre, and O. Bojar, “Turning Whisper into Real-Time Transcription System,” in *Proceedings of the 13th International Joint Conference on Natural Language Processing (IJCNLP'23)*, S. Saha and H. Sujaini, Eds. Bali, Indonesia: ACL, Nov. 2023, pp. 17–24.

[33] P. Polák, N.-Q. Pham, T. N. Nguyen, D. Liu, C. Mullov, J. Niehues, O. Bojar, and A. Waibel, “CUNI-KIT System for Simultaneous Speech Translation Task at IWSLT 2022,” in *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT'22)*, E. Salesky, M. Federico, and M. Costa-jussà, Eds. Dublin, Ireland: ACL, May 2022, pp. 277–285.

[34] V. Pratap, Q. Xu, J. Kahn, G. Avidov, T. Likhomanenko, A. Hannun, V. Liptchinsky, G. Synnaeve, and R. Collobert, “Scaling Up Online Speech Recognition Using ConvNets,” in *Proceedings of the 21st Annual Conference of the International Speech Communication Association (INTERSPEECH'20)*, B. Xu and T. F. Zheng, Eds. Shanghai, China: ISCA, Oct. 2020, pp. 3376–3380.

[35] X. Cui, S. Lu, and B. Kingsbury, “Federated Acoustic Modeling for Automatic Speech Recognition,” in *Proceedings of the 46th International Conference on Acoustics, Speech and Signal Processing (ICASSP'21)*, T. Davidson and D. Yu, Eds. Toronto, Canada: IEEE, Jun. 2021, pp. 6748–6752.

[36] Y. Zhang, L. Xu, A. Mendoza, G. Yang, P. Chinpruthiwong, and G. Gu, “Life After Speech Recognition: Fuzzing Semantic Misinterpretation for Voice Assistant Applications,” in *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS'19)*. San Diego, CA, USA: The Internet Society, Feb. 2019, pp. 1–15.

[37] K. Kuhn, V. Kersken, B. Reuter, N. Egger, and G. Zimmermann, “Measuring the Accuracy of Automatic Speech Recognition Solutions,” *ACM Transactions on Accessible Computing*, vol. 16, no. 4, pp. 25:1–25:23, Jan. 2024.

[38] N. Jamal, S. Shanta, F. Mahmud, and M. Sha'abani, “Automatic Speech Recognition (ASR) Based Approach for Speech Therapy of Aphasic Patients: A Review,” in *Proceedings of the International Conference on Electrical and Electronic Engineering (IC3E'17)*. Johor Bahru, Malaysia: AIP, Aug. 2017.

[39] M. B. Mustafa, F. Rosdi, S. S. Salim, and M. U. Mughal, “Exploring the Influence of General and Specific Factors on the Recognition Accuracy of an ASR System for Dysarthric Speaker,” *Expert Systems with Applications*, vol. 42, no. 8, pp. 3924–3932, May 2015.

[40] S. Latif, J. Qadir, A. Qayyum, M. Usama, and S. Younis, “Speech Technology for Healthcare: Opportunities, Challenges, and State of the Art,” *IEEE Reviews in Biomedical Engineering*, vol. 14, pp. 342–356, 2021.

[41] D. Loakes, “Does Automatic Speech Recognition (ASR) Have a Role in the Transcription of Indistinct Covert Recordings for Forensic Purposes?” *Frontiers in Communication*, vol. 7, Aug. 2022.

[42] M. Komter, “Institutional and Academic Transcripts of Police Interrogations,” *Frontiers in Communication*, vol. 7, Aug. 2022.

[43] L. R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[44] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. P. Pang,

“Conformer: Convolution-Augmented Transformer for Speech Recognition,” in *Proceedings of the 21st Annual Conference of the International Speech Communication Association (INTERSPEECH’20)*, B. Xu and T. F. Zheng, Eds. Shanghai, China: ISCA, Oct. 2020, pp. 5036–5040.

[45] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S.-y. Chang, K. Rao, and A. Grusenstein, “Streaming End-to-end Speech Recognition for Mobile Devices,” in *Proceedings of the 44th International Conference on Acoustics, Speech, and Signal Processing (ICASSP’19)*. Brighton, United Kingdom: IEEE, May 2019, pp. 6381–6385.

[46] M. B. Hoy, “Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants,” *Medical Reference Services Quarterly*, vol. 37, no. 1, pp. 81–88, Jan./Mar. 2018.

[47] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, “Transformer in Transformer,” in *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS’21)*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Lian, and J. Wortman Vaughan, Eds., vol. 34. Virtual: Curran Associates, Inc., Dec. 2021, pp. 15 908–15 919.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. M. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention Is All You Need,” in *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS’17)*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Long Beach, CA, USA: Curran Associates, Inc., Dec. 2017, pp. 6000–6010.

[49] V. Pratap, A. Tjandra, B. Shi, P. Tomasello, A. Babu, S. Kundu, A. Elkahky, Z. Ni, A. Vyas, M. Fazel-Zarandi, A. Baevski, Y. Adi, X. Zhang, W.-N. Hsu, A. Conneau, and M. Auli, “Scaling Speech Technology to 1,000+ Languages,” *Journal of Machine Learning Research*, vol. 25, pp. 1–52, Mar. 2024.

[50] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An ASR Corpus Based on Public Domain Audio Books,” in *Proceedings of the 40th International Conference on Acoustics, Speech, and Signal Processing (ICASSP’15)*, D. Gray and D. Cochran, Eds. Brisbane, Australia: IEEE, Apr. 2015, pp. 5206–5210.

[51] S. J. Park, J. Salazar, A. Jansen, K. Kinoshita, Y. M. Ro, and R. Skerry-Ryan, “Long-Form Speech Generation with Spoken Language Models,” in *Proceedings of the 42nd International Conference on Machine Learning (ICML’25)*. Vancouver, BC, Canada: PMLR, Jul. 2025, pp. 48 245–48 261.

[52] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert, “MLS: A Large-Scale Multilingual Dataset for Speech Research,” in *Proceedings of the 21st Annual Conference of the International Speech Communication Association (INTERSPEECH’20)*, B. Xu and T. F. Zheng, Eds. Shanghai, China: ISCA, Oct. 2020, pp. 2757–2761.

[53] A. Conneau, M. Ma, S. Khanuja, Y. Zhang, V. Axelrod, S. Dalmia, J. Riesa, C. Rivera, and A. Bapna, “FLEURS: FEW-Shot Learning Evaluation of Universal Representations of Speech,” in *Proceedings of IEEE Spoken Language Technology Workshop (SLT’22)*, S. Watanabe, S. Khudanpur, J. Hirschberg, M. Saracclar, and M. Delcroix, Eds. Doha, Qatar: IEEE, Jan. 2023, pp. 798–805.

[54] A. Ali and S. Renals, “Word Error Rate Estimation for Speech Recognition: e-WER,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL’18)*, I. Gurevych and Y. Miyao, Eds. Melbourne, Australia: ACM, Jul. 2018, pp. 29–24.

[55] D. Klakow and J. Peters, “Testing the Correlation of Word Error Rate and Perplexity,” *Speech Communication*, vol. 38, no. 1–2, pp. 19–28, Sep. 2002.

[56] Y.-Y. Wang, A. Acero, and C. Chelba, “Is Word Error Rate a Good Indicator for Spoken Language Understanding Accuracy,” in *Proceedings of the 7th Workshop on Automatic Speech Recognition and Understanding (ASRU’03)*. St. Thomas, U.S. Virgin Islands: IEEE, Nov./Dec. 2003, pp. 577–582.

[57] I. Guyon, J. Makhoul, R. Schwartz, and V. Vapnik, “What Size Test Set Gives Good Error Rate Estimates?” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 52–64, Jan. 1998.

[58] C. Park, H. Kang, and T. Hain, “Character Error Rate Estimation for Automatic Speech Recognition of Short Utterances,” in *Proceedings of the 32nd European Signal Processing Conference (EUSIPCO’24)*. Lyon, France: IEEE, Aug. 2024, pp. 131–135.

[59] R. Sawata, Y. Kashiwagi, and S. Takahashi, “Improving Character Error Rate is Not Equal to Having Clean Speech: Speech Enhancement for ASR Systems with Black-Box Acoustic Models,” in *Proceedings of the 47th International Conference on Acoustics, Speech and Signal Processing (ICASSP’22)*, W.-S. Gan and K.-K. Ma, Eds. Singapore, Singapore: IEEE, May 2022, pp. 991–995.

[60] C. Spiccia, A. Augello, G. Pilato, and G. Vassallo, “Semantic Word Error Rate for Sentence Similarity,” in *Proceedings of the 10th International Conference on Semantic Computing (ICSC’16)*, T. Li, A. Scherp, D. Ostrowski, and W. Wang, Eds. Laguna Hill, CA, USA: IEEE, Feb. 2016, pp. 266–269.

[61] S. Raybaud, D. Langlois, and K. Smaïli, ““This Sentence Is Wrong.” Detecting Errors in Machine-Translated Sentences,” *Machine Translation*, vol. 25, pp. 1–34, Jul. 2011.

[62] Y. Li and B. Liu, “A Normalized Levenshtein Distance Metric,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1091–1095, Jun. 2007.

[63] W. J. Heeringa, “Measuring Dialect Pronunciation Differences Using Levenshtein Distance,” PhD Thesis, University of Groningen, Groningen, Netherlands, 2004.

[64] G.-X. Shi, W.-Q. Zhang, G.-B. Wang, J. Zhao, S.-Z. Chai, and Z.-Y. Zhao, “Timestamp-Aligning and Keyword-Biasing End-to-End ASR Front-End for a KWS System,” *EURASIP Journal on Audio, Speech and Music Processing*, pp. 21:1–21:14, Jul. 2021.

- [65] T. Han, W. Xie, and A. Zisserman, “Temporal Alignment Networks for Long-Term Video,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR’22)*, K. Dana, G. Hua, S. Roth, D. Samaras, and R. Singh, Eds. New Orleans, LA, USA: IEEE, Jun. 2022, pp. 2896–2906.
- [66] J. Li, Y. Meng, Z. Wu, H. Meng, Q. Tian, Y. Wang, and Y. Wang, “Neufa: Neural Network Based End-to-End Forced Alignment with Bidirectional Attention Mechanism,” in *Proceedings of the 47th International Conference on Acoustics, Speech and Signal Processing (ICASSP’22)*, W.-S. Gan and K.-K. Ma, Eds. Singapore, Singapore: IEEE, May 2022, pp. 8007–8011.
- [67] X. Chu, Y. He, K. Chakrabarti, and K. Ganjam, “TEGRA: Table Extraction by Global Record Alignment,” in *Proceedings of the 41st International Conference on Management of Data (SIGMOD’15)*, S. B. Davison and Z. Ives, Eds. Melbourne, Australia: ACM, May/Jun. 2015, pp. 1713–1728.
- [68] R. Wang, Z. Xu, and F. X. Lin, “Efficient Whisper on Streaming Speech,” *arXiv preprint arXiv:2412.11272*, 2024.
- [69] M. Bain, J. Huh, T. Han, and A. Zisserman, “WhisperX: Time-Accurate Speech Transcription of Long-Form Audio,” in *Proceedings of the 24th INTERSPEECH Conference (INTERSPEECH’23)*, S. King, K. Knill, and P. Wagner, Eds. Dublin, Ireland: ISCA, Aug. 2023, pp. 4489–4493.
- [70] G. Navarro, “A Guided Tour to Approximate String Matching,” *Computing Surveys*, vol. 33, no. 1, pp. 31–88, Mar. 2001.
- [71] M. M. Deza and E. Deza, *Encyclopedia of Distances*. Springer, 2009.
- [72] P. H. Sellers, “On the Theory and Computation of Evolutionary Distances,” *Journal on Applied Mathematics*, vol. 26, no. 4, pp. 787–793, 1974.
- [73] V. I. Levenshtein, “Binary Codes Capable of Correcting Deletions, Insertions and Reversals,” *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, Feb. 1966.
- [74] G. Dimauro, D. Caivano, V. Bevilacqua, F. Girardi, and V. Napoletano, “VoxTester, Software for Digital Evaluation of Speech Changes in Parkinson Disease,” in *Proceedings of the 11th International Symposium on Medical Measurements and Applications (MeMeA’16)*. Benevento, Italy: IEEE, May 2016, pp. 1–6.
- [75] G. Dimauro, V. Di Nicola, V. Bevilacqua, D. Caivano, and F. Girardi, “Assessment of Speech Intelligibility in Parkinson’s Disease Using a Speech-To-Text System,” *IEEE Access*, vol. 5, pp. 22 199–22 208, Nov. 2017.
- [76] G. Dimauro and F. Girardi, “Italian Parkinson’s Voice and Speech,” 2019.
- [77] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer, 2012.



Federico Bruzzone is currently a Ph.D. student in Computer Science at Università degli Studi di Milano, Italy. He was born in 2000 and since he was a child he has been passionate about computer science and music. He got his bachelor degree in Musical Computer Science, the master degree in Computer Science and currently he is involved in the research activity of the ADAPT Lab. His main research interests are (but are not limited to) programming languages and compilers, software maintenance and evolution. For any question he can be contacted at federico.bruzzone@unimi.it



Walter Cazzola is currently a Full Professor in the Computer Science Department of the Università degli Studi di Milano, Italy and the Chair of the ADAPT laboratory. He designed the mChaRM framework, @Java, [a]C#, Blueprint programming languages and he is currently involved in the designing and development of the Neverlang language workbench. He also designed the JavAdaptor dynamic software updating framework and its front-end FiGA. His research interests include software maintenance, evolution and comprehension, programming methodologies and languages. He served on the program committees or editorial boards of the most important conferences and journals about his research topics. He is associate editor for the Journal of Computer Languages published by Elsevier. More information are available at <https://cazzola.di.unimi.it> and he can be contacted at cazzola@di.unimi.it for any question.



Matteo Brancaleoni is a Senior Staff Engineer at VoiSmart Srl, Italy, where he develops unified communication systems with a focus on VoIP infrastructures and videoconferencing platforms. His expertise includes real-time communications, telephony protocols, WebRTC, and distributed architectures. His research and professional activities center on designing scalable, reliable communication frameworks and exploring their extension with artificial intelligence. He is particularly interested in protocol design, communication middleware, and AI integration into real-time distributed systems. He can be contacted at matteo.brancaleoni@voismart.it



Dario Pellegrino is currently a Master of Science student in Computer Science. He obtained his Bachelor’s degree in Computer Science with a thesis project developed in collaboration with the ADAPT Lab and VoiSmart, focusing on the SWIM project. Recently, he started working as a Junior Software Engineer at VoiSmart, contributing to research and development activities on AI-based digital agents and assistants in the domain of telephony and PBX systems. His main research interests include software engineering and artificial intelligence, with a particular enthusiasm for both low-level aspects such as machine learning and the application of AI to software engineering. He can be contacted at dario.pellegrino@voismart.it