Thinking While Speaking: Inference-Time Knowledge Transfer for Responsive and Intelligent Conversational Voice Agents

Vidya Srinivas*, **Zachary Englhardt***, **Vikram Iyer, Shwetak Patel**
Paul G. Allen School of Computer Science & Engineering
{vysri,zacharye}@cs.washington.edu

Abstract

Voice agents face a fundamental tension: the reasoning, retrieval, and tool use that make foundation models capable are iterative and slow, while conversational interaction demands responses on a millisecond timescale. Smaller, real-time models meet the latency bar but cannot match foundation models on complex tasks, leaving current voice agents to trade away either responsiveness or capability. We introduce *conversational infill*, where a small `Talker` model both immediately generates contextually grounded responses to hide the latency of an external `Reasoner` model and fluently integrates streamed `Reasoner` knowledge into its responses during inference. We curate a 290,571-example synthetic dataset spanning six domains and demonstrate that this task is learnable across seven widely used small language models ranging from 135M to 1.7B parameters[1]. Our system implementation, *ConvFill*, sustains millisecond-level time-to-first-response while closing the accuracy gap to within 6.3% of the corresponding frontier `Reasoner` performance. In a live user study ($n = 18$) with `Talker` deployments running on an Apple M2 SoC, participants rank ConvFill on par with frontier models overall, prefer it for retrieval-heavy tasks, and rate it significantly more responsive. These results show that conversational infill unlocks a new point on the latency–capability Pareto frontier, offering a practical path toward voice agents that are both responsive and highly capable.

1 Introduction

Advances in large language models (LLMs) have enabled increasingly capable task-oriented systems for applications such as customer service, virtual assistants, tutoring, and interactive software agents. However, voice-based conversational models consistently lag behind their text-only counterparts (Lin et al., 2025). This gap stems from a fundamental tension: the techniques that enhance text-based model capabilities, such as multi-step reasoning, tool calls, and retrieval, are inherently iterative and time-consuming, while spoken interaction demands responses on a millisecond timescale to maintain a seamless user experience (Jacoby et al., 2024). These pressures pull in opposite directions: advanced capabilities improve answers but erode the responsiveness that conversational interaction depends on (Lin et al., 2026).

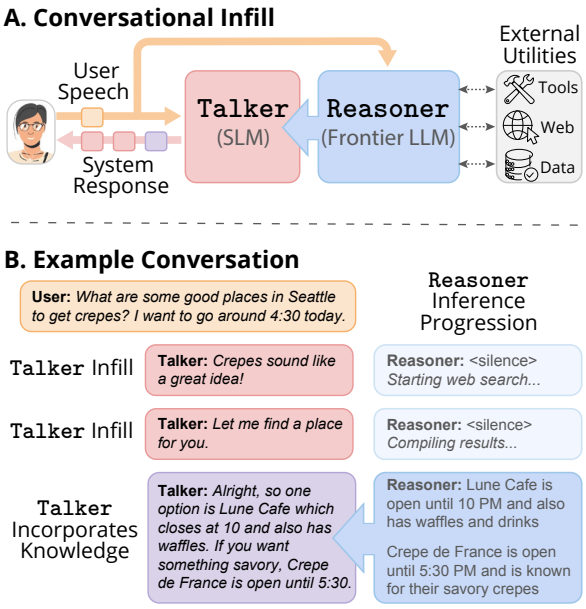

Figure 1: **Conversational infill.** **(A)** A lightweight `Talker` model (red) immediately starts responding while a `Reasoner` model (blue) performs higher-latency reasoning and tool calls. When the `Reasoner` responds, the `Talker` incorporates this information into subsequent responses (purple). **(B)** A conversation turn showing user speech, `Talker` response, and the background `Reasoner` inference progression. The third response phrase (purple) is conditioned on `Reasoner` context.

To bridge this gap, we introduce **conversational infill**, a language model collaboration task in which

*Equal contribution.
[1]Dataset: github.com/zenglhardt/convfill-dataset, Code and Models: github.com/vysri/conversational-infill

a lightweight Talker model interacts directly with the user, while a powerful Reasoner model guides it with knowledge during inference. Our approach is inspired at a high level by the Talker-Reasoner concept presented in Christakopoulou et al. (2024), but is distinct in formalization and implementation. The Talker-Reasoner division of labor allows the Talker to satisfy the millisecond-level responsiveness that spoken interaction requires while allowing the Reasoner to contribute its full capabilities without real-time constraints.

In our task, the Reasoner model is a frontier model able to handle multi-turn context and leverage reasoning and tool calls to generate knowledge chunks. These concise, non-conversational phrases prioritize the essential information needed to respond to a user utterance. The Talker model is a small (e.g., on-device) language model (SLM). The Talker model's primary role is to seamlessly transform knowledge chunks from the Reasoner into natural conversation. When no external knowledge is available, the Talker generates contextually grounded filler responses to hide Reasoner latency while maintaining conversational flow. For a task overview and representative interaction turn, see Figure 1.

Existing real-time voice systems fall into two categories. Cascaded systems combine separate automatic speech recognition (ASR), text-input language models, and text-to-speech (TTS), compounding latency and introducing gaps that break conversational flow. Full-duplex speech models (Défossez et al., 2024; Iribe et al., 2024; Gao et al., 2025) generate audio directly, reducing latency by bypassing ASR and TTS steps. In comparison to cascaded approaches, full-duplex systems often use a single LLM backbone, sacrificing flexibility, while real-time constraints limit their size (Wang et al., 2025a).

ConvFill, our end-to-end system implementing the conversational infill task, balances these tradeoffs. It keeps the flexibility of a cascaded design while avoiding the latency that usually comes with it. Because the user interacts only with the Talker, which receives Reasoner knowledge as it arrives, filler responses mask Reasoner inference time and the Talker never stalls waiting on the Reasoner. ConvFill's cascaded structure also keeps training tractable. The Talker takes transcriptions as input and therefore can be trained on text alone, without the resource-intensive audio modeling that full-duplex approaches require (Défossez et al., 2024; Cuervo and Marxer, 2024). Lastly, ConvFill is not tied to a single choice of Talker or Reasoner model, meaning that any capable Talker or Reasoner can be slotted in independently.

We formalize the conversational infill task and demonstrate its feasibility and utility via an interactive end-to-end system. Concretely, we:

1. Curate and release the ConvFill dataset, consisting of 290,571 rigorously validated training examples across six domains.
2. Show that the conversational infill task is learnable across seven SLMs spanning four model families (Qwen, Llama, Gemma, and SmolLM).
3. Demonstrate that Talker models retain millisecond-level time-to-first-response while closing the accuracy gap to within 6.3% of the corresponding Reasoner performance.
4. Build a voice-based, real-time ConvFill system deployed on an Apple M2 SoC and conduct a live evaluation ($n = 18$). Users rate ConvFill on par with frontier models for interaction quality and significantly better for perceived latency.

To foster further work on model collaboration for responsive, capable voice systems, we release all of our artifacts: the ConvFill dataset, our generation and validation pipeline, the training and inference code for the end-to-end ConvFill system, and the fine-tuned weights for all seven Talker models.

2 Hiding Latency in Conversations

Many techniques have been explored to compensate for latency in conversational systems. Prolonged silence, especially during real-time chat, results in interrupted conversational flow (Levinson and Torreira, 2015; Li and Chen, 2019). At a minimum, initial responses from conversational systems can acknowledge that the user has been heard, which is essential for responsive, natural-feeling conversation (Skantze, 2021). In the simplest case, a system responds with a fixed set of generic infill phrases, such as *"I'm thinking."* This is often extended by employing heuristics to select between different sets of infill phrases based on context (Mahmood et al., 2025; Gao et al., 2025).

A more recent class of proposed approaches leverages dynamic infill methods that make use of multiple models at inference time. If the lower-latency model provides incorrect information, the higher-capability model corrects it with a more definitive answer. Such corrections, however, come

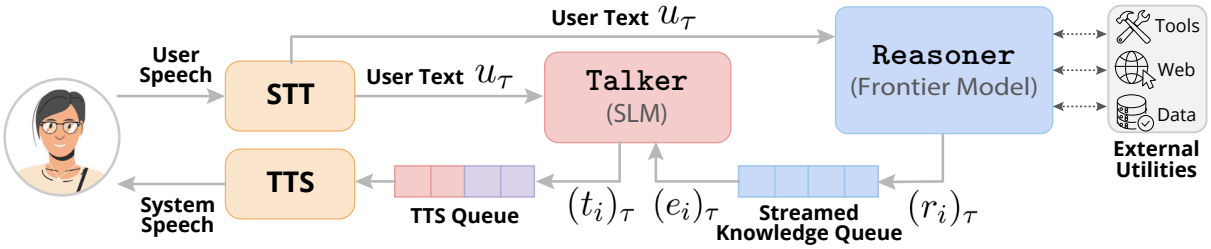

Figure 2: **ConvFill System Architecture.** User speech is transcribed using STT and is sent concurrently to the `Talker` and `Reasoner`. The `Talker` generates filler responses (red) while waiting for the `Reasoner` (blue) to push knowledge chunks into the streamed knowledge queue. When there is `Reasoner` knowledge present in the streamed knowledge queue, the `Talker` ingests it and produces conditioned responses (purple). All `Talker` responses are passed to the TTS queue and processed to yield spoken output.

at a cost: repeated revisions or inconsistencies—especially in spoken dialogue—have been shown to harm user trust, confidence, and engagement. Our architecture instead provides a consistent voice throughout full, multi-turn conversations by ensuring that the user only ever interacts with the `Talker`, which is trained to incorporate external knowledge rather than relaying it verbatim.

3 Task Formulation

We propose a Talker-Reasoner architecture (Christakopoulou et al., 2024) with a high-performance model as the `Reasoner` and a low-latency conversational model as the `Talker`. This results in a clear division of responsibilities between the two models. While the `Reasoner` uses tool calls, reasoning, and other advanced, yet high-latency, capabilities to address a user query, the `Talker` model must:

1. produce conversationally contingent phrases to hide `Reasoner` latency *when necessary*
2. consume raw, inference-time information from the `Reasoner` *when available* and transform it into fluent, contingent conversation

In the sections that follow, we refer to our particular system-specific implementation of the conversational infill task as *ConvFill*.

3.1 Task Setup

In the following text, `Talker` refers to a lightweight small language model (SLM) capable of running on consumer-grade hardware. `Reasoner` refers to a high-performance model that can manage multi-turn context and engage with tools and external knowledge sources. We refer to a *turn* as one exchange between a user and the `Talker` model. We use the notation $a \rightarrow b$ to mean that a entails b (Williams et al., 2018).

3.1.1 `Talker` Interface

Let τ denote the turn index in the conversation, where a conversation may consist of multiple turns. For turn τ, u_τ is the user utterance at turn τ. For this single turn, the `Talker` model responds in a sequence of phrases to form its full response, T_τ, where $T_\tau = (t_1, t_2, \ldots, t_p)_\tau$. h_τ denotes conversation history for turn τ, with a lookback of ℓ turns:

$$h_\tau(\ell) = (u_{\tau-\ell} \oplus T_{\tau-\ell}, \ldots, u_{\tau-1} \oplus T_{\tau-1})$$

where $\oplus$ denotes concatenation. Within a single turn τ, let $i \in [1, \ldots, p]$ denote the phrase index.

The `Talker` model operates at the turn level, generating response T_τ phrase by phrase. At each index i within the turn, the `Talker` conditions its generation on static and dynamic context. Static context includes conversation history $h_\tau(\ell)$, where ℓ is variable, and the current user utterance, u_τ. Dynamic context is the knowledge stream $(e_1, \ldots, e_i)_\tau$ from the `Reasoner` up to position i, and phrases $(t_1, \ldots, t_{i-1})_\tau$ already generated in the current turn.

3.1.2 `Reasoner` Interface

The `Reasoner` interface asynchronously produces external knowledge for the `Talker` at inference time. It generates a stream of external knowledge chunks, $E_\tau = (e_1, e_2, \ldots, e_p)_\tau$. The e_i may be either a knowledge chunk from the `Reasoner`, r_i, or a silence token, s_i. The r_i are semantically complete text phrases that convey the results of reasoning, retrieval, inference, tool calls, or other computation. Silence tokens s_i are inserted dynamically if the `Reasoner` model has not sent a knowledge chunk after a predetermined time. During inference, the `Reasoner` has access to the full conversation history $h_\tau(\tau - 1)$ and the current user utterance u_τ and uses these as context to generate each r_i. We assume that the `Reasoner` has a context window to store sufficient conversation history.

3.1.3 Inference-Time Collaboration

The Talker and Reasoner operate concurrently. The Reasoner streams knowledge chunks asynchronously, while the Talker continuously consumes from this stream. At each position i, the Talker observes the next element e_i.

If $e_i = r_i$ (a knowledge chunk), the Talker should generate t_i conditioned on r_i in addition to the conversation history and its prior phrases. If $e_i = s_i$ (a silence token), which signals that the Reasoner has not produced output within a threshold duration, the Talker should generate t_i conditioned only on the user utterance, conversation history, and prior phrases. Importantly $e_i = s_i$ and $e_i = r_i$ cannot occur at the same time. This is represented with notation in the equation below:

$$\begin{cases} h_\tau(\ell) \oplus [u_\tau \oplus (t_1, \cdots, t_{i-1})_\tau] \oplus (r_i)_\tau \to t_i & \text{if } e_i = r_i \\ h_\tau(\ell) \oplus [u_\tau \oplus (t_1, \cdots, t_{i-1})_\tau] \to t_i & \text{if } e_i = s_i \end{cases}$$

This creates continuous output while respecting the Reasoner's variable latency and preventing the Talker from stalling while waiting for knowledge. We enforce the constraint that $|T_\tau| = |E_\tau|$.

4 ConvFill Dataset

To train the Talker models, we require a corpus of example conversations with user utterances paired with factual responses and corresponding conversational rephrasings. In addition, as specified in item 1, maintaining grounding solely in past and current inputs is crucial as no conversational response should reference information that would not be visible to the Talker at inference time. We create training examples that meet these requirements through a two-part process involving synthetic data generation paired with a series of phrase, turn, and conversation-level checks to detect structural and semantic violations.

Generation. Conversations are structured as JSON objects containing a conversation array of turns. Turns carry a user utterance, a thoughts array beginning with 0–3 <sil> silence placeholders followed by statements corresponding to Reasoner model phrases, and a response array with phrases corresponding to Talker responses. Entries in locations corresponding to <sil> placeholders are conversationally contingent filler phrases, while entries corresponding to Reasoner phrases incorporate the meaning of these phrases in a form and style consistent with the prior conversation.

To ground conversations in realistic use cases, we first generate samples based on task-oriented dialogues from DSTC8 (Rastogi et al., 2020), retaining the labeled user utterances and using the labeled system responses as the contents of the Reasoner thoughts array. This array is padded with <sil> placeholders at the beginning of each turn; Talker responses are generated conditioned on this array. To further expand the variety of conversational topics, we construct a second corpus consisting of randomly seeded topics in the domains of general advice, assistant queries, event planning, customer service, education, and medicine. Before writing conversations, we create a list of 1,000 unique conversation premise seeds per domain in order to increase conversation diversity. A conversation template consisting of 7–10 turns with 0–3 <sil> placeholders per turn is created; all three streams are generated by an LLM according to this template, unlike in the DSTC8 case.

Validation. Despite significant effort tuning the prompts used to generate data, we found that current frontier models such as Claude Opus 4.6 (Anthropic, 2026a) still frequently make errors such as subtly referencing future thoughts array contents in earlier response phrases. To prevent errors like this from contaminating the training dataset, we iteratively re-generate each conversation until it passes a strict four-stage validator.

Candidate generations are first parsed to verify that they contain the appropriate number of turns and that matching Talker and Reasoner phrase pairs are present. Next, we detect ordering errors by computing pairwise BERTScore (Zhang et al., 2020) between each entry in the thoughts array and each entry in the response array for each turn to ensure each entry relates most highly to the entry within its temporally aligned pair. Third, we utilize DeBERTaV3 (He et al., 2023) fine-tuned on MNLI (Williams et al., 2018; Laurer, 2023) to verify each grounded phrase pair exhibits entailment (see Section B.3 for details). Finally, we verify that no proper nouns appear in any response field before first appearing in user or thoughts fields to prevent entity hallucinations.

Composition. The final ConvFill dataset consists of 8,443 conversations yielding 290,571 training examples: 111,552 silence-based filler examples and 179,019 knowledge-grounded examples in a JSONL format. The dataset is fully in English. The API cost of end-to-end dataset generation using Claude Opus 4.6 was approximately $2,400,

including retries due to validation failures. For prompt templates, validation details, and additional dataset statistics, see Appendix B.

5 Infill Model Training and Inference

5.1 Training Pipeline

We create a training pipeline that can be used to fine-tune models for the conversational infill `Talker` task. The pipeline is configured for each `Talker` model via control token sequences that delimit roles and turn boundaries. Most pretrained SLMs already define control sequences for the user role (`[USER]`), assistant role (`[ASST]`), and end-of-turn boundary (`[END]`). For the infill task, we add a new control sequence, `[KNOWLEDGE]`, and map it through each model's existing control-token template. Examples of expanded control sequences for each model and role can be found in Table 10. We add a new special token, `<sil>`, to indicate cases in which `Reasoner` knowledge is not yet available.

```
[USER] (history) user utterance τ − 1 [END]
[ASST] (history) assistant utterance τ − 1 [END]
[USER] user utterance τ [END]
[KNOWLEDGE] external knowledge chunk (e_i) [END]
[ASST] t_1, ..., t_{i−1}, t_i [END]
```

Figure 3: **Format for ConvFill training and inference.** During training, the model predicts the red text. The figure shows, in blue, a history lookback of $\ell = 1$, as defined in Section 3.1.

Training examples for `Talker` model fine-tuning are constructed as shown in Figure 3, following the task description in item 1. Training examples contain a history lookback, shown in blue in Figure 3, to ensure consistency across turns. Further history can be added in the same format if ℓ is increased. All training examples also contain the most recent user utterance, an external knowledge chunk, and, optionally, previous `Talker` phrases for the same turn, displayed in black in Figure 3.

5.2 Inference Pipeline

We design ConvFill to implement the `Talker-Reasoner` architecture defined in item 1. We use a series of queues to implement ConvFill as shown in Figure 2. The first queue buffers `Reasoner` knowledge chunks for ingestion by the `Talker`, and the second queue buffers `Talker` responses for ingestion by TTS. During ConvFill inference, a user utterance is transcribed and sent in parallel to both the `Talker` and `Reasoner`. The `Talker` performs greedy inference to process all available knowledge chunks and queue TTS responses. If the streamed knowledge queue is empty and the TTS queue is non-empty, the `Talker` waits until another knowledge chunk is available. However, if the TTS queue is empty, ConvFill dynamically produces a `<sil>` token for the `Talker` to ingest, causing it to generate a conversationally contingent infill phrase. In practice, `<sil>` tokens are inserted most frequently at the start of a conversation turn while waiting for the `Reasoner` to push a knowledge chunk into the streamed knowledge queue.

5.3 `Talker` and `Reasoner` Models

We fine-tune seven `Talker` models to perform the conversational infill task: Gemma 3 270M, Gemma 3 1B (Team et al., 2025), Qwen3 0.6B (Yang et al., 2025), SmolLM2 135M, SmolLM2 360M, SmolLM2 1.7B (Allal et al., 2025), and Llama 3.2 1B (Grattafiori et al., 2024). This covers a range of parameter counts from 135 M to 1.7 B across 4 model families. Multiple sizes of SmolLM and Gemma are included to investigate scaling within model families.

Per-model training configurations are available in Table 11 and Section C.1. All `Talker` models are trained with a history lookback of size one ($\ell = 1$; see Section 3.1 for the definition). In our evaluations, we use three `Reasoner` models across different families: Claude Opus 4.7 (Anthropic, 2026b), GPT-5.5 (OpenAI, 2026), and Gemini 3.1 Pro (Google DeepMind, 2026). The seven `Talker` and three `Reasoner` models yield 21 different ConvFill system configurations, which we evaluate across benchmarks and user studies. We convert all `Talker` models to the INT8 MLX format using MLX-LM (ML Explore, 2024) to enable benchmarking and evaluation directly on target hardware; `Talker` model inference for all experiments in Section 6 and Section 7 runs on an Apple MacBook Pro (M2 SoC, 16 GB of system memory).

6 Benchmark Evaluation

No single metric suffices to evaluate ConvFill: a system can answer accurately but express the answer unfaithfully, ground faithfully but omit critical information, or remain factually grounded while failing to address the user's intent. These failure modes are particularly difficult to disentangle in a dual-model system operating over an evolving dialogue, where errors can originate in the `Reasoner`,

Metric	Definition	Units	Method	Single-turn	Multi-turn	Live Interaction
Accuracy	Proportion of responses that correctly answer the question	%	LLM Judge (QA)	✓		
Conditional Accuracy	`Talker` accuracy excluding instances of incorrect `Reasoner` knowledge	%	LLM Judge (QA)	✓		
Entailment	Ratio of `Reasoner` thoughts within each turn that entail the corresponding **non-filler** `Talker` phrase	%	NLI Classifier	✓	✓	
Non-Contradiction	Ratio of **filler** `Talker` phrases within each turn that do not contradict `Reasoner` phrases	%	NLI Classifier	✓	✓	
Coverage	Degree to which `Reasoner` knowledge is reflected in the `Talker` response	Likert Scale (1–5)	LLM Judge	✓	✓	
Faithfulness	Degree of consistency between `Reasoner` and `Talker`	Likert Scale (1–5)	LLM Judge	✓	✓	
Helpfulness	Binary indicator of whether `Talker` addresses the user's utterance in an appropriate manner	True/False (0–1)	LLM Judge	✓	✓	
Time to First Response (TTFR)	Time from transcription end to TTS queueing	ms	Live Logging			✓
User Experience	Subjective ratings for Latency, Clarity, Fluency, Response Length, Coherence, Task Completion, Naturalness, Satisfaction	Likert Scale (1–5)	Survey (Likert)			✓
System Preference	Ordered ranking of systems according to preference	Rank (1–3)	Survey (Ranking)			✓

Table 1: **Evaluation metrics for ConvFill.** Check marks indicate which evaluation settings include each metric. Definitions are provided per-metric, describing what evaluation aspects each metric covers. Method indicates how the metric was measured. Metrics are measured across single-turn & multi-turn benchmarks and live interaction.

the `Talker`, or their interaction. We therefore construct a comprehensive evaluation spanning single-turn benchmarks, multi-turn benchmarks, and a live interaction study with human users, summarized in Table 1. This section focuses on benchmark evaluations; live interaction results are presented in Section 7.

6.1 Methods

For single-turn QA accuracy benchmarking, we compare four conditions: Base SLM (no conversational infill), ConvFill `Talker`, ConvFill `Reasoner`, and Frontier Model (no conversational infill). For NLI and LLM-as-a-Judge metrics in this section, we evaluate ConvFill `Talker` outputs only.

6.1.1 Benchmark Metrics

NLI Metrics. Following terminology presented in Williams et al., we define turn-level notions of *Entailment* and *Non-Contradiction* for the conversational infill task. Since `<sil>`-conditioned and `Reasoner`-knowledge-conditioned phrases serve distinct purposes, we evaluate them differently—*Entailment* applies *only* to `Talker` phrases in a turn that are generated from `Reasoner` knowledge. *Non-Contradiction* applies *only* to `Talker` phrases in a turn that are `<sil>`-conditioned.

Entailment measures the proportion of knowledge-conditioned `Talker` phrases that are entailed by their corresponding `Reasoner` phrase. This determines the `Talker`'s ability to ground its responses in `Reasoner`-provided knowledge. *Non-Contradiction* measures the proportion of `<sil>`-conditioned `Talker` phrases that do not contradict *any* of the `Reasoner` phrases in the turn. This measures the `Talker`'s ability to generate non-disruptive filler phrases for a turn. For both metrics, we normalize by number of relevant phrases per turn to standardize across turns with varied phrase count.

LLM-as-a-Judge Metrics. Individual phrases generated by the `Talker` can score highly on NLI metrics, but when combined together still omit critical information, fail to address user intent, or remain incoherent at the turn level. We therefore complement phrase-level NLI with LLM-as-a-Judge metrics (Zheng et al., 2023) that operate on the entire dialogue turn, motivated by prior work characterizing desirable model behavior in terms of helpfulness and faithfulness (Ouyang et al., 2022). The judge metrics are derived from full conversational context (user utterance, `Reasoner` thoughts, `Talker` response, and optionally history lookback). Treating the `Reasoner` output as ground truth, these metrics distinguish between complementary failure modes: whether the `Talker` adequately covers the `Reasoner`'s knowledge (Coverage, Faithfulness) and whether it appropriately addresses the user (Helpfulness).

Coverage measures how much of the available `Reasoner` knowledge is reflected in the overall response, capturing omissions or under-specification, analogous to content-selection and recall-oriented evaluation in summarization (Nenkova and Passonneau, 2004). *Faithfulness* measures whether the response introduces distortions, contradictions, or

unsupported claims relative to Reasoner knowledge, capturing hallucination and misrepresentation errors (Maynez et al., 2020; He et al., 2025). Together, these metrics form a precision-recall view of grounding quality, separating omission failures from unsupported addition or distortion.

Helpfulness captures the user-intent alignment component of instruction-following evaluation: whether the response meaningfully addresses the user's utterance, is coherent as a conversational turn, and is useful in context. Unlike broader notions of helpfulness that may implicitly include factual adequacy, our helpfulness metric isolates conversational usefulness from correctness with respect to the Reasoner knowledge, which is measured separately through Coverage and Faithfulness. Unlike Coverage and Faithfulness, which evaluate the Talker relative to the Reasoner, Helpfulness evaluates the Talker relative to the user. This isolates conversational capability from factual grounding. We express Helpfulness as a binary indicator, as human annotators in our pre-study pilot found a 1–5 scale difficult to apply consistently. Following prior work showing that thresholded Likert responses correlate more strongly with human judgments than direct binary questions (Movva et al., 2024; Wang et al., 2025b), we elicit a Likert score from the LLM-as-a-Judge evaluator and binarize at a threshold of ≥ 3.

LLM-as-a-Judge grading is performed by GPT-4o (OpenAI, 2024a) using the prompt in Section F.1. We observe sufficient inter-rater reliability between human graders, with a minimum Krippendorff's α of 0.67, and human–LLM agreement, with a minimum Kendall's τ_b of 0.55 and point-biserial r of 0.69. Per-metric statistical analysis and the human grading rubric are provided in Section D.6.

6.1.2 Evaluation Conditions

We run inference on text inputs only in this section. To simulate Reasoner latency, we prepend a random number of <sil> tokens (0–3) before knowledge chunks during evaluation. We verify that this number of <sil> tokens matches or exceeds the number generated in a real-world deployment of the ConvFill system (Table 16). All SLM inference (Base SLM and Talker) for this section uses the setup specified in Section 5.3.

6.1.3 Single-Turn Evaluations

We use two question-answering (QA) datasets for single-turn evaluation. SimpleQA (Wei et al., 2024) consists of human-written, short-response questions spanning a broad range of factual domains; because examples are adversarially filtered during construction to retain only questions answered incorrectly by GPT-3.5 (OpenAI, 2024b) or GPT-4 (OpenAI et al., 2024), it represents a set of intrinsically difficult factual QA examples. The LLAMA1 test set (Research, 2026), by contrast, consists of LLM-generated open-domain factual questions and is not filtered for failures of strong models, making it a comparatively easier factual QA benchmark.

For benchmarking with SimpleQA, we randomly sample 1,000 examples from the test split to form our evaluation set. Questions are prepared by following the SimpleQA dataset prompt and evaluation script (OpenAI, 2026) and passed into the Talker and Reasoner models, simulating ConvFill. Reasoner responses are prepended with <sil> tokens (see Section 6.1.2) and are used as the basis for Talker generation. All Talker-generated phrases together make up the final response.

Responses were graded for accuracy via the suggested GPT-4o judge model and evaluation prompt (OpenAI, 2026). Responses classified as *Not Attempted* and *Incorrect* are both considered to be incorrect for the purposes of tabulating accuracy metrics. Additional NLI and LLM-as-a-Judge metrics are calculated as indicated in Table 1. LLAMA1 (Research, 2026) is a similarly formatted dataset that covers simpler general knowledge and elementary science; models were evaluated on the full 300-question test set with the same prompting and judge model as SimpleQA.

Conditions. We evaluate the following four conditions: Base SLM (the non-fine-tuned base SLM), ConvFill Talker (our fine-tuned model, fed by knowledge from the Reasoner at inference time), ConvFill Reasoner (the knowledge fed to the Talker during inference, graded independently), and, as a baseline, the Frontier model (the standalone frontier model corresponding to the ConvFill Reasoner, prompted directly for QA independent of ConvFill).

6.1.4 Multi-Turn Evaluations

Multi-turn conversations pose an additional challenge because each response depends on the dialogue state established by previous turns; single-

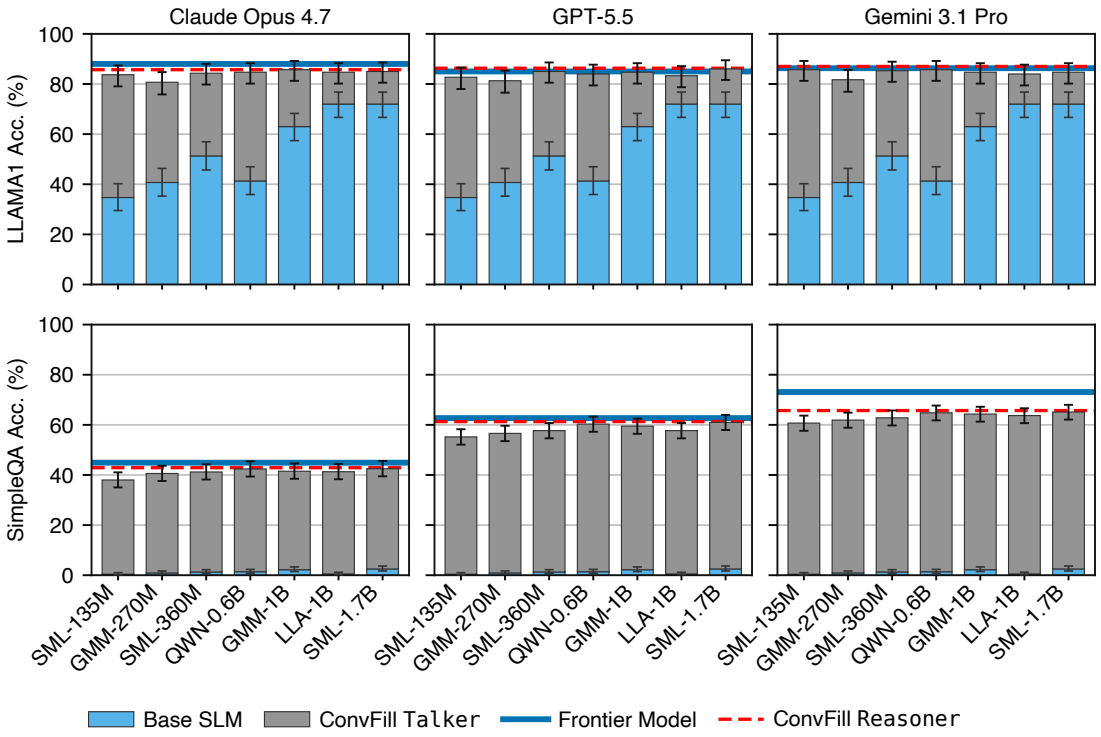

Figure 4: **QA accuracy for model configurations across LLAMA1 and SimpleQA.** Each panel compares seven `Talker` models paired with one `Reasoner`. Bars show Base SLM and ConvFill `Talker` accuracies; horizontal lines mark Frontier model and ConvFill `Reasoner` accuracies. Error bars show 95% confidence intervals. SML=SmolLM, GMM=Gemma 3, QWN = Qwen3, LLA=Llama. Full numerical results are in Appendix E.

turn QA evaluations alone cannot capture failures in context tracking, referent resolution, consistency, or error accumulation across a conversation. For multi-turn evaluation, we use two datasets that differ substantially in conversational complexity. MultiWOZ (Budzianowski et al., 2020) contains human–assistant task-oriented dialogues that average 13.5 turns and require tracking evolving user goals, constraints, and decisions across often multi-domain interactions. In contrast, Everyday Conversations (Face, 2024) consists of synthetic 3–4-turn conversations with simpler structure, making it a comparatively easier conversational continuity test.

Unlike in Section 6.1.3, where we simulate the entire ConvFill system during evaluation, here, we treat the original dataset *assistant* text at each turn as the `Reasoner` knowledge by splitting on sentence boundaries and prepending `<sil>` tokens. In order to account for history lookback, outputs are generated sequentially starting from the first turn of dialogue with each subsequent `Talker` response depending on the user field, simulated `Reasoner` knowledge, and the previous turn `Talker` response. We repeat this for all turns in the conversation.

For each ConvFill-simulated conversation constructed from the 1,000-conversation MultiWOZ test set, we select a random turn for grading, excluding the first and last turns (which are usually greetings or closings). The corresponding `Talker`-generated part of the turn is then graded on the metrics indicated in Table 1. This creates 991 graded turns, as conversations of two turns or less are implicitly excluded. We repeat this same process for the test set of Everyday Conversations, yielding 119 graded turns.

Conditions. We only evaluate the ConvFill `Talker` model output for this scenario, as all other multi-turn inputs directly use dataset contents or are generated by previous `Talker` inference turns.

6.2 Results

Figure 4 displays the accuracy of the ConvFill `Talker` model on the single-turn QA task relative to the corresponding Base SLM, `Reasoner`, and Frontier model. Since `Reasoner` knowledge chunks enable `Talker` performance gains, the accuracy of each `Reasoner` represents a ceiling for the corresponding `Talker` model performance. We see consistently across both SimpleQA and LLAMA1 benchmarks that ConvFill `Talker` models perform within 0%–6.3% of their corresponding `Reasoner` models. ConvFill `Talker` models, across `Reasoner` models, demonstrate gains as

	LLAMA1					SimpleQA				
Model	Ent.	Non-C.	Cov.	Faith.	Help.	Ent.	Non-C.	Cov.	Faith.	Help.
SmolLM2 135M	95.1	94.1	4.97	4.68	0.97	90.0	89.5	4.84	4.41	0.90
Gemma 3 270M	93.8	96.7	4.96	4.68	0.96	89.7	97.2	4.84	4.54	0.93
SmolLM2 360M	95.7	94.8	5.00	4.84	0.99	95.4	94.4	4.91	4.68	0.94
Qwen3 0.6B	95.4	90.2	5.00	4.87	0.99	96.6	93.7	4.92	4.83	0.97
Gemma 3 1B	96.3	86.8	5.00	4.86	0.99	95.6	92.2	4.91	4.72	0.96
Llama 3.2 1B	96.7	89.4	5.00	4.79	0.99	95.5	89.2	4.92	4.66	0.96
SmolLM2 1.7B	95.8	98.1	5.00	4.88	0.99	97.0	95.8	4.92	4.88	0.98
	Everyday Conversations					**MultiWOZ**				
Model	Ent.	Non-C.	Cov.	Faith.	Help.	Ent.	Non-C.	Cov.	Faith.	Help.
SmolLM2 135M	92.3	95.6	4.99	4.87	1.00	83.9	85.9	4.83	4.57	0.91
Gemma 3 270M	85.1	96.5	4.99	4.77	0.99	82.9	81.0	4.85	4.55	0.92
SmolLM2 360M	91.4	97.9	5.00	4.91	1.00	82.8	84.5	4.86	4.68	0.94
Qwen3 0.6B	92.0	96.7	5.00	4.88	1.00	83.0	81.8	4.90	4.74	0.94
Gemma 3 1B	84.8	98.2	5.00	4.91	1.00	82.3	81.2	4.85	4.52	0.92
Llama 3.2 1B	87.8	97.7	5.00	4.86	1.00	84.5	81.3	4.87	4.60	0.93
SmolLM2 1.7B	85.1	99.1	5.00	4.97	1.00	83.6	82.8	4.89	4.74	0.95

Table 2: **Compact NLI and GPT-4o judge metrics for QA datasets (top) and sampled multi-turn datasets (bottom).** QA values are averaged across the three hosted `Reasoner` models. Entailment (Ent.) and Non-Contradiction (Non-C.) are percentages; Coverage (Cov.), Faithfulness (Faith.), and Helpfulness (Help.) use the scales described in Table 1. Full results by `Reasoner` model and confidence intervals are in Appendix E.

high as 51% on LLAMA1 and 63.4% on SimpleQA, relative to the corresponding base SLMs.

For the LLAMA1 dataset, the standalone `Reasoner` accuracy relative to Frontier-model-only accuracy is 2.3% less for Claude, 0.7% greater for Gemini, and 1.3% greater for GPT-5.5. For the SimpleQA dataset, the standalone `Reasoner` accuracy relative to Frontier-model-only accuracy is 2.0% lower for Claude, 7.4% lower for Gemini, and 1.4% lower for GPT-5.5. Across these, we observe that the `Reasoner`-Frontier gap is roughly 2%, with the notable exception of Gemini 3.1 Pro, for which the gap is almost triple that of the other configurations on the SimpleQA task.

We display calculated NLI and LLM-as-a-Judge metrics in Table 2 for all four datasets. To determine the significance of trends with respect to `Talker` parameter size, we utilize the Benjamini–Yekutieli False Discovery Rate correction (Benjamini and Yekutieli, 2001) to control for false positives while accounting for potential dependencies across multiple hypothesis tests. Full significance test results are shown in Table 20.

On the two QA datasets, we observe a significant positive correlation between Base SLM parameter count and Accuracy, confirming that larger models have stronger underlying question-answering capabilities prior to any ConvFill system integration. This baseline result aligns with established scaling intuitions and provides a foundation for interpreting downstream ConvFill system trends. On the more challenging SimpleQA dataset, significant positive scaling trends emerge across nearly all metrics (Accuracy, Entailment, Coverage, Faithfulness, and Helpfulness), indicating that model size meaningfully improves response quality when the task places real demands on model capability. On the easier LLAMA1 dataset, Accuracy, Entailment, and Coverage are uniformly high across all model sizes, leaving little room for scaling trends to emerge. Positive trends are nonetheless still observed in Faithfulness and Helpfulness, suggesting these metrics remain sensitive to scale even when general correctness is largely saturated.

A similar pattern appears in the multi-turn evaluations. In MultiWOZ, we observe significant positive scaling trends for Coverage, Faithfulness, and Helpfulness, indicating that larger `Talker` models are better able to use `Reasoner` knowledge in longer, task-oriented dialogues. In contrast, no significant scaling trends emerge on Everyday Conversations. This divergence likely reflects the differing complexity of the two datasets and mirrors the single-turn contrast between SimpleQA and LLAMA1: scaling trends are most visible when the benchmark is challenging enough that model capability, rather than task saturation, limits performance.

Non-Contradiction is the main exception to the otherwise positive performance scaling with respect to `Talker` model size. It shows no clear relationship with `Talker` model size on the two

single-turn QA datasets or on Everyday Conversations; on MultiWOZ it shows a negative correlation with model size. We turn to within-family comparisons to examine this unexpected trend with additional nuance. Across the three SmolLM2-derived `Talker` models, scaling trends are broadly consistent with the overall parameter count results, with the notable exception of Non-Contradiction: a significant positive correlation between size and Non-Contradiction is observed on both QA datasets, a reversal of the cross-family trend. The Gemma 3 family shows the opposite pattern, with larger models exhibiting worse Non-Contradiction on QA datasets. We discuss these observations further in Section 8.

7 User Evaluation

Rather than evaluating perceived metrics from static transcripts or recordings, we build a full end-to-end, interactive ConvFill system prototype to enable users to interact with it directly. We conduct a user study (18 participants: 10 male, 8 female, ages 20–63). The study included a live interaction portion in which participants conversed with voice agents both with and without the `Talker-Reasoner` architecture, and a grading portion in which participants rated Coverage, Faithfulness, and Helpfulness of samples to generate a validation corpus for LLM-as-a-Judge evaluation.

7.1 Methods

We evaluate three conversational system configurations: Base SLM, Frontier, and ConvFill. In the Base SLM configuration, participants conversed directly with a small language model prompted for conversational interaction. In the Frontier configuration, participants conversed directly with a frontier model prompted for conversational interaction. In the ConvFill configuration, participants conversed with the full ConvFill system, in which a fine-tuned `Talker` model generated the user-facing response while receiving streamed knowledge from a `Reasoner` model. Llama 3.2 1B, Qwen3 0.6B, and Gemma 3 270M were used as Base SLMs, while Claude Opus 4.7 and GPT-5.5 were used as Frontier models. The ConvFill counterparts for these models were used as `Talker` and `Reasoner` models in this evaluation, as described in Section 5.3.

Study Tasks. In this evaluation, we augment standard chat to include two *out-of-distribution tasks* that do not appear in our fine-tuning data: retrieval-augmented generation (RAG) (Lewis et al., 2021), and tool use through Model Context Protocol (MCP) (Anthropic, 2024). We refer to the three tasks as Direct, RAG, and MCP. In the Direct task (standard chat), participants completed one of three task-oriented dialogue scenarios without external retrieval or tool access. In the RAG task, a database was connected to each Base SLM, `Reasoner`, or frontier model; participants completed scenarios requiring database information. In the MCP task, participants queried an email inbox via MCP.[2]

Interaction Conditions. Participants each completed all three task types—Direct, RAG, and MCP—following a Latin Square order (MacKenzie, 2024). Within each task, the order of system configurations (Base SLM, Frontier, or ConvFill) was randomized; the MCP configuration only featured Frontier and ConvFill configurations due to the Base SLM's inability to make MCP calls.

Participants were instructed to converse with each system for sufficient time to form an opinion, with a suggested minimum of 4 turns. After each interaction, participants rated each system on a 5-point Likert scale using the metrics in Table 3. After completing all interactions for a task, participants also ranked the systems in order of preference. Full details are provided in Section D.4.

Real-Time Interactive System. Each ConvFill live interaction is conducted with an end-to-end interactive prototype fully implementing the ConvFill system. We enable voice-based interaction using faster-whisper (SYSTRAN, 2023) for automatic transcription and the macOS Speech Synthesis Manager (Apple, 2026) for text-to-speech. The six Base SLM–Frontier LLM pairs were uniformly assigned across the 18 participants, ensuring each pair appeared three times. Pairs were held constant within each user session. We ran Base SLM and ConvFill `Talker` models as specified in Section 5.3; additional implementation details are provided in Section D.5 and the demo software in the project repository.

Metrics. We adopt eight metrics for evaluation of user perception. *Latency* refers to the user perception of whether the system responded fast enough to maintain natural conversation. Following Munro and Derwing's intelligibility and comprehensibility metrics, we define *Clarity* as the degree to which

[2]For MCP, the *Base SLM* configuration was omitted because the model could not make structured MCP calls.

Metric	Base SLM	Frontier	ConvFill
Latency	3.44 ± 1.25	3.46 ± 1.28	4.24 ± 0.85
Clarity	4.53 ± 0.84	4.81 ± 0.48	4.69 ± 0.64
Fluency	3.69 ± 1.45	4.83 ± 0.42	4.56 ± 0.72
Resp. Length	3.00 ± 1.43	4.07 ± 0.91	3.94 ± 1.05
Coherence	2.94 ± 1.16	4.72 ± 0.46	4.28 ± 1.32
Task Comp.	2.33 ± 1.49	4.35 ± 0.89	4.31 ± 0.84
Naturalness	3.61 ± 1.05	4.41 ± 0.74	3.83 ± 1.08
Satisfaction	2.03 ± 1.03	3.80 ± 1.05	3.81 ± 1.03

Table 3: **Likert ratings per system aggregated across tasks.** Mean $\pm$ std. dev. are reported. Base SLM has a sample size of $N = 36$, and Frontier and ConvFill have a sample size of $N = 54$. Metrics are described in Section 7.1.

Task	System	Rank 1	Rank 2	Rank 3
Direct	Base SLM	0	3	15
	Frontier	10	7	1
	ConvFill	8	8	2
RAG	Base SLM	1	2	15
	Frontier	5	11	2
	ConvFill	12	5	1
MCP	Frontier	7	11	–
	ConvFill	11	7	–

Table 4: **Rank distribution by system and task.** Participant counts; $N = 18$ per task. Rank 1 = best. Overall, participants rate ConvFill better than or generally on par with the Frontier model configuration.

the user could understand the words the system said and *Fluency* as how well the users could understand ideas expressed by the system. *Response Length* refers to how appropriate the user felt the length of the response was. *Coherence* measures whether sentences relate to each other and progress logically (Barzilay and Lapata, 2005). *Task Completion* refers to whether system responses provided information needed to complete the given task (Mohammadi et al., 2025). *Naturalness* refers to whether the language used felt appropriate for the dialogue context (Danieli and Gerbino, 1996). *Satisfaction* refers to how satisfied users were with their conversation with the system. We give each of the aforementioned metrics names so that we can refer to them; during the study, participants were only given *descriptions* of the metrics to refer to (see Section D.2).

Infill Grading. Participants completed an infill grading task in which they were given a user utterance, a set of external knowledge chunks, and a set of conversational responses. They were asked to grade Coverage and Faithfulness on 1–5 scales and Helpfulness as a binary judgment. Grading results were collected and used for correlation with LLM-as-a-Judge ratings mentioned in Section 6. For details, see Section D.6.

7.2 Results

Across all task and system conditions, the study collected 144 interactions comprising 635 total turns.

User Ratings. Summarized Likert rating results are in Table 3. We compare Frontier and ConvFill ratings with two one-sided tests (TOST) and a non-parametric Wilcoxon rank-sum test (see Section D.7). ConvFill was rated equivalently to the Frontier system on Clarity, Fluency, Response Length, Coherence, Task Completion, and Satisfaction ($p < 0.01$ for all, see Section D.7). ConvFill exceeded the Frontier configuration on Latency, but fell below the Frontier configuration on Naturalness: some users who were not used to models using filler phrases scored this metric lower. The Base SLM condition was rated significantly below ConvFill and Frontier across all metrics.

System Rankings. Rankings were pooled across all tasks, yielding $N = 54$ observations per system, except for the Base SLM, which only has $N = 36$ observations due to inability to make MCP calls. Users ranked the models for each task (1: best, 3: worst for Direct and RAG; 1: best, 2: worst for MCP) as shown in Table 4. Rank-1 tests show no preference for ConvFill vs. Frontier on Direct and MCP tasks, but ConvFill was strongly preferred for RAG tasks (Section D.8). Base SLM was ranked last by 15/18 users on Direct and RAG tasks and was excluded in the MCP task.

Latency & Filler Analysis. We analyze the time-to-first-response (TTFR) of all four configurations across all tasks in Figure 5. We define *TTFR* as the time from receipt of the user's transcribed message to the first generated full sentence, rather than the first token. We choose this granularity because it reflects when a complete sentence is ready to be voiced by the TTS system. We observe that all three tasks (Direct, RAG, and MCP) exhibit distinct, non-overlapping TTFR distributions, with ConvFill `Reasoner` and Frontier configurations exhibiting similar TTFRs across tasks. ConvFill `Reasoner` TTFR varied substantially, from a mean of 2,947 ms on Direct tasks to 7,242 ms on MCP tasks, due to latency of context retrieval and tool calls. Despite this, the ConvFill `Talker` filler responses remained consistently fast across tasks (Direct: 542 ms, RAG: 976 ms, and MCP: 478 ms), with per-sample speedups averaging 7.4×, 9.2×, and 19.1× relative to the `Reasoner` response as shown in Table 14. ConvFill emitted more filler

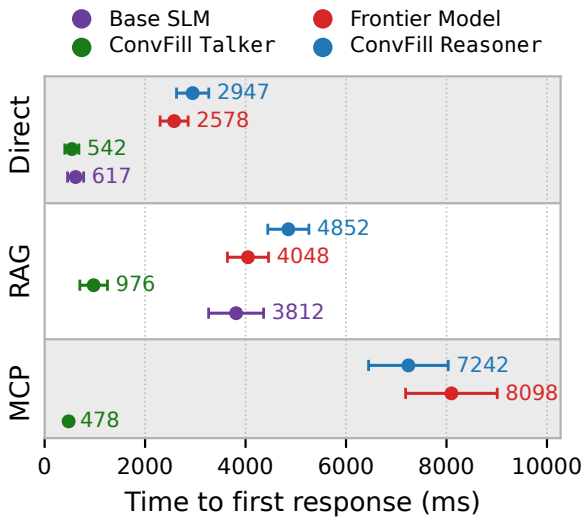

Figure 5: **TTFR Analysis per Task.** Horizontal shaded regions indicate each task, while labeled points with 95% CI error bars indicate latency. ConvFill `Talker` TTFR matches or improves on Base SLM TTFR despite ConvFill `Reasoner` and Frontier models demonstrating similar TTFR across tasks. Full statistics are in Table 15.

tokens as `Reasoner` TTFR increased, compensating for varied `Reasoner` latencies across tasks (Direct: $\mu = 2{,}947$ ms, RAG: $\mu = 4{,}852$ ms, MCP: $\mu = 7{,}242$ ms). ConvFill `Talker` TTFR was similar to the Base SLM for Direct tasks (542 ms vs. 617 ms), but much lower for RAG tasks (976 ms vs. 3,812 ms), as the Base SLM incurs the burden of processing retrieved context while ConvFill offloads this to the `Reasoner`.

8 Discussion

Dynamic Fillers for Perceived Responsiveness. ConvFill substantially reduces both actual and perceived latency relative to frontier models, while retaining their accuracy on QA tasks. ConvFill consistently delivers millisecond-level-TTFR responses across three different task types with `Reasoner` latencies. This benefit is especially pronounced for tasks involving external resources, due to the inevitable access time to query external databases or email servers via third-party providers; particularly, these latencies are not reduced via model-level optimization. ConvFill dynamically adapts to variable `Reasoner` latency by inserting additional `<sil>` tokens, emitting more `Talker` filler sequences as task complexity grows. This adaptive filling allows a single `Talker` to handle a range of latency conditions and variable infill lengths without task-specific TTFR tuning. User study participants' perceived latency ratings reflect these measurements: ConvFill received the strongest latency scores of any configuration, demonstrating alignment between TTFR measurements and subjective experience.

Reducing Local Inference-Time Burden. ConvFill's measured `Talker` TTFR is equal to or lower than that of the corresponding base SLMs across all tasks. Additionally, assisted by the `Reasoner` model, the `Talker` recovers almost all of the frontier-model-only performance while maintaining this TTFR. Simply using a small on-device model does not solve the TTFR problem—particularly in the RAG task, the base SLM infers almost as slowly as the frontier model despite being identical in size and architecture to the `Talker`. This is because the unassisted base SLM must independently process the full retrieved context, while the `Talker` gets phrase-based, condensed knowledge from the `Reasoner` model. Due to this modularity, the ConvFill architecture also reduces the inference-time token-processing burden on the `Talker` model, enabling it to remain responsive in conversation.

Accuracy and Frontier Model Parity. Accuracy results demonstrate that a `Talker` can faithfully incorporate `Reasoner` knowledge: the end-to-end ConvFill system approaches frontier model-only accuracy on QA tasks. There are two points where ConvFill performance could suffer relative to the unmodified Frontier Model baseline: prompting changes that induce the phrase-based `Reasoner` output format could hurt task performance; and `Reasoner` knowledge could be lost or corrupted as the `Talker` integrates it into conversation.

As illustrated in Section 6, `Talker` accuracy on QA tasks generally approaches the performance ceiling set by the corresponding Frontier Model baseline. The one exception to this occurs with the Gemini 3.1 Pro `Reasoner` on SimpleQA, where `Talker` performance trails baseline performance by an average of 9.8%. Examining the ConvFill `Reasoner` results more closely indicates that this anomalous behavior results from degradation in frontier model performance when switched to the `Reasoner` output format. Thus, the discrepancy is better characterized as model-specific sensitivity of Gemini 3.1 Pro to the ConvFill `Reasoner` prompting context, rather than as a failure mode introduced by the `Talker` component itself.

This observation should not be discarded as an outlier; it is consistent with prior work showing that seemingly minor prompt choices such as format-

ting or example ordering can produce substantial, model-dependent performance differences (Zhuo et al., 2024; Lu et al., 2022; Sclar et al., 2024). Our evaluation uses a single generic prompt structure across `Reasoner` models rather than tuning prompts separately for each model, providing a conservative estimate under a uniform deployment setting. In application-specific deployments, further task- and model-specific prompt adaptation could verify `Reasoner` robustness and provide an additional path to improve ConvFill performance.

Task Difficulty and Metric Saturation. Benchmark results reveal a more nuanced picture of what the conversational infill task necessitates beyond QA accuracy alone. Grounding metrics such as Entailment and Coverage are consistently strong across model sizes and datasets, suggesting that `Talker` models reliably learn to incorporate and reflect `Reasoner` knowledge in their responses. Faithfulness and Helpfulness, which capture subtle aspects of response quality, namely whether the `Talker` distorts meaning or fails to address the user's intent, show greater variance across models and are more sensitive to task difficulty, with the MultiWOZ dataset yielding notably lower scores than the QA and Everyday Conversations datasets. This suggests that while the mechanical task of knowledge incorporation is learnable across all of the model sizes we evaluate, producing responses that are accurate in representation and appropriate in conversational context remains a more difficult and discriminating challenge. On sufficiently easy tasks, metric saturation obscures scaling behavior entirely, suggesting that benchmarks for ConvFill-type systems should be calibrated to a difficulty level where model capability is the limiting factor rather than the task ceiling.

`Talker` Model Scaling. Accuracy, Coverage, Faithfulness, and Helpfulness all scale positively with model size, particularly on more challenging samples where performance is not saturated. Entailment scales on single-turn QA benchmarks but not on multi-turn benchmarks. We attribute this to the NLI checker being unable to robustly handle multi-turn nuances, further motivating our choice to evaluate with additional LLM-as-a-Judge metrics. The Non-Contradiction metric demonstrates opposing trends between SmolLM and Gemma model families Table 20. This suggests that other factors, such as model architecture, training methods, and dataset composition, play a larger part in performance on this metric.

Live Interaction Takeaways. Conducting a live end-to-end evaluation enables users to assess the full ConvFill system directly under scenarios representative of real-world use. The live setting also adds two forms of realism beyond the benchmark conditions. First, it introduces two task types absent from training: retrieval-augmented generation and tool use through MCP, leaving the Direct task as the only condition close to the training distribution. Second, it runs the full system on real users' unscripted interactions, with participants given only high-level guidance on potential topics and queries (Section 7). Under live conditions, results follow the general trend of the earlier benchmarks, with ConvFill and Frontier configurations significantly exceeding the Base SLM across metrics. ConvFill is statistically equivalent to the Frontier configuration on all metrics except for Latency, where it exceeds the Frontier configuration and Naturalness, where it trails (Section D.7).

In aggregate, users rank ConvFill on par with the Frontier configuration for the Direct and MCP tasks and prefer it outright for RAG as shown in Table 4, suggesting that better responsiveness is preferable, even at the cost of some naturalness. The RAG and MCP tasks are also the two cases where the `Reasoner` must issue a query and wait to receive and encode an external result before responding, incurring the highest latency in our system. The clearest user preference appears on RAG rather than on MCP, even though MCP carries the longest `Reasoner` delays. These longer delays produce more infill phrases (2.35 vs 1.34 per turn on average) before the `Reasoner` response arrives (Table 16). A possible explanation is that ConvFill introduces the most benefit during medium-length inference delays, and that at higher infill counts it may begin to detract from the experience relative to silence.

Latency-Adaptive Behavior. The naturalness-responsiveness tradeoff suggests that simply filling all moments of silence with infill may not be optimal. Instead, a future ConvFill system could anticipate how long the `Reasoner` is likely to take and adapt accordingly. When a fast response is expected, ConvFill could hold off on non-grounded infill and wait briefly to answer from `Reasoner` knowledge. For longer delays, ConvFill could dynamically generate pauses when indicated by `Talker` filler content (e.g. *"Let me take a moment to check your inbox..."*), rather than sustaining constant filler until the response arrives. Tuning

infill for context and expected latency in this way, rather than applying it uniformly, offers a path to preserving the responsiveness users value while recovering the naturalness that filler can cost when in excess.

Development Feasibility. We demonstrate that the conversational infill task is learnable across a diverse set of small model architectures and parameter scales, from 135M to 1.7B parameters, spanning widely used Gemma, Qwen, SmolLM, and Llama families. Resource requirements for fine-tuning range from 2.8–48.9 GPU hours on an RTX 6000, with a total estimated cost of $133.70 to train all 7 `Talker` models. Full, per-model breakdowns of cost and GPU hours are found in Table 11. Overall, adopting ConvFill for new application scenarios or `Talker-Reasoner` pairings does not require large-scale training infrastructure and can be done cheaply on previous-generation hardware.

We generated and verified our synthetic dataset at a one-time cost of $2,400. This cost is due to both the size of the dataset and the sample-generation retries needed to meet strict validation requirements.

Target Devices. Our evaluation is representative of real deployment in both target hardware and model precision. We benchmark and run our live interaction study on a laptop with an Apple M2 SoC and 16 GB of memory, using the same INT8-quantized `Talker` models suitable for deployment, with graph sizes ranging from 136 MB to 1.69 GB (Table 8). Measured latency, reported accuracy, and grounding metrics therefore reflect realistic deployment conditions (Laskaridis et al., 2024) and illustrate the feasibility of deployment across a wide range of existing similarly equipped consumer and edge devices. Smartphones are an especially natural target: the `Talker` weights occupy only a fraction of the RAM in recent phones, whose SoCs include dedicated neural hardware with native support for INT8 inference (Tan et al., 2024). This makes conversational infill a practical approach to bridge the latency-capability tradeoff between frontier-level intelligence and real-time responsiveness for applications like conversational digital assistants.

Decoupling Model Behavior and Capability. The broader implication of the ConvFill architecture is a clean separation between the model responsible for conversational behavior (`Talker`) and the model responsible for advanced capabilities and reasoning (`Reasoner`). Decoupling the tasks in this way presents several advantages relative to prior conversational AI approaches.

First, it makes the `Talker` independently trainable. This enables optimizing for natural, task-appropriate spoken interaction without incurring the high cost of directly fine-tuning frontier text or audio models (Défossez et al., 2024; Xia et al., 2024; Jin et al., 2023). This also enables task-specific adaptation while avoiding the possibility of fine-tuning-induced performance degradation (Luo et al., 2025). Beyond this, we envision use cases in which the `Talker` can be adapted to enable stylistic variations or configurable conversational behaviors.

Second, this makes the ConvFill system modular—we show that `Reasoner` models can be freely swapped without retraining or altering the prompting of `Talker` models. This facilitates leveraging the variety of pre-existing tools and data sources already integrated with frontier-class models as drop-in `Reasoner` replacements without re-engineering or tuning them for real-time latency and conversational behavior.

Interaction Consistency Across Upgrades. A longstanding challenge in deploying conversational systems or chatbots is keeping conversational behavior consistent across model iterations and upgrades, as users recognize these changes and often respond negatively (Freitas et al., 2025). Maintaining a consistent customized `Talker` model across `Reasoner` updates could enable model service providers to more easily maintain a consistent user experience. This possibility is especially attractive given the rapidly changing landscape of frontier model performance and token costs (Chen et al., 2023).

9 Related Work

Recent work has divided speech and reasoning across models to balance responsiveness and intelligence. Mini-Omni-Reasoner (Xie et al., 2025) applies token-level thinking-in-speaking. SplitReason (Akhauri et al., 2025) offloads reasoning to backend LLMs. Qwen2.5-Omni (Xu et al., 2025) presents a Thinker-Talker architecture in which the Talker takes high-level representations from the Thinker and generates speech tokens. Speculative and online speculative decoding (Leviathan et al., 2023; Liu et al., 2023) allow faster generation that is later verified or refined by a larger model, improving effective decoding speed. Li and Grover (2025) extend this approach to cascaded speech models. Chen et al. facilitate retrieval from a text-

audio hybrid knowledge base. Arora et al. predict tool calls in parallel with user utterances. Kuroki et al. use an oracle-based LLM system to refine potential responses during user utterances. Chien et al. integrate different RAG backends with a full-duplex voice model frontend, handling retrieval delays of up to two seconds.

10 Conclusion

We introduce conversational infill, a novel task leveraging a `Talker-Reasoner` approach. This enables lightweight and responsive SLM `Talker` models to respond promptly with context-aware fillers while also incorporating external knowledge from frontier-class `Reasoner` models into their responses at inference time. Consistently producing millisecond-level time-to-first-response across tasks with variable `Reasoner` latency and across `Reasoner` models, ConvFill maintains grounding in external knowledge while preserving task accuracy. ConvFill presents a promising approach to creating modular, responsive, intelligent voice agents for task-oriented dialogue.

Together, these properties place ConvFill at a new point on the latency–capability Pareto frontier (Lin et al., 2025) by coupling the responsiveness of lightweight SLMs with the raw task performance of foundation models. We believe the architectural flexibility enabled by conversational infill presents a clear path toward developing conversational AI systems that excel at task-oriented and tool-use scenarios that are underserved by current state-of-the-art voice models. To encourage the broader community across AI and interactions research to experiment with and further validate the concepts presented in this paper, we release our model weights, dataset, and code for training and inference.

Limitations

While most safety concerns in ConvFill's two-stage architecture are mitigated by a guardrailed `Reasoner`, the `Talker` model operates independently and may not immediately comply with safety constraints, as fillers are generated before `Reasoner` responses are available. Future work should explore tighter safety coupling between the two components. Our evaluation did not capture cases of `Reasoner` timeouts or excessive latency. In practice, prolonged filler sequences could become disruptive, and future work should investigate appropriate inter-filler pauses and fallback strategies for such scenarios. Additionally, a larger user evaluation sample would improve the statistical robustness and generalizability of our findings. Finally, future work could evaluate the applicability of ConvFill to languages other than English.

Ethical Considerations

Our user study involved human participants who were informed of the study's purpose and compensated for their participation. The study did not involve vulnerable populations and posed no foreseeable risk to participants. Our study was approved by the Institutional Review Board at our institution (Study #STUDY00023284). Our synthetically generated dataset does not contain any personally identifiable information and was vetted for inappropriate content.

Acknowledgements

We thank Maximus Powers for help with initial user interface prototyping and Alisa Liu for valuable feedback on early drafts of this paper.

This research was supported in part by NSF awards 2338736 and 2310515.

Vikram Iyer holds concurrent appointments as an Assistant Professor in the Paul G. Allen School of Computer Science and Engineering at the University of Washington and as an Amazon Visiting Academic. This paper describes work performed at the University of Washington and is not associated with Amazon.

References

Yash Akhauri, Anthony Fei, Chi-Chih Chang, Ahmed F AbouElhamayed, Yueying Li, and Mohamed S Abdelfattah. 2025. Splitreason: Learning to offload reasoning. *arXiv preprint arXiv:2504.16379*.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, and 3 others. 2025. Smollm2: When smol goes big – data-centric training of a small language model. *Preprint*, arXiv:2502.02737.

Anthropic. 2024. Model context protocol specification. Version 2024-11-05.

Anthropic. 2026a. Claude opus 4.6 system card. Technical report, Anthropic.

Anthropic. 2026b. Claude opus 4.7 system card. Technical report, Anthropic.

Apple. 2026. Speech Synthesis Manager.

Siddhant Arora, Haidar Khan, Kai Sun, Xin Luna Dong, Sajal Choudhary, Seungwhan Moon, Xinyuan Zhang, Adithya Sagar, Surya Teja Appini, Kaushik Patnaik, Sanat Sharma, Shinji Watanabe, Anuj Kumar, Ahmed Aly, Yue Liu, Florian Metze, and Zhaojiang Lin. 2025. Stream rag: Instant and accurate spoken dialogue systems with streaming tool usage. *Preprint*, arXiv:2510.02044.

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 141–148, Ann Arbor, Michigan. Association for Computational Linguistics.

Yoav Benjamini and Daniel Yekutieli. 2001. The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4).

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2020. Multiwoz – a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *Preprint*, arXiv:1810.00278.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *arXiv preprint*. ArXiv:2305.05176 [cs.LG].

Yifu Chen, Shengpeng Ji, Haoxiao Wang, Ziqing Wang, Siyu Chen, Jinzheng He, Jin Xu, and Zhou Zhao. 2025. Wavrag: Audio-integrated retrieval augmented generation for spoken dialogue models. *Preprint*, arXiv:2502.14727.

Chung-Ming Chien, Manu Orsini, Eugene Kharitonov, Neil Zeghidour, Karen Livescu, and Alexandre Défossez. 2026. Moshirag: Asynchronous knowledge retrieval for full-duplex speech language models. *Preprint*, arXiv:2604.12928.

Konstantina Christakopoulou, Shibl Mourad, and Maja Matarić. 2024. Agents thinking fast and slow: A talker-reasoner architecture. *arXiv preprint arXiv:2410.08328*.

Santiago Cuervo and Ricard Marxer. 2024. Scaling properties of speech language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, page 351–361. Association for Computational Linguistics.

Morena Danieli and Elisabetta Gerbino. 1996. Metrics for evaluating dialogue strategies in a spoken language system. *Preprint*, arXiv:cmp-lg/9612003.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. 2024. Moshika MLX Q8: Moshi speech-text foundation model (mlx, 8-bit quantized). `https://huggingface.co/kyutai/moshika-mlx-q8`. Model card. License: CC-BY 4.0.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. 2024. Moshi: a speech-text foundation model for real-time dialogue. *Preprint*, arXiv:2410.00037.

Hugging Face. 2024. Everyday conversations for llms. `https://huggingface.co/datasets/HuggingFaceTB/everyday-conversations-llama3.1-2k`.

Julian De Freitas, Noah Castelo, Ahmet K. Uğuralp, and Zeliha Oğuz-Uğuralp. 2025. Lessons From an App Update at Replika AI: Identity Discontinuity in Human-AI Relationships. *arXiv preprint*. ArXiv:2412.14190 [cs.HC].

Heting Gao, Hang Shao, Xiong Wang, Chaofan Qiu, Yunhang Shen, Siqi Cai, Yuchen Shi, Zihan Xu, Zuwei Long, Yike Zhang, Shaoqi Dong, Chaoyou Fu, Ke Li, Long Ma, and Xing Sun. 2025. Lucy: Linguistic understanding and control yielding early stage of her. *Preprint*, arXiv:2501.16327.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. 2025. Scaling synthetic data creation with 1,000,000,000 personas. *Preprint*, arXiv:2406.20094.

Google Cloud. 2026. Accelerator-optimized VM pricing. `https://cloud.google.com/products/compute/pricing/accelerator-optimized#nvidia-rtx-virtual-workstations-`. Accessed: June 10, 2026.

Google DeepMind. 2026. Gemini 3.1 pro model card. Technical report, Google DeepMind.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jacqueline He, Howard Yen, Margaret Li, Stella Li, Zhiyuan Zeng, Weijia Shi, Yulia Tsvetkov, Danqi Chen, Pang Wei Koh, and Luke Zettlemoyer. 2025. Precise information control in long-form text generation. In *Advances in Neural Information Processing Systems*, volume 38, pages 92727–92792. Curran Associates, Inc.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. *arXiv preprint*. ArXiv:2111.09543.

Brendan Iribe, Ankit Kumar, and The Sesame Team. 2024. Crossing the uncanny valley of voice. Blog post.

Derek Jacoby, Tianyi Zhang, Aanchan Mohan, and Yvonne Coady. 2024. Human latency conversational turns for spoken avatar systems. *Preprint*, arXiv:2404.16053.

Hongpeng Jin, Wenqi Wei, Xuyu Wang, Wenbin Zhang, and Yanzhao Wu. 2023. Rethinking learning rate tuning in the era of large language models. *Preprint*, arXiv:2309.08859.

So Kuroki, Yotaro Kubo, Takuya Akiba, and Yujin Tang. 2026. Kame: Tandem architecture for enhancing knowledge in real-time speech-to-speech conversational ai. *Preprint*, arXiv:2510.02327.

Stefanos Laskaridis, Kleomenis Katevas, Lorenzo Minto, and Hamed Haddadi. 2024. MELTing point: Mobile Evaluation of Language Transformers. *arXiv preprint*. ArXiv:2403.12844 [cs.LG].

Moritz Laurer. 2023. MoritzLaurer/DeBERTa-v3-base-mnli · Hugging Face.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR.

Stephen C. Levinson and Francisco Torreira. 2015. Timing in turn-taking and its implications for processing models of language. *Frontiers in Psychology*, Volume 6 - 2015.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv preprint*. ArXiv:2005.11401 [cs.CL].

Shasha Li and Chien-Hsiung Chen. 2019. The effects of visual feedback designs on long wait time of mobile application user interface. *Interacting with Computers*, 31(1):1–12.

Shufan Li and Aditya Grover. 2025. Predgen: Accelerated inference of large language models through input-time speculation for real-time speech interaction. *arXiv preprint arXiv:2506.15556*.

Guan-Ting Lin, Chen Chen, Zhehuai Chen, and Hung yi Lee. 2026. Full-duplex-bench-v3: Benchmarking tool use for full-duplex voice agents under real-world disfluency. *Preprint*, arXiv:2604.04847.

Yueqian Lin, Zhengmian Hu, Qinsi Wang, Yudong Liu, Hengfan Zhang, Jayakumar Subramanian, Nikos Vlassis, Hai Helen Li, and Yiran Chen. 2025. Voice evaluation of reasoning ability: Diagnosing the modality-induced performance gap. *Preprint*, arXiv:2509.26542.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. 2023. Online speculative decoding. *arXiv preprint arXiv:2310.07177*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2025. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *Preprint*, arXiv:2308.08747.

I Scott MacKenzie. 2024. *Human-Computer Interaction: An Empirical Research Perspective*, 2 edition. Elsevier.

Amama Mahmood, Junxiang Wang, Bingsheng Yao, Dakuo Wang, and Chien-Ming Huang. 2025. User interaction patterns and breakdowns in conversing with llm-powered voice assistants. *International Journal of Human-Computer Studies*, 195:103406.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On Faithfulness and Factuality in Abstractive Summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

ML Explore. 2024. MLX LM: Run LLMs with MLX. `https://github.com/ml-explore/mlx-lm`.

Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. 2025. Evaluation and benchmarking of llm agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, page 6129–6139. ACM.

Rajiv Movva, Pang Wei Koh, and Emma Pierson. 2024. Annotation alignment: Comparing LLM and human annotations of conversational safety. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9048–9062, Miami, Florida, USA. Association for Computational Linguistics.

Murray J. Munro and Tracey M. Derwing. 1995. Foreign accent, comprehensibility, and intelligibility in the speech of second language learners. *Language Learning*, 45:73–97.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating Content Selection in Summarization: The Pyramid Method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 145–152, Boston, Massachusetts, USA. Association for Computational Linguistics.

OpenAI. 2024a. GPT-4o system card. Technical report, OpenAI.

OpenAI. 2024b. Introducing ChatGPT.

OpenAI. 2026. GPT-5.5 system card. Technical report, OpenAI.

OpenAI. 2026. openai/simple-evals. Original-date: 2024-04-11T22:38:17Z.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. GPT-4 Technical Report. *arXiv preprint*. ArXiv:2303.08774 [cs.CL].

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *arXiv preprint*. ArXiv:2203.02155 [cs].

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8689–8696.

Google Research. 2026. google-research-datasets/LLAMA1-Test-Set. Original-date: 2024-03-13T18:05:24Z.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting. *arXiv preprint*. ArXiv:2310.11324 [cs.CL].

Silero Team. 2024. Silero VAD: Pre-trained enterprise-grade voice activity detector (VAD), number detector and language classifier. `https://github.com/snakers4/silero-vad`.

Gabriel Skantze. 2021. Turn-taking in conversational systems and human-robot interaction: A review. *Computer Speech & Language*, 67:101178.

SYSTRAN. 2023. faster-whisper: Faster Whisper transcription with CTranslate2. `https://github.com/SYSTRAN/faster-whisper`.

Fuwen Tan, Royson Lee, Łukasz Dudziak, Shell Xu Hu, Sourav Bhattacharya, Timothy Hospedales, Georgios Tzimiropoulos, and Brais Martinez. 2024. MobileQuant: Mobile-friendly Quantization for On-device Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9761–9771, Miami, Florida, USA. Association for Computational Linguistics.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Bin Wang, Xunlong Zou, Geyu Lin, Shuo Sun, Zhuohan Liu, Wenyu Zhang, Zhengyuan Liu, AiTi Aw, and Nancy F. Chen. 2025a. AudioBench: A universal benchmark for audio large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4297–4316, Albuquerque, New Mexico. Association for Computational Linguistics.

Victor Wang, Michael JQ Zhang, and Eunsol Choi. 2025b. Improving LLM-as-a-Judge Inference with the Judgment Distribution. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23173–23199, Suzhou, China. Association for Computational Linguistics.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *Preprint*, arXiv:2411.04368.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Yuchen Xia, Jiho Kim, Yuhan Chen, Haojie Ye, Souvik Kundu, Cong Hao, and Nishil Talati. 2024. Understanding the performance and estimating the cost of llm fine-tuning. *Preprint*, arXiv:2408.04693.

Zhifei Xie, Ziyang Ma, Zihang Liu, Kaiyu Pang, Hongyu Li, Jialin Zhang, Yue Liao, Deheng Ye, Chunyan Miao, and Shuicheng Yan. 2025. Mini-omni-reasoner: Token-level thinking-in-speaking in large speech models. *arXiv preprint arXiv:2508.15827*.

Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, and 1 others. 2025. Qwen2.5-omni technical report. *arXiv preprint arXiv:2503.20215*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. ProSA: Assessing and Understanding the Prompt Sensitivity of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1950–1976, Miami, Florida, USA. Association for Computational Linguistics.

A Appendix

This appendix contains additional details needed to reproduce the results of our experiments, as well as longer illustrative examples to support the main text. The appendix is laid out as follows:

- Appendix B: Dataset Generation Details
- Appendix C: Model Training Details
- Appendix D: User Study Details
- Appendix E: Full Benchmark Evaluation Results
- Appendix F: Inference and Evaluation Prompts

A.1 AI Use Statement

Large language model (LLM) assistants were used in the preparation of this work for programming assistance (including figure plotting and table generation), document formatting, and proofreading. All scientific content, experimental design, and conclusions are the work of the authors.

B Dataset Generation

The released dataset comes from two complementary generation tracks that share a single scaffold-fill-validate-retry pipeline: a *topic-seeded freeform* track that synthesizes full conversations from short topic prompts in six domains, and a *DSTC8 schema-guided scaffolded* track that re-skins existing task-oriented dialogues from DSTC8 (Rastogi et al., 2020) into the conversational infill format while pinning the original user utterances and system content as the thought stream. Both tracks emit JSON conversations with the parallel `thoughts`/`response` arrays shown in Fig. 3 and pass through the same structural, NLI, and BERTScore validators, with the scaffolded track adding two checks that enforce preservation of the source content and the inference-time visibility constraint. Validation thresholds throughout the pipeline were selected by inspection. We describe each track and the validators in turn; code, prompt templates, topic lists, and the full configuration files will be released with the dataset.

B.1 Topic-Seeded Freeform Generation

Domain coverage and topic seeding. The freeform track covers six conversational domains: *advice*, *assistant queries*, *event planning*, *customer service*, *education*, and *medical conversation*. Each domain is keyed by its own seed file of around 1,000 lines, with one conversation generated per line. Two seeding strategies are used. For advice, assistant queries, and event planning, each line is a free-text description of the conversational topic (for example, "`How can I get better at setting and following through on goals?`"), paired with a fixed user persona such as `a helpful assistant`. For customer service, education, and medical conversations, each line is a ⟨*persona, scenario*⟩ pair joined with a comma ("`Radiologic Technologist, The technologist instructs the patient to hold their breath and remain still for the chest X-ray.`"); the persona supplies the P2 role and the scenario seeds the topic. We generated personas to be domain-appropriate but not over-specified; highly specific personas borrowed verbatim from PersonaHub (Ge et al., 2025) tended to derail conversations toward unrelated biographical detail. Topic and persona lists were generated up front with the same LLM used for conversation generation and then frozen.

Per-conversation scaffold construction. For each topic, the generator samples the global shape of the conversation, then renders a JSON skeleton with typed placeholders that the LLM fills in. For each topic, we draw a turn count $T \sim \mathcal{U}\{7, \ldots, 10\}$, a per-turn silence schedule $s_1, \ldots, s_T$ with $s_t \sim \mathcal{U}\{0, \ldots, 3\}$ specifying the number of contiguous `<sil>` infill slots at the start of each turn, and a per-turn substance count $n_t \sim \mathcal{U}\{1, \ldots, 5\}$ specifying the number of substance lines after the infill block. Every position in the skeleton is a typed placeholder (`<USER>`, `<INFILL_N>`, `<THOUGHT_N>`, `<RESPONSE_N>`), and the `thoughts` array already contains exactly s_t leading `"<sil>"` strings followed by `<THOUGHT_N>` placeholders. Pinning the silence schedule into the JSON structure makes it a property of the prompt rather than something the LLM must obey through natural-language instructions alone. A representative turn skeleton for $s_t = 2, n_t = 3$ is:

```
{
"user": "<USER>",
"thoughts": ["<sil>", "<sil>", "<THOUGHT_1>", "<THOUGHT_2>", "<THOUGHT_3>"],
"response": ["<INFILL_1>", "<INFILL_2>", "<RESPONSE_1>", "<RESPONSE_2>", "<RESPONSE_3>"]
}
```

The skeleton is wrapped in a `conversation` array of T such turns and serialized as JSON.

Prompt assembly. We use a three-message chat structure: a user message containing the rendered instruction template and one in-context example conversation, a fixed assistant acknowledgment that the model will preserve the scaffold structure, and a final user message containing the JSON skeleton to be filled. The fixed acknowledgment pushes the model past its planning step and makes the skeleton the most recent input it conditions on. The freeform prompt template is shown in Prompt B.1. The `{example}` slot at the bottom is filled with a single hand-authored example conversation per domain, which anchors the style of *thoughts* (terse, single-idea, factual) and *responses* (conversational, paired one-to-one with the

corresponding thought). NOTE: the prompt has been modified slightly to fit within the document. To precisely re-create our setup, use the full prompt files from the project repository.

Prompt B.1: Freeform conversation generation template

```
Generate a {num_turns}-turn conversation with two people, P1 and P2.
    * All conversations are spoken out loud, so spell out any numbers or
      expressions and remove quotation marks around words.
    * Both sides should be proactive and contribute to the conversation.
    * P1 is {P1}
    * P2 is {P2}
    * Conversation topic: {topic}
    * Conversation MUST be EXACTLY {num_turns} TURNS LONG and end naturally.
    * All turns MUST have content.
    * Allowed characters in user, thoughts, and response strings: English
      letters, digits, spaces, and these punctuation marks ONLY:
      . , ! ? ; : ' - / ( ) "
      Spell out anything else (e.g. "and" not "&", "dollars" not "$").
    * Do not include names; all references should be in first or second
      person.

Response Rules (how P2 actually speaks):
    * The response array represents P2 speaking in real time, line by line.
    * Some lines are "infill": contextually appropriate dialogue P2 says
      BEFORE having the full answer. These are NOT generic fillers like
      "hmm" or "let me think." Instead, they are natural continuations of
      the conversation: acknowledging what P1 said, restating part of the
      question, offering a relevant opinion, showing understanding, and so
      on.
    * The remaining lines deliver substantive information conversationally.
    * Each response entry is the conversational delivery of its
      corresponding thought. Keep responses tight rather than
      paragraph-length.
    * VARY the number of sentences per turn; every one should be different.
    * Turns should be standalone (do not refer to previous turns).

INFILL LINE SPECIFICATION:
    Each turn has a specified number of infill lines.
    Good infill lines:
      - Acknowledge or react to what P1 specifically said
      - Restate or reflect on the question in P2's own words
      - Share a brief relevant opinion that connects to the topic
      - Show genuine engagement with the specific situation
    Bad infill lines (do NOT produce):
      - Generic fillers ("Hmm", "Let me think", "Good question")
      - Empty acknowledgments that could apply to any conversation
      - Phrases that stall without adding contextual value

    Per-turn infill counts (MUST be followed exactly):
{sil_schedule_text}

    * If a turn specifies 0 infill lines, P2 jumps straight to substance.
    * If a turn specifies N infill lines, the first N entries in response
      must be infill, and the first N entries in thoughts must be "<sil>".
    * After the infill lines, each turn MUST have between {substance_min}
      and {substance_max} substantive lines.

INFILL QUALITY REQUIREMENTS:
    * Every infill line must be SPECIFIC to what P1 just said.
    * Do NOT reuse the same phrase or pattern across turns.
    * Infill lines should vary naturally and flow into the substantive
      content that follows.

Output in JSON Format (ONLY the JSON, no commentary):
    * Wrap output in a "conversation" array containing turn objects.
    * Each turn object must have:
        * user:     What P1 says
        * thoughts: Concise, direct information for each part of P2's turn
```

```
        * response: How P2 actually says it in conversation
    * ALL FIELDS MUST BE NON-EMPTY.
    * thoughts and response arrays MUST have the same number of elements.

Thoughts rules (the core information P2 wants to convey):
    * Infill entries MUST be exactly "<sil>".
    * Non-infill thoughts should read like a knowledgeable person's
      direct, concise answer. No small talk, no personality, no filler;
      just the core information.
    * SAME VOICE AS RESPONSE. The thought IS the content P2 conveys, not
      a self-instruction or third-person description.
    * ONE IDEA PER THOUGHT. NO compound sentences linked by "because",
      "while", "and", "which", "since", "so", or semicolons that combine
      distinct ideas.
    * If you have more ideas than thought slots, pick the most important
      and drop the rest.
    * Each thought must capture the full meaning of its corresponding
      response entry, in order.

You will be given a JSON scaffold to complete. Replace each placeholder:
- <USER>:       What P1 says
- <THOUGHT_N>:  Concise, direct, factual thought
- <RESPONSE_N>: Conversational delivery of the corresponding thought
- <INFILL_N>:   Contextual dialogue continuing the conversation
- "<sil>" entries must remain exactly as "<sil>"

Do NOT add, remove, or reorder any entries. Return ONLY the completed
JSON.

Example conversation:
{example}
```

B.2 DSTC8 Schema-Guided Scaffolded Generation

The freeform track produces stylistically homogeneous, opinion- and advice-heavy dialogues but underrepresents goal-directed slot-filling exchanges (bookings, queries, and transactional confirmations). To broaden coverage, we add a second track that takes existing task-oriented dialogues from DSTC8 (Rastogi et al., 2020) and re-skins them into the conversational infill format, keeping the original user and system content while letting the LLM author the infill and response delivery.

Source corpus. We sample from the DSTC8 `train` split, which contains 26 services spanning roughly 16 domains (banks, buses, calendar, events, flights, homes, hotels, media, movies, music, rental cars, restaurants, ridesharing, services, travel, weather, and others). Each dialogue is a sequence of strictly alternating USER and SYSTEM turns with verbatim utterances and service annotations.

DSTC8-to-scaffold conversion. Each dialogue is converted to a conversation scaffold by walking through turns in order and pairing every USER turn with the following SYSTEM turn:

- the USER utterance is pinned verbatim as the turn's `user` string;
- the SYSTEM utterance is sentence-segmented using a regex split on sentence-final punctuation; the resulting sentences become the substance entries of the `thoughts` array;
- a turn-specific silence count $s_t \sim \mathcal{U}\{0, \ldots, 3\}$ is prepended as `"<sil>"` entries, matching the freeform convention;
- the `response` positions are emitted as `<INFILL_N>` and `<RESPONSE_N>` placeholders to be filled by the LLM.

Unlike the freeform track, the LLM in this track fills only the `response` array; `user` and `thoughts` entries are pre-filled with real DSTC8 content and must be reproduced verbatim (enforced by structural validation, §B.3).

Inference-time visibility constraint. Because each scaffold encodes a complete DSTC8 dialogue, the LLM sees the entire conversational arc, including all future user utterances and all future SYSTEM content, while authoring infills for the earliest turns. This creates a causality mismatch between generation and inference: at deployment, the ConvFill `Talker` only sees turns that have already been spoken (the previous turn and the current turn so far), but the generating LLM has access to the whole future of the dialogue and tends to leak details from later turns into earlier infills. A common failure is producing an infill that references a proper noun (a restaurant name, a movie title, or an address) that has not yet been mentioned at the point in the dialogue where the infill is supposed to be spoken.

We address this in two places. First, the scaffold prompt template defines an explicit *visibility window*: when generating an infill at turn t, the model is told to ground it only in the previous turn (the user message and the assistant response) and the current turn so far (the user message and any earlier infills already placed). Second, Stage 4 of the validation cascade enforces the constraint programmatically using a proper-noun visibility check (§B.3), giving an extra layer of robustness against leaks that slip past the prompt.

The scaffold prompt template is largely identical to the freeform one in Prompt B.1; it removes the topic and turn-count slots (both implicit in the scaffold), replaces the freeform thought rules with the response rules below, and adds the visibility-window constraint. The substantive changes are shown in Prompt B.2.

Prompt B.2: DSTC8 scaffold deltas (substantive additions to Prompt B.1)

```
You are completing a partial conversation between two people, P1 and P2.
The "user" turns and the "thoughts" arrays are already filled in by an
external source; you only fill the response arrays.
    * P1 is {p1}
    * P2 is {p2}
    * The conversation comes from a {service_pretty} domain; P2 is a
      knowledgeable assistant for that domain.

VISIBILITY WINDOW:
    Infill visibility: when P2 produces an infill line, the only visible
    context is the previous turn (P1's user message and P2's complete
    response) and the current turn so far (P1's user message and any
    earlier infill lines already produced in this turn's response array).
    Each infill must be grounded only in that visible context.

    Concrete example of the most common failure mode:
    Suppose in turn 1 the user asked to play "Casablanca" and the
    assistant queued it. Many turns later, the dialogue ends:

      previous turn: user "Yes, play now."
                     response "Great, your movie is starting now! Enjoy!"
      current turn:  user "Thank you. Good bye."
                     thoughts ["<sil>", "<sil>", "<sil>", "you're welcome."]

      WRONG infill: "Hope you enjoy Casablanca."
        ^ "Casablanca" was named MANY turns ago, not in the previous turn
          or the current turn. Forbidden, even though it sounds natural.
      RIGHT infill: "It was my pleasure helping you out today."
        ^ grounded in the immediate goodbye exchange.

RESPONSE-ONLY FILLING:
    * user strings MUST be copied verbatim. Any change is a failure.
    * thoughts non-"<sil>" entries MUST be copied verbatim. Any change
      is a failure.
    * "<sil>" entries in thoughts MUST remain exactly as "<sil>".
    * Each response must capture the FULL meaning of the corresponding
      thought and INTRODUCE NO NEW INFORMATION beyond what the paired
      thought contains. No extra facts, no new offers, no new follow-up
      questions, no extra details.
```

Check	Failure condition
JSON parse	Output is not valid JSON.
Placeholder leak	<USER>, <THOUGHT_N>, etc. remain in the output.
Required fields	Any turn missing `user`, `thoughts`, or `response`.
Turn count	$T \notin [7, 10]$ (freeform only; scaffold uses DSTC8 length).
Equal-length arrays	|thoughts| $\neq$ |response| in any turn.
Sil count	$\#\{$<sil>$\} \neq s_t$ in turn t.
Sil contiguity	<sil> not in positions $0, \ldots, s_t - 1$, or any <sil> at position $\geq s_t$.
Substance count	$n_t \notin [1, 5]$ (freeform) or $[1, 10]$ (scaffold).
Min substance length	Any non-<sil> thought shorter than 5 characters (10 in scaffold).
Character set	Any disallowed symbol in `response` (and in `user`/`thoughts` for freeform; relaxed for scaffold since DSTC8 source can contain $, %, etc.).
Filler diversity	Any infill phrase used more than $2\times$ in the conversation.
Scaffold pin	(scaffold only) `user` or non-<sil> `thoughts` not byte-equal to source.

Table 5: Stage 1 structural validation checks.

B.3 Validation Cascade

Every generated conversation passes through a four-stage cascade. Stages 1–3 apply to both tracks; Stage 4 applies only to the DSTC8 scaffolded track, where the source content gives the validator extra ground truth. A conversation must pass every stage to be admitted; a failure at Stage 1 triggers a single in-chat correction request, and a failure at any stage that survives correction counts against the per-topic retry budget. Threshold values reported below were chosen by inspection.

Stage 1: Structural validation. Checks performed on each turn and the conversation as a whole are summarized in Table 5. Key rules: <sil> entries must be contiguous at the start of the `thoughts` array; the per-turn <sil> count must match the scaffold; `thoughts` and `response` arrays must have equal length per turn; no infill phrase (case-folded and trimmed) may be reused more than two times across the entire conversation (`MAX_FILLER_REUSE = 2`); and generated text uses a restricted spoken character set (English letters, digits, whitespace, and the punctuation `. , ! ? ; : ' - / ( ) "`).

On a structural failure, the pipeline issues one in-chat correction request: the model sees its own bad output and a targeted error message, and is asked to return a corrected JSON in the same chat. If the correction also fails, the topic counts as one attempt and is retried from scratch with a fresh schedule.

Stage 2: NLI semantic validation. We score every non-<sil> (thought, response) pair using a publicly available DeBERTaV3-base model fine-tuned on MNLI (He et al., 2023; Williams et al., 2018)[3], treating the thought as the premise and the response as the hypothesis. Two gates are applied:

- **Per-pair gate.** For each non-<sil> pair (t_i, r_i) in any turn, reject the conversation if the contradiction probability $P(\text{contradiction} \mid t_i, r_i) > 0.20$.
- **Whole-turn gate.** For each turn, the premise is the user utterance concatenated with all non-<sil> thoughts, and the hypothesis is the user utterance concatenated with all response entries. Reject if the whole-turn contradiction probability exceeds 0.30.

The per-pair gate catches local divergences where a single response changes the meaning of its paired thought. The whole-turn gate catches aggregate drift where each pair is individually plausible but the turn as a whole reframes the underlying message.

Stage 3: BERTScore alignment validation. NLI catches semantic contradiction within a (thought, response) pair but not *positional misalignment*: a response at position i whose content semantically belongs to position j in the same turn. We catch this with BERTScore (Zhang et al., 2020) F1 using `microsoft/deberta-xlarge-mnli` (layer 40), applying two constraints at every non-<sil> position i:

- **Anchor.** $\text{F1}(t_i, r_i) \geq 0.70$: the response at position i must be at least moderately similar to the thought at position i.

[3]The `MoritzLaurer/DeBERTa-v3-base-mnli` checkpoint from the Hugging Face Hub.

Setting	Value
Generation LLM	Claude Opus 4.6
Decoding temperature	1.0
Max output tokens	4000
Turn count T (freeform)	$\mathcal{U}\{7, 10\}$
Turn count T (scaffold)	DSTC8 dialogue length
Per-turn sil count s_t	$\mathcal{U}\{0, 3\}$
Per-turn substance n_t (freeform)	$\mathcal{U}\{1, 5\}$
Per-turn substance n_t (scaffold)	DSTC8 sentence count
Min substance length	5 chars (freeform), 1 (scaffold)
Max infill phrase reuse	2
NLI model	DeBERTaV3-base-MNLI
Per-pair contradiction max	0.20
Whole-turn contradiction max	0.30
BERTScore model	DeBERTa-xlarge-MNLI, layer 40
Anchor F1 floor	0.70
Scaffold thought F1 floor	0.95

Table 6: Generation and validation hyperparameters.

- **Cross.** $\text{F1}(t_i, r_i) > \max_{j \neq i} \text{F1}(t_i, r_j)$: the response at position i must be *more* similar to its paired thought than any other response in the same turn (infill positions included). If some other response is more aligned with t_i, the arrays are likely transposed or duplicated, and we reject.

Stage 4: Scaffold-preservation checks (DSTC8 only). Two additional checks run for the scaffolded track.

- **Thought consistency.** For each turn, we compute BERTScore F1 between the joined non-`<sil>` thoughts and the verbatim source SYSTEM utterance from DSTC8. Reject if $\text{F1} < 0.95$. The `thoughts` array is already string-pinned by Stage 1, so F1 should be near 1.0; this floor is a backstop against drift introduced by sentence-splitting edge cases.
- **Proper-noun visibility window.** This is the programmatic implementation of the visibility constraint motivated in §B.2. We extract Title-Case mid-sentence proper nouns from each response line using a regex (*Casablanca*, *San Francisco*, *Auzerais Avenue*). For each detected proper noun in $\text{response}_{t,i}$, we check case-insensitive membership in the union of (a) the current turn's user utterance, (b) the current paired thought $\text{thoughts}_{t,i}$, (c) earlier non-`<sil>` thoughts in turn t, (d) the previous turn's user utterance, and (e) the previous turn's non-`<sil>` thoughts. Previous-turn responses are intentionally excluded from the allowed corpus: they are LLM-generated and may themselves carry leaked content. Any proper noun outside this window causes the conversation to be rejected.

B.4 Generation Settings

Table 6 reports the hyperparameters needed to reproduce the dataset. The full per-domain configuration JSON files will be released with the dataset.

B.5 Output Format and Yield

Each accepted conversation is emitted as a single JSON object on its own line of a JSONL file, with the shape `{"conversation": [{"user", "thoughts", "response"}, ...]}`. Scaffolded conversations additionally carry a top-level `scaffold_metadata` object recording the DSTC8 service and dialogue ID.

Table 7 reports the per-track yield. The released dataset contains 6,005 freeform conversations (50,324 turns) and 2,438 DSTC8-scaffolded conversations (24,184 turns), for a total of 8,443 conversations and 74,508 turns.

Track / Domain	Convs.	Turns	Turns/Conv.
Advice	1,000	8,308	8.31
Assistant queries	1,000	8,408	8.41
Customer service	1,000	8,313	8.31
Education	1,000	8,421	8.42
Medical	1,000	8,400	8.40
Event planning	1,005	8,474	8.43
Freeform subtotal	6,005	50,324	8.38
DSTC8 scaffolded	2,438	24,184	9.92
Total	**8,443**	**74,508**	8.83

Table 7: Per-track conversation and user-infill turn counts in the released dataset.

C Model Training Details

This section contains details about the model training pipeline and model training parameters used in our experiments. It is intended to provide an overview of how to reproduce or extend the experiments in this paper.

C.1 Base SLM Models

The table below contains details about the Base SLM models that were fine-tuned to perform the conversational infill task, including the Hugging Face ID of each model.

Model Family	# Param.	Size (BF16)	Size (INT8)	Hugging Face ID
Gemma3 IT	270M	511 MB	272 MB	`google/gemma-3-270m-it`
Gemma3 IT	1B	1.86 GB	1013 MB	`google/gemma-3-1b-it`
Qwen3	0.6B	1.40 GB	604 MB	`Qwen/Qwen3-0.6B`
SmolLM2 IT	135M	257 MB	136 MB	`HuggingFaceTB/SmolLM2-135M-Instruct`
SmolLM2 IT	360M	690 MB	367 MB	`HuggingFaceTB/SmolLM2-360M-Instruct`
SmolLM2 IT	1.7B	3.19 GB	1.69 GB	`HuggingFaceTB/SmolLM2-1.7B-Instruct`
Llama 3.2 IT	1.2B	2.30 GB	1.22 GB	`meta-llama/Llama-3.2-1B-Instruct`

Table 8: **Fine-tuned Base SLM models.** Seven models were fine-tuned to perform the conversational infill task. "IT" indicates that the base model was originally instruction-tuned.

C.2 Frontier Models

Model Name	Accessed Through	Model Identifier	Role(s)
GPT-5.5	OpenAI API	`gpt-5.5-2026-04-23`	ConvFill `Reasoner`
Claude Opus 4.7	Anthropic API	`claude-opus-4-7`	ConvFill `Reasoner`
Gemini 3.1 Pro	Google AI Studio	`3.1-pro-preview-01-2026`	ConvFill `Reasoner`
GPT-4o	OpenAI API	`gpt-4o-2024-08-06`	Automated Grading
Claude Opus 4.6	Anthropic API	`claude-opus-4-6`	Dataset Generation

Table 9: **API-based models used for development, inference, and testing.** Model identifiers denote the specific tagged versions used in this work.

C.3 Control Sequence Expansions for Different `Talker` Models

We use existing model control tokens as boundary tokens to mark spans for user, assistant, and knowledge segments of training and inference examples.

Model	Control Token Sequence	Expansion
Gemma 3	[USER] [KNOWLEDGE] [ASST] [END]	<start_of_turn>user <start_of_turn>knowledge <start_of_turn>model <end_of_turn>
Llama 3.2	[USER] [KNOWLEDGE] [ASST] [END]	<|start_header_id|>user<|end_header_id|> <|start_header_id|>knowledge<|end_header_id|> <|start_header_id|>assistant<|end_header_id|> <|eot_id|>
Qwen3	[USER] [KNOWLEDGE] [ASST] [END]	<|im_start|>user <|im_start|>knowledge <|im_start|>assistant <|im_end|>
SmolLM2	[USER] [KNOWLEDGE] [ASST] [END]	<|im_start|>user <|im_start|>knowledge <|im_start|>assistant <|im_end|>

Table 10: Control sequence expansions per model. [KNOWLEDGE] is a new role introduced by our pipeline and mapped through each model's existing control-token template.

C.4 Shared Training Parameters for Talker Models

The following describes the Talker model training parameters. Each Talker model was trained on a 90/10 train/evaluation split made up of the dataset in Appendix B. Models were trained on either 4 NVIDIA RTX 6000 GPUs or 4 NVIDIA A40 GPUs. GPU hours are documented in Table 11.

Model	Parameters	Batch Size	Eff. Batch Size	Chat Template	GPU Hours	Cost (USD)
SmolLM2 IT	135M	4	16	ChatML	2.80	$3.07
Gemma 3 IT	270M	4	16	Gemma	4.53	$4.96
SmolLM2 IT	360M	4	16	ChatML	11.60	$12.70
Qwen3	0.6B	4	16	ChatML	6.94	$7.60
Gemma 3 IT	1B	2	8	Gemma	31.70	$34.71
Llama 3.2 IT	1B	2	8	Llama-3	15.60	$17.08
SmolLM2 IT	1.7B	2	8	ChatML	48.93	$53.58
Total					**122.10**	**$133.70**
Shared hyperparameters (all models)						
Learning rate	1×10^{-4}					
LR schedule	Cosine with warmup					
Warmup steps	200					
Weight decay	0.01					
Gradient accumulation	4					
Epochs	1					
Special tokens	<sil>					

Table 11: Training configurations for all fine-tuned Talker models. Effective batch size = batch size $\times$ gradient accumulation steps. GPU hours = wall-clock time $\times$ 4 GPUs. Cost estimated at $1.095/GPU/hr (Google Cloud, 2026).

D User Study Details

This section provides an in-depth description of our user study procedures for the conversational infill study.

D.1 Ethical Considerations

Our study was approved by the Institutional Review Board at our institution. Participants engaged in sessions of approximately 45 minutes for both the interaction and prompt grading portions of the study. Participants were informed that their participation in our study was voluntary and that the data generated during the study would be included in an academic publication. Participants were compensated for their participation.

D.2 Study Instructions

Participants were provided with the following prompt to begin the study.

You will interact with three different systems. After interacting with each system, you will fill out a survey with some questions about the experience. There will be three blocks in the study where you will perform three different tasks. At the end of the block, you will rank each of the systems in order of preference.

You will be provided with a sheet to guide your conversation. The aim of this sheet is to give you a goal to accomplish and suggest things to say. For each of the tasks, use the system to gather the information and insights you desire. Ask whatever questions feel natural to you; there is no script to follow. Work through the task in whatever order makes sense. Converse with the system for several turns. Once you have formed an opinion about it (e.g., after 4–5 turns of conversation), you may stop.

As you interact with each agent, pay attention to how you feel about the interaction. Notice things like:

1. *How easy or difficult it is to get the information you need*
2. *How the agents respond*
3. *How helpful the agent's responses are*
4. *How the conversation flows*
5. *Anything that you like or dislike about the interaction*

After the introductory prompt, participants were introduced to the metrics that they would use for Likert-scale ratings later in the study. The following descriptions were provided, without labels. However, we provide the labels here for clarity and so that they can be referenced with the body of the text:

1. *Latency: The system responded promptly enough to maintain a natural conversation.*
2. *Clarity: I could understand the words the system said.*
3. *Fluency: I could understand the ideas expressed by the system.*
4. *Response Length: The length of the responses felt appropriate.*
5. *Coherence: Sentences related to each other and progressed in a logical fashion.*
6. *Task Completion: The system's responses provided the information I needed to complete the assigned task.*
7. *Naturalness: The language used felt natural and appropriate for the context.*
8. *Satisfaction: I was satisfied with my conversation(s) with the system.*

D.3 Study Tasks

The following section contains details about the tasks that were assigned to participants in the categories of Direct, RAG, and MCP.

D.3.1 Direct Tasks

Direct tasks refer to task-oriented dialogues that can progress without necessarily having access to tools, external databases, or external tools. Examples include trip planning, looking up recipes, brainstorming ideas for writing, and planning events. Three different scenarios of this style were presented to

participants, and they were instructed to choose the one that they wanted to have a conversation about. Locations and names are redacted to preserve anonymity during the submission process. All three system configurations (Base SLM, Frontier, and ConvFill) are used for this task.

Scenario 1: Birthday Party Planning Your task is to plan a birthday party for your friend using the agent. You need to figure out where to host it, what activities to include, the costs involved, how to decorate, and any other details that matter to you. Talk to the agent naturally and ask whatever questions you need. There is no single right way to do this; explore the information in whatever order makes sense to you, and feel free to change your mind as you learn more. Suggestions for conversation:

- Venue(s) for hosting 20 people for a birthday party in [LOCATION]
- Primary activity or activities for the party (you can choose)
- Cost estimates for activities (your budget is decided by you)
- Vendor policies (e.g., can they accommodate outside food/drinks at any of the venues?)
- Decoration ideas that match the theme (you choose the theme)
- Timeline/logistics for setup (e.g., how far in advance to order some supplies)
- Any additional details you discovered or decided on

Scenario 2: Train Travel Your task is to plan a train trip from [LOCATION A] to [LOCATION B] using the agent. You need to figure out which route(s) make sense, how long the trip takes, what it costs, what discounts you might qualify for, and what to expect in terms of food and amenities on the train. Talk to the agent naturally and ask whatever questions you need. There is no single right way to do this; explore the information in whatever order makes sense to you. Suggestions for conversation:

- Available train route options from [LOCATION A] to [LOCATION B]
- Whether direct routes exist or if connections are required
- Total travel time for your chosen route(s)
- Approximate ticket price(s)
- Student discount options and how to apply them
- Food options available on the train
- Onboard amenities (sleeping accommodations, showers, etc.)
- Any other details you discovered or questions you resolved

Scenario 3: Baking a Cake Your task is to use the agent to find a recipe for making a cake that uses Greek yogurt. You need to figure out what recipe to use, how to incorporate the texture, ingredients, and toppings you want, how long it will take to prepare and bake, and whether you can realistically finish it before your friends arrive. Talk to the agent naturally and ask whatever questions you need. There is no single right way to do this; explore the information in whatever order makes sense to you. Suggestions for conversation:

- Recipe for a Greek yogurt cake
- How to make the cake fluffy
- How to incorporate other ingredients that you like (you can choose)
- Some sort of topping that will go with the cake
- Total time needed for preparation and baking
- Whether the recipe fits your timeline before your friends arrive
- Any ingredient substitutions or variations you want

D.3.2 RAG Task

This task connects the conversational model to a retrieval-augmented generation (RAG) database. All three system configurations (Base SLM, Frontier, and ConvFill) are used for this task. The RAG database was constructed by scraping a set of webpages related to a certain institution. The RAG system operates as a two-stage retrieve-then-rerank pipeline over a pre-built index of [INSTITUTION]'s documentation. At inference time, the user query is embedded using OpenAI `text-embedding-3-large` (3,072 dimensions)

and L2-normalized before being issued to a FAISS index containing 367 pre-embedded document chunks. The top 30 candidate chunks are retrieved in this first stage. The 30 candidates are then passed to a cross-encoder reranker, which scores each (query, chunk) pair jointly and reorders the candidates by relevance. The top three reranked chunks are passed to the downstream language model for grounded response generation.

The participant prompt for the RAG task is as follows:

Your friend is a [PERSON] at the [INSTITUTION] and they are just getting started. They gave you a list of some questions that they want to find out about, and your job is to answer those questions. You do not need to specify the institution; simply ask the questions. Your task is to help your friend find answers to their questions. There is no single right way to do this; explore the information in whatever order makes sense to you, and feel free to change your mind as you learn more.

Just like above, a list of suggested topics is provided. However, to maintain anonymity during the submission period, this list is excluded.

D.3.3 MCP Task

In the MCP task, models are connected via Model Context Protocol to an IMAP email server. In this task, we support only the Frontier and ConvFill system configurations because Base SLM models are unable to make properly formatted MCP calls.

The participant prompt for the MCP task is as follows: You are a busy person, and you need to check some things related to your email. You have a conversational agent that can help you out. Use the conversational agent to ask questions about your email. The agent can see your inbox and will respond appropriately.

Your task is to answer some of the following questions on your checklist. There is no single right way to do this; explore the information in whatever order makes sense to you, and feel free to change your mind as you learn more. Suggestions for conversation:

- Number of emails in inbox: _____
- Any unread emails: _____
- `[EVENT]` date: _____
- Summary of last on-call handoff: _____
- Any build failures: _____
- Rust updates for the week: _____
- Deadline for Q1 roadmap review: _____

D.4 Participant Condition Assignment

The eighteen participants were assigned to three experimental conditions (tasks: Direct, RAG, MCP) in a fully counterbalanced Latin square design, such that each condition appeared exactly six times in each ordinal position (first, second, third) across participants. Within each condition, participants evaluated three system configurations: Base SLM, Frontier, and ConvFill (except for MCP which had only Frontier and ConvFill). The presentation order of these configurations was also counterbalanced across participants to control for ordering effects. `Talker` and `Reasoner` models were drawn from two pools (Llama 3.2, Qwen3 0.6B, and Gemma 270M for `Talker`; Claude Opus 4.7 and OpenAI GPT-5.5 for `Reasoner`) and assigned to participants in a rotating manner. Each participant encountered all three conditions and all three model types (MCP for Base SLM was skipped), with the specific ordering of both conditions and models varied to prevent position bias from confounding condition or model preference judgments.

D.5 Live Interaction Setup

During the study, participants interacted with a web-based voice system. The system features a press-and-hold button through which the participant records audio to send to the system. The speech-to-text service that we use is faster-whisper (SYSTRAN, 2023), a re-implementation of OpenAI's Whisper built on the CTranslate2 inference engine. We deploy the base Whisper model ($\sim$74M parameters) running on CPU

with INT8 quantization and a beam size of 1 (greedy decoding). Voice activity detection is enabled via the Silero VAD integration (Silero Team, 2024) to filter non-speech regions before transcription.

Spoken responses are synthesized on the host machine via the macOS `say` command, which delegates to Apple's system Speech Synthesis framework (Apple, 2026). We used the host's default system voice at the voice's default speaking rate. LLM output is stripped of Markdown before being passed to `say`.

D.5.1 Live Interaction Example

The following shows a user view of the system during interaction. All Base SLM models and ConvFill `Talker` models are run with INT8 quantization through MLX on an Apple M2 SoC. Although there was an option to type an input, all participants used voice input to simulate realistic voice-based interaction.

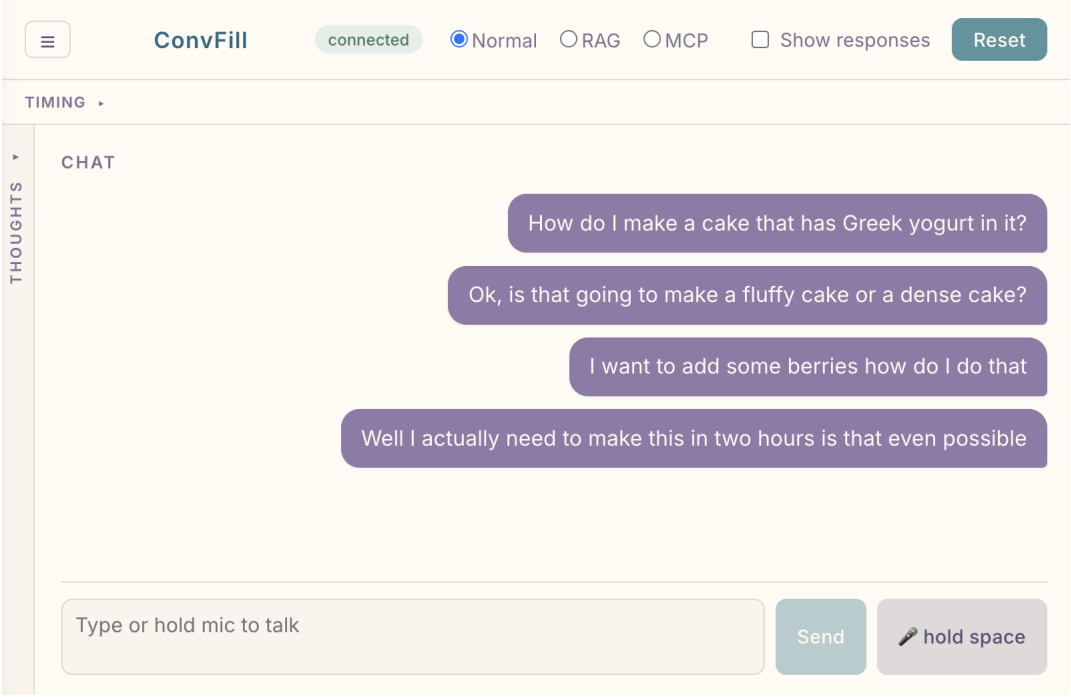

Figure 6: **Live interaction system.** Participants see their transcribed text but do not see the model responses; they are only able to hear the responses spoken out loud.

D.6 Infill Grading

After their live interactions in the study, participants were each asked to grade 12 samples consisting of a user utterance, a set of external knowledge chunks, and a set of conversational responses. Below is an example of a sample that was provided to the participants.

User: Try anything with Indian food; that would work as well. I will need the address if you find one, thanks!

Thoughts	Conversational Response
• I have found 4 Indian restaurants on the east side. • There are two moderate and two expensive. • Did you have a price range in mind?	Indian food, great choice! Let me look into what's available in the area. I found four Indian restaurants on the east side. There are two moderate and two expensive options. Did you have a price range in mind?

Table 12: **Example of infill grading.** Samples of this format were provided to participants for grading.

Participants graded a total of 108 unique samples. Each sample was graded by two participants, yielding a total of 216 grading examples collected from participants.

Equal numbers of samples were selected from each of the datasets described in this paper (27 per dataset): MultiWOZ, Everyday Conversations, LLAMA1, and SimpleQA. Each example was constructed through the following method. First, a dataset was fixed. In the case of single-turn datasets (LLAMA1 and SimpleQA), a single turn was selected without replacement. Then, a Frontier model was selected (one of Claude Opus 4.7, Gemini 3.1 Pro, and GPT-5.5), and a `Talker` model was randomly selected from one of the `Talker` models in Section 5.3 without replacement, such that each model was selected at least once, to produce the conversational response conditioned on the external knowledge (referred to as "Thoughts" in Table 12).

Participants were given the following descriptions of each of the metrics for grading:

- **Coverage:** Key facts represented in the thought are reflected in the response. Given the context of the user's query, the important information from the thought is reflected in the response.

  5: All important information is in the response.
  4: Some information is missing, but it does not affect the takeaways in a major way.
  3: Some information is missing, but the user will be led in the right direction even if they do not have the complete picture.
  2: Some information is conveyed, but not the important parts.
  1: Information from the thought does not appear in the response.

- **Faithfulness:** Statements in the response do not contradict the thoughts or introduce additional unsupported claims.

  5: Nothing in the response contradicts the thoughts or introduces unsupported claims; there are unambiguously no distortions of meaning.
  4: Small differences arise from paraphrasing or conversational rephrasing, but they are at worst inconsequential technicalities.
  3: There is at most one minor inaccuracy, distortion, or unsupported claim, but this would not significantly affect the user's overall understanding.
  2: There is a clear inaccuracy, fabrication, or distortion that would lead the listener to a wrong conclusion on some part of the query.
  1: There are multiple contradictions or fabrications; the listener leaves with a substantively wrong understanding.

- **Helpfulness:** The conversational response answers the user's query. This relates to the form and structure of the response, not whether it appropriately incorporates the contents of the thoughts.

  Yes: The response answers the user's question or statement, regardless of whether it is correct or incorporates the thoughts.
  No: The response does not answer the user's question, even if it accurately incorporates the thoughts.

Inter-rater reliability was strong across all three evaluation dimensions (Coverage (1–5), Faithfulness (1–5), Helpfulness (Y/N)), with Krippendorff's α of 0.672 (ordinal) for Coverage, 0.67 (ordinal) for Faithfulness, and 0.751 (nominal) for Helpfulness, indicating substantial agreement among human annotators.

We also calculated human-LLM agreement with the grading presented in Section D.6 for Coverage, Faithfulness, and Helpfulness. Human-LLM agreement was moderate, with Kendall's τ_b of 0.55 and 0.554 for the ordinal Coverage and Faithfulness dimensions, respectively, and point-biserial r of 0.69 for the binary Helpfulness dimension, with threshold agreement reaching 91.70% for Helpfulness at a mean absolute distance of 0.332–0.494 across ordinal scales.

D.7 TOST Test Details for Likert-Scale Ratings

Metric	Frontier mean	ConvFill mean	Diff	U_{lower}	p_{lower}	U_{upper}	p_{upper}	Equiv.
Latency	3.46	4.24	-0.78	1326.0	.795	576.0	$<.001$	No
Clarity	4.81	4.69	$+0.13$	2566.0	$<.001$	582.0	$<.001$	Yes
Fluency	4.83	4.56	$+0.28$	2615.0	$<.001$	864.0	$<.001$	Yes
Response Length	4.07	3.94	$+0.13$	2000.0	$<.001$	1062.0	.007	Yes
Coherence	4.61	4.15	$+0.46$	2385.0	$<.001$	1089.0	.010	Yes
Task Completion	4.35	4.31	$+0.04$	2097.0	$<.001$	940.0	$<.001$	Yes
Naturalness	4.41	3.83	$+0.57$	2352.0	$<.001$	1441.0	.459	No
Satisfaction	3.80	3.81	-0.02	1863.0	.006	1030.0	.004	Yes

Table 13: **Full TOST results.** Wilcoxon rank-sum results for Frontier vs. ConvFill, aggregated across all tasks ($N = 54$ per system, $\delta = 0.5$, $\alpha = .05$). Diff = Frontier mean $-$ ConvFill mean. p_{lower} tests H_0: Frontier $-$ ConvFill $\leq -\delta$; p_{upper} tests H_0: Frontier $-$ ConvFill $\geq +\delta$. Equivalence is declared when both $p < .05$.

D.8 Test Details for User Rankings across Tasks

Pairwise ranking preferences between the Frontier and ConvFill configurations were assessed using Fisher's exact tests on rank-1 counts per task (i.e., how many participants ranked each system first). For the Direct task, the Frontier configuration received 10 rank-1 votes and ConvFill received 8 ($p = .740$). For the RAG task, ConvFill received 12 rank-1 votes compared to 5 for the Frontier configuration ($p = .044$), indicating a significant preference for ConvFill. For the MCP task, the Base SLM configuration was excluded because it could not complete tool-call interactions. Of the remaining two systems, ConvFill received 11 rank-1 votes and the Frontier configuration received 7. As neither system received any rank-3 votes in the MCP task, a two-sided binomial test was used in place of Fisher's exact test, treating each participant's rank-1 assignment as a binary choice ($p = .481$).

D.9 ConvFill Speedup, TTFR, & Filler Details

This section provides a full breakdown of the conversational infill `Talker` speedups over the conversational infill `Reasoner` as well as TTFR.

Task	n	Mean (×)	SD (×)	95% CI ±
Direct	82	7.38	4.53	0.99
RAG	73	9.20	5.75	1.34
MCP	93	19.12	12.93	2.66

Table 14: **ConvFill speedup by task.** Speedup = convfill_reasoner_latency / convfill_talker_latency. Values > 1 indicate that the filler arrived sooner than the `Reasoner` response would have.

Task	Group	Mean ± SD (ms)	95% CI ±
Direct	Base SLM	617 ± 689	160
	`Talker`	542 ± 636	140
	`Reasoner`	2947 ± 1459	321
	Frontier	2578 ± 1321	279
RAG	Base SLM	3812 ± 2392	547
	`Talker`	976 ± 1177	275
	`Reasoner`	4852 ± 1751	409
	Frontier	4048 ± 1706	409
MCP	Base SLM	–	–
	`Talker`	478 ± 300	61
	`Reasoner`	7242 ± 3850	793
	Frontier	8098 ± 4213	912

Table 15: **Latency (ms) by task and mode.** Base SLM has no MCP condition because the base SLM cannot reliably execute tool calls. ConvFill `Talker` latency is experienced by users; ConvFill `Reasoner` latency is the model's latency.

Task	Mean <sil>	Std <sil>	95% CI ±
Direct	1.16	0.40	0.09
RAG	1.34	0.53	0.12
MCP	2.35	1.12	0.23

Table 16: **<sil> filler tokens per turn by task.** <sil> token counts were recorded during the user study. Mean and standard deviation of <sil> tokens generated per turn, with 95% confidence interval half-widths.

E Full Benchmark Evaluation Results

Dataset	Model / baseline	Base SLM	Claude Opus 4.7 Talker		GPT-5.5 Talker		Gemini 3.1 Pro Talker	
			Acc.	Cond. acc.	Acc.	Cond. acc.	Acc.	Cond. acc.
SimpleQA	Frontier	–	44.9 [41.8,48.0]	–	62.7 [59.7,65.6]	–	73.1 [70.3,75.8]	–
	Reasoner	–	42.9 [39.9,46.0]	–	61.3 [58.2,64.3]	–	65.7 [62.7,68.6]	–
	SmolLM2 135M	0.4 [0.2,1.0]	38.0 [35.0,41.0]	87.4 [83.9,90.2]	55.2 [52.1,58.3]	89.1 [86.4,91.3]	60.7 [57.6,63.7]	91.9 [89.6,93.8]
	Gemma 3 270M	0.9 [0.5,1.7]	40.6 [37.6,43.7]	93.0 [90.2,95.1]	56.6 [53.5,59.6]	91.2 [88.7,93.2]	61.9 [58.8,64.9]	93.6 [91.5,95.2]
	SmolLM2 360M	1.3 [0.8,2.2]	41.2 [38.2,44.3]	94.2 [91.5,96.0]	57.7 [54.6,60.7]	93.3 [91.1,95.0]	62.8 [59.8,65.7]	94.8 [92.9,96.3]
	Qwen3 0.6B	1.4 [0.8,2.3]	42.4 [39.4,45.5]	97.2 [95.2,98.4]	60.3 [57.2,63.3]	97.6 [96.0,98.5]	64.8 [61.8,67.7]	98.5 [97.2,99.2]
	Gemma 3 1B	2.2 [1.5,3.3]	41.5 [38.5,44.6]	96.0 [93.7,97.5]	59.5 [56.4,62.5]	96.2 [94.4,97.5]	64.3 [61.3,67.2]	97.6 [96.1,98.5]
	Llama 3.2 1B	0.5 [0.2,1.2]	41.3 [38.3,44.4]	94.9 [92.4,96.6]	57.7 [54.6,60.7]	93.5 [91.2,95.2]	63.7 [60.7,66.6]	96.5 [94.8,97.7]
	SmolLM2 1.7B	2.5 [1.7,3.7]	42.5 [39.5,45.6]	97.2 [95.2,98.4]	61.0 [57.9,64.0]	98.7 [97.4,99.3]	65.1 [62.1,68.0]	98.6 [97.4,99.3]
LLAMA1	Frontier	–	88.0 [83.8,91.2]	–	85.0 [80.5,88.6]	–	86.3 [82.0,89.8]	–
	Reasoner	–	85.7 [81.2,89.2]	–	86.3 [82.0,89.8]	–	87.0 [82.7,90.3]	–
	SmolLM2 135M	34.7 [29.5,40.2]	83.7 [79.1,87.4]	95.7 [92.5,97.6]	82.7 [78.0,86.5]	95.4 [92.1,97.3]	85.7 [81.2,89.2]	97.7 [95.1,98.9]
	Gemma 3 270M	40.7 [35.3,46.3]	80.7 [75.8,84.7]	91.4 [87.4,94.3]	81.3 [76.5,85.3]	93.1 [89.3,95.6]	81.7 [76.9,85.6]	92.7 [88.9,95.3]
	SmolLM2 360M	51.3 [45.7,56.9]	84.3 [79.8,88.0]	97.7 [95.0,98.9]	85.0 [80.5,88.6]	97.7 [95.0,98.9]	85.3 [80.9,88.9]	97.7 [95.1,98.9]
	Qwen3 0.6B	41.3 [35.9,47.0]	84.7 [80.2,88.3]	96.9 [94.0,98.4]	84.0 [79.4,87.7]	96.1 [93.0,97.9]	85.7 [81.2,89.2]	97.3 [94.6,98.7]
	Gemma 3 1B	63.0 [57.4,68.3]	85.7 [81.2,89.2]	97.3 [94.5,98.7]	84.7 [80.2,88.3]	96.9 [94.0,98.4]	84.7 [80.2,88.3]	96.2 [93.1,97.9]
	Llama 3.2 1B	72.0 [66.7,76.8]	84.7 [80.2,88.3]	96.9 [94.0,98.4]	83.3 [78.7,87.1]	95.4 [92.1,97.3]	84.0 [79.4,87.7]	95.8 [92.6,97.6]
	SmolLM2 1.7B	72.0 [66.7,76.8]	85.0 [80.5,88.6]	97.7 [95.0,98.9]	86.0 [81.6,89.5]	98.5 [96.1,99.4]	84.7 [80.2,88.3]	96.2 [93.1,97.9]

Table 17: QA accuracy (%) on SimpleQA ($N = 1000$) and LLAMA1 ($N = 300$). Cells report accuracy with full 95% Wilson score confidence interval $[\mathrm{lo}, \mathrm{hi}]$ over questions. For each ConvFill `Talker` column, Conditional Accuracy restricts accuracy to questions where the corresponding `Reasoner` bullet output was judged correct. Frontier and `Reasoner` rows score hosted frontier model answers and `Reasoner` outputs, respectively; the Base SLM column compares direct Base SLM responses with ConvFill responses conditioned on hosted `Reasoner` thoughts.

Dataset	Reasoner	Model	Entailment	Non-Contradiction	Coverage	Faithfulness	Helpfulness	Raw Helpfulness
SimpleQA	Claude Opus 4.7	SmolLM2 135M	90.4 [88.8,91.9]	90.0 [88.5,91.8]	4.91 [4.87,4.94]	4.43 [4.36,4.49]	0.91 [0.90,0.93]	4.35 [4.29,4.41]
		Gemma 3 270M	90.3 [88.6,91.9]	97.3 [96.5,98.1]	4.92 [4.89,4.95]	4.59 [4.53,4.64]	0.96 [0.95,0.97]	4.55 [4.50,4.60]
		SmolLM2 360M	95.7 [94.6,96.9]	94.3 [93.2,95.4]	4.97 [4.95,4.99]	4.72 [4.67,4.77]	0.96 [0.95,0.97]	4.66 [4.60,4.70]
		Qwen3 0.6B	97.4 [96.6,98.1]	93.6 [92.4,94.9]	4.99 [4.98,5.00]	4.87 [4.84,4.90]	0.99 [0.98,0.99]	4.83 [4.80,4.86]
		Gemma 3 1B	95.8 [94.8,97.0]	92.1 [90.6,93.5]	4.97 [4.95,4.99]	4.76 [4.72,4.80]	0.98 [0.97,0.99]	4.73 [4.69,4.77]
		Llama 3.2 1B	97.1 [96.2,98.0]	89.6 [88.2,91.2]	4.99 [4.97,5.00]	4.74 [4.70,4.78]	0.98 [0.97,0.99]	4.75 [4.71,4.79]
		SmolLM2 1.7B	97.2 [96.4,98.0]	95.8 [94.8,96.8]	4.99 [4.97,5.00]	4.89 [4.86,4.92]	0.99 [0.99,1.00]	4.84 [4.81,4.87]
	GPT-5.5	SmolLM2 135M	90.3 [88.5,92.0]	89.7 [87.9,91.5]	4.89 [4.85,4.92]	4.46 [4.39,4.53]	0.91 [0.89,0.93]	4.36 [4.30,4.42]
		Gemma 3 270M	88.0 [86.0,89.8]	98.3 [97.5,99.0]	4.85 [4.81,4.90]	4.54 [4.49,4.60]	0.92 [0.90,0.94]	4.45 [4.39,4.51]
		SmolLM2 360M	95.5 [94.2,96.7]	95.5 [94.4,96.5]	4.95 [4.92,4.97]	4.72 [4.66,4.77]	0.95 [0.94,0.96]	4.64 [4.59,4.69]
		Qwen3 0.6B	96.5 [95.5,97.5]	94.0 [92.6,95.3]	4.98 [4.96,4.99]	4.86 [4.82,4.89]	0.98 [0.97,0.99]	4.82 [4.78,4.85]
		Gemma 3 1B	95.2 [94.0,96.5]	92.6 [91.1,94.1]	4.97 [4.95,4.99]	4.75 [4.71,4.79]	0.97 [0.96,0.98]	4.74 [4.70,4.78]
		Llama 3.2 1B	94.2 [92.8,95.5]	89.0 [87.2,90.7]	4.98 [4.96,4.99]	4.65 [4.60,4.70]	0.96 [0.95,0.97]	4.68 [4.63,4.73]
		SmolLM2 1.7B	98.3 [97.5,99.0]	96.9 [95.9,97.8]	4.98 [4.96,4.99]	4.92 [4.90,4.95]	0.99 [0.98,0.99]	4.88 [4.85,4.91]
	Gemini 3.1 Pro	SmolLM2 135M	89.4 [87.8,90.9]	88.9 [87.3,90.4]	4.74 [4.68,4.80]	4.36 [4.28,4.43]	0.88 [0.86,0.90]	4.24 [4.17,4.31]
		Gemma 3 270M	90.9 [89.6,92.3]	96.0 [95.0,96.9]	4.74 [4.68,4.80]	4.50 [4.44,4.56]	0.91 [0.90,0.93]	4.39 [4.32,4.46]
		SmolLM2 360M	94.9 [93.9,95.9]	93.3 [92.2,94.5]	4.80 [4.74,4.85]	4.60 [4.54,4.65]	0.91 [0.90,0.93]	4.48 [4.41,4.55]
		Qwen3 0.6B	95.8 [94.8,96.6]	93.5 [92.2,94.7]	4.80 [4.75,4.85]	4.75 [4.71,4.79]	0.94 [0.93,0.95]	4.63 [4.57,4.68]
		Gemma 3 1B	95.9 [94.9,96.8]	92.0 [90.6,93.3]	4.80 [4.74,4.85]	4.67 [4.61,4.72]	0.94 [0.92,0.95]	4.55 [4.49,4.62]
		Llama 3.2 1B	95.1 [94.0,96.0]	89.1 [87.5,90.7]	4.79 [4.74,4.84]	4.58 [4.52,4.63]	0.93 [0.91,0.94]	4.54 [4.48,4.60]
		SmolLM2 1.7B	95.5 [94.5,96.4]	94.6 [93.6,95.6]	4.80 [4.75,4.86]	4.81 [4.77,4.85]	0.95 [0.93,0.96]	4.67 [4.61,4.73]
LLAMA1	Claude Opus 4.7	SmolLM2 135M	95.8 [93.9,97.3]	94.8 [93.1,96.4]	4.99 [4.97,5.00]	4.63 [4.53,4.71]	0.98 [0.97,0.99]	4.61 [4.53,4.68]
		Gemma 3 270M	94.3 [92.7,95.8]	97.3 [95.8,98.6]	4.97 [4.93,4.99]	4.59 [4.48,4.68]	0.95 [0.92,0.97]	4.63 [4.54,4.72]
		SmolLM2 360M	97.3 [96.1,98.3]	95.4 [93.1,97.4]	5.00 [5.00,5.00]	4.86 [4.79,4.91]	0.99 [0.98,1.00]	4.83 [4.77,4.88]
		Qwen3 0.6B	95.8 [94.4,97.2]	90.6 [87.2,93.8]	5.00 [5.00,5.00]	4.83 [4.76,4.89]	0.98 [0.96,0.99]	4.85 [4.79,4.91]
		Gemma 3 1B	96.4 [95.1,97.8]	87.1 [83.4,90.6]	5.00 [5.00,5.00]	4.84 [4.77,4.89]	0.99 [0.98,1.00]	4.87 [4.82,4.92]
		Llama 3.2 1B	96.3 [94.9,97.5]	89.8 [86.8,92.9]	5.00 [5.00,5.00]	4.78 [4.72,4.84]	0.99 [0.98,1.00]	4.87 [4.82,4.92]
		SmolLM2 1.7B	96.4 [94.8,97.7]	98.6 [97.8,99.4]	5.00 [5.00,5.00]	4.88 [4.83,4.93]	0.99 [0.98,1.00]	4.91 [4.86,4.94]
	GPT-5.5	SmolLM2 135M	92.9 [90.2,95.4]	93.1 [90.7,95.3]	4.92 [4.85,4.98]	4.67 [4.59,4.75]	0.96 [0.94,0.98]	4.48 [4.39,4.57]
		Gemma 3 270M	90.7 [87.6,93.6]	97.1 [95.3,98.6]	4.93 [4.87,4.98]	4.72 [4.63,4.80]	0.96 [0.93,0.98]	4.61 [4.52,4.70]
		SmolLM2 360M	95.1 [92.9,97.3]	94.0 [91.1,96.5]	5.00 [5.00,5.00]	4.83 [4.76,4.89]	0.99 [0.97,1.00]	4.75 [4.69,4.81]
		Qwen3 0.6B	94.7 [92.1,96.9]	89.1 [85.3,92.6]	5.00 [5.00,5.00]	4.88 [4.82,4.93]	0.99 [0.98,1.00]	4.86 [4.80,4.92]
		Gemma 3 1B	96.4 [94.4,98.1]	85.6 [81.5,89.6]	5.00 [4.99,5.00]	4.87 [4.81,4.92]	0.99 [0.98,1.00]	4.85 [4.80,4.90]
		Llama 3.2 1B	96.2 [94.3,97.9]	87.8 [84.3,91.4]	5.00 [5.00,5.00]	4.76 [4.69,4.83]	0.99 [0.98,1.00]	4.81 [4.75,4.87]
		SmolLM2 1.7B	95.2 [93.0,97.2]	97.9 [96.2,99.2]	5.00 [5.00,5.00]	4.88 [4.82,4.93]	0.99 [0.98,1.00]	4.87 [4.82,4.91]
	Gemini 3.1 Pro	SmolLM2 135M	96.5 [95.0,98.0]	94.5 [92.7,96.1]	4.99 [4.98,5.00]	4.75 [4.67,4.82]	0.97 [0.96,0.99]	4.67 [4.59,4.74]
		Gemma 3 270M	96.3 [94.7,97.8]	95.8 [93.8,97.5]	4.99 [4.97,5.00]	4.73 [4.64,4.80]	0.96 [0.94,0.98]	4.73 [4.66,4.80]
		SmolLM2 360M	94.6 [92.8,96.2]	95.0 [92.5,97.2]	5.00 [5.00,5.00]	4.85 [4.78,4.90]	0.99 [0.98,1.00]	4.78 [4.72,4.84]
		Qwen3 0.6B	95.8 [93.9,97.6]	90.8 [87.7,93.8]	5.00 [5.00,5.00]	4.90 [4.84,4.95]	0.99 [0.97,1.00]	4.87 [4.82,4.92]
		Gemma 3 1B	96.1 [94.4,97.7]	87.7 [84.4,91.1]	5.00 [5.00,5.00]	4.88 [4.82,4.93]	0.99 [0.98,1.00]	4.88 [4.83,4.92]
		Llama 3.2 1B	97.7 [96.2,98.9]	90.7 [87.9,93.4]	4.99 [4.98,5.00]	4.84 [4.78,4.89]	0.99 [0.99,1.00]	4.87 [4.83,4.92]
		SmolLM2 1.7B	95.9 [94.3,97.4]	97.9 [96.4,99.0]	5.00 [5.00,5.00]	4.87 [4.81,4.93]	0.99 [0.98,1.00]	4.88 [4.83,4.93]

Table 18: Thought-conditioned ConvFill `Talker` quality on QA datasets using live hosted `Reasoner` thoughts. Entailment is the NLI entailment rate for response fragments against their paired thoughts. Cells report mean with full 95% bootstrap confidence interval $[\mathrm{lo}, \mathrm{hi}]$ over questions; NLI values are percentage points and judge values are on the displayed scales. Bootstrap intervals resample the question unit consistently for fractional NLI scores, Likert means, and thresholded helpfulness. Raw Helpfulness is included for transparency but not used in analyses.

Dataset	Model	Entailment	Non-Contradiction	Coverage	Faithfulness	Helpfulness	Raw Helpfulness
MultiWOZ	SmolLM2 135M	83.9 [82.0,85.9]	85.9 [83.9,87.6]	4.83 [4.79,4.87]	4.57 [4.51,4.61]	0.91 [0.89,0.93]	4.23 [4.16,4.29]
	Gemma 3 270M	82.9 [80.9,84.7]	81.0 [79.1,83.0]	4.85 [4.81,4.89]	4.55 [4.49,4.61]	0.92 [0.91,0.94]	4.27 [4.21,4.33]
	SmolLM2 360M	82.8 [80.9,84.6]	84.5 [82.6,86.4]	4.86 [4.82,4.90]	4.68 [4.63,4.72]	0.94 [0.92,0.95]	4.34 [4.28,4.40]
	Qwen3 0.6B	83.0 [81.2,84.8]	81.8 [79.7,83.6]	4.90 [4.87,4.93]	4.74 [4.69,4.78]	0.94 [0.93,0.96]	4.43 [4.37,4.48]
	Gemma 3 1B	82.3 [80.5,84.2]	81.2 [79.2,83.2]	4.85 [4.81,4.89]	4.52 [4.46,4.58]	0.92 [0.90,0.93]	4.26 [4.20,4.32]
	Llama 3.2 1B	84.5 [82.8,86.3]	81.3 [79.1,83.2]	4.87 [4.83,4.90]	4.60 [4.55,4.65]	0.93 [0.91,0.95]	4.33 [4.28,4.40]
	SmolLM2 1.7B	83.6 [81.9,85.4]	82.8 [80.9,84.7]	4.89 [4.86,4.93]	4.74 [4.70,4.78]	0.95 [0.94,0.96]	4.46 [4.41,4.51]
Everyday Conversations	SmolLM2 135M	92.3 [88.7,95.8]	95.6 [93.1,97.9]	4.99 [4.97,5.00]	4.87 [4.79,4.94]	1.00 [1.00,1.00]	4.83 [4.76,4.90]
	Gemma 3 270M	85.1 [79.6,89.8]	96.5 [93.8,98.6]	4.99 [4.98,5.00]	4.77 [4.66,4.86]	0.99 [0.96,1.00]	4.86 [4.77,4.94]
	SmolLM2 360M	91.4 [87.0,95.4]	97.9 [95.8,99.5]	5.00 [5.00,5.00]	4.91 [4.84,4.97]	1.00 [1.00,1.00]	4.93 [4.87,4.97]
	Qwen3 0.6B	92.0 [88.7,95.8]	96.7 [93.9,98.9]	5.00 [5.00,5.00]	4.88 [4.79,4.95]	1.00 [1.00,1.00]	4.91 [4.85,4.96]
	Gemma 3 1B	84.8 [79.8,90.1]	98.2 [96.2,99.8]	5.00 [5.00,5.00]	4.91 [4.84,4.97]	1.00 [1.00,1.00]	4.96 [4.91,4.99]
	Llama 3.2 1B	87.8 [83.5,92.3]	97.7 [95.9,99.3]	5.00 [5.00,5.00]	4.86 [4.77,4.93]	1.00 [1.00,1.00]	4.89 [4.83,4.95]
	SmolLM2 1.7B	85.1 [80.4,89.9]	99.1 [97.8,100.0]	5.00 [5.00,5.00]	4.97 [4.94,4.99]	1.00 [1.00,1.00]	4.99 [4.97,5.00]

Table 19: Thought-conditioned ConvFill `Talker` quality on sampled multi-turn dialogue turns using dataset-derived conversation thoughts. One non-edge turn is sampled per dialogue for grading; earlier turns in the same dialogue are generated only as hidden warm-up to preserve ConvFill self-history. Metrics and setup are identical to Section 6.1.4, but with more detailed confidence intervals. Cells report mean with full 95% bootstrap confidence interval $[\mathrm{lo}, \mathrm{hi}]$ over sampled turns; NLI values are percentage points and judge values are on the displayed scales. Bootstrap intervals resample the sampled-turn unit consistently for fractional NLI scores, Likert means, and thresholded helpfulness. Raw Helpfulness is included for transparency but not used in analyses.

Trend	Setting	Dataset	Reasoner	QA acc.	Cond. acc.	Ent.	Non-C.	Help.	Coverage	Faith.
All models	Base SLM	SimpleQA	–	**+ BY**	–	–	–	–	–	–
All models	CF dataset	SimpleQA	Data.	**+ BY**	–	**+ BY**	ns	**+ BY**	**+ BY**	**+ BY**
All models	CF live	SimpleQA	Claude Opus 4.7	**+ BY**	**+ BY**	**+ BY**	ns	**+ BY**	**+ BY**	**+ BY**
All models	CF live	SimpleQA	GPT-5.5	**+ BY**	**+ BY**	**+ BY**	ns	**+ BY**	**+ BY**	**+ BY**
All models	CF live	SimpleQA	Gemini 3.1 Pro	**+ BY**	**+ BY**	**+ BY**	ns	**+ BY**	**+ BY**	**+ BY**
All models	Base SLM	LLAMA1	–	**+ BY**	–	–	–	–	–	–
All models	CF dataset	LLAMA1	Data.	**+ BY**	–	**+ BY**	ns	**+ BY**	**+ BY**	**+ BY**
All models	CF live	LLAMA1	Claude Opus 4.7	ns	ns	ns	ns	**+ BY**	ns	**+ BY**
All models	CF live	LLAMA1	GPT-5.5	ns	ns	**+ BY**	ns	**+ BY**	**+ BY**	**+ BY**
All models	CF live	LLAMA1	Gemini 3.1 Pro	ns	ns	ns	ns	**+ BY**	ns	**+ BY**
All models	CF sampled	MultiWOZ	–	–	–	ns	**- BY**	**+ BY**	**+ BY**	**+ BY**
All models	CF sampled	Everyday	–	–	–	ns	ns	ns	ns	ns
SmolLM	Base SLM	SimpleQA	–	**+ BY**	–	–	–	–	–	–
SmolLM	CF dataset	SimpleQA	Data.	**+ BY**	–	**+ BY**	**+ BY**	**+ BY**	**+ BY**	**+ BY**
SmolLM	CF live	SimpleQA	Claude Opus 4.7	**+ BY**	**+ BY**	**+ BY**	**+ BY**	**+ BY**	**+ BY**	**+ BY**
SmolLM	CF live	SimpleQA	GPT-5.5	**+ BY**	**+ BY**	**+ BY**	**+ BY**	**+ BY**	**+ BY**	**+ BY**
SmolLM	CF live	SimpleQA	Gemini 3.1 Pro	**+ BY**	**+ BY**	**+ BY**	**+ BY**	**+ BY**	**+ BY**	**+ BY**
SmolLM	Base SLM	LLAMA1	–	**+ BY**	–	–	–	–	–	–
SmolLM	CF dataset	LLAMA1	Data.	**+ BY**	–	**+ BY**	**+ BY**	**+ BY**	**+ BY**	**+ BY**
SmolLM	CF live	LLAMA1	Claude Opus 4.7	ns	ns	ns	**+ BY**	ns	ns	**+ BY**
SmolLM	CF live	LLAMA1	GPT-5.5	ns	ns	ns	**+ BY**	ns	**+ BY**	**+ BY**
SmolLM	CF live	LLAMA1	Gemini 3.1 Pro	ns	ns	ns	ns	ns	ns	ns
SmolLM	CF sampled	MultiWOZ	–	–	–	ns	**- BY**	**+ BY**	**+ BY**	**+ BY**
SmolLM	CF sampled	Everyday	–	–	–	ns	ns	ns	ns	ns
Gemma	Base SLM	SimpleQA	–	ns	–	–	–	–	–	–
Gemma	CF dataset	SimpleQA	Data.	**+ BY**	–	**+ BY**	**- BY**	**+ BY**	**+ BY**	**+ BY**
Gemma	CF live	SimpleQA	Claude Opus 4.7	ns	ns	**+ BY**	**- BY**	ns	ns	**+ BY**
Gemma	CF live	SimpleQA	GPT-5.5	**+ BY**	**+ BY**	**+ BY**	**- BY**	**+ BY**	**+ BY**	**+ BY**
Gemma	CF live	SimpleQA	Gemini 3.1 Pro	**+ BY**	**+ BY**	**+ BY**	**- BY**	**+ BY**	**+ BY**	**+ BY**
Gemma	Base SLM	LLAMA1	–	**+ BY**	–	–	–	–	–	–
Gemma	CF dataset	LLAMA1	Data.	**+ BY**	–	**+ BY**	**- BY**	**+ BY**	**+ BY**	**+ BY**
Gemma	CF live	LLAMA1	Claude Opus 4.7	**+ BY**	**+ BY**	ns	**- BY**	**+ BY**	ns	**+ BY**
Gemma	CF live	LLAMA1	GPT-5.5	ns	ns	**+ BY**	**- BY**	**+ BY**	ns	**+ BY**
Gemma	CF live	LLAMA1	Gemini 3.1 Pro	ns	ns	ns	**- BY**	ns	ns	**+ BY**
Gemma	CF sampled	MultiWOZ	–	–	–	ns	ns	ns	ns	ns
Gemma	CF sampled	Everyday	–	–	–	ns	ns	ns	ns	ns

Table 20: Scaling-test comparison using conservative Benjamini–Yekutieli FDR correction (Benjamini and Yekutieli, 2001). Cells report trend direction only when BY-significant; ns = not BY-significant, and – = not applicable. *All models* denotes an exploratory parameter trend across all seven `Talker` models and is shown first for orientation; SmolLM is the primary within-family trend, and Gemma is a two-point 270M-to-1B contrast. CF = ConvFill; Data. = dataset/reference thoughts; QA acc. = Accuracy; Cond. acc. = Conditional Accuracy; Ent. = Entailment; Non-C. = Non-Contradiction; Help. = Helpfulness; Faith. = Faithfulness.

F Inference and Evaluation Prompts

This section of the appendix contains the exact per-task prompts used during live interaction and the LLM-as-a-Judge prompt used for single- and multi-turn evaluation.

F.1 LLM-as-a-Judge Prompt

Prompt F.1: LLM-as-a-Judge Prompt

```
You are evaluating ONE turn of a ConvFill conversation, a real-time two-model
system. A reasoner LLM produces a stream of factual "thoughts" (bullet-like
knowledge), and a small talker model weaves those thoughts into a fluent
spoken response. Your job is to rate this **single turn** across three rubric
dimensions.

# Rubric

Single-turn evaluation. All metrics on a 1-5 Likert scale, 5 = best.

You are given exactly one user query, the bullet-list of thoughts that were
streamed in for that turn, and the conversational response the system produced. Word-for-word
    match is not required -- what matters is whether
the listener leaves this turn with the right understanding and can act on
it.

**Use the thoughts as the only factual source.** Do not penalize the response
for failing to answer parts of the user query that are not answered by the
thoughts. The task is to judge how well the response conveys the available
thoughts, not whether the reasoner thoughts themselves were complete.

Keep the three dimensions separate:

- Coverage is about whether the response includes the relevant thought
  content. Ignore whether that content is phrased awkwardly or attached to
  the wrong relation; score those problems under Faithfulness instead.
  The denominator is the thoughts, not every part of the user question: if
  the user asks two things but the thoughts answer only one, Coverage asks
  whether the response conveyed that one available answer.
- Faithfulness is about whether the response changes, contradicts, or adds
  unsupported meaning beyond the thoughts.
- Helpfulness is about whether a normal user would consider this response
  aligned and structurally consistent with their query, independent of
  the other two dimensions.

## 1. Coverage

*Does the response convey the relevant information expressed in the thoughts?*

A fact in the thoughts is "load-bearing" if it is relevant to the user query
or is the main available answer. Score on whether that available information
is present in the response, not whether every part of the user's query was
answerable from the thoughts.
Paraphrase, reordering, compression, unit/format conversion, and casual
wording are fine.

This metric is about errors of *omission* only. If the response includes the
right value or entity but puts it in a clumsy or partly wrong frame, give
Coverage credit for including it and use Faithfulness to penalize the
distortion. Do not lower Coverage merely because the response adds filler,
extra unsupported phrasing, or conversational scaffolding.

When the thoughts themselves are incomplete, do **not** lower Coverage for
missing facts that are absent from the thoughts. If the thoughts answer only
one part of a multi-part user query, Coverage asks whether the response
conveys that available part.
```

Prompt F.2: LLM-as-a-Judge Prompt contd. (extends F.1)

```
If the key answer value, name, number, or entity from the thoughts appears in
the response, give substantial Coverage credit even when the response attaches
it to the wrong relation or frames it awkwardly. Relation errors belong under
Faithfulness. For short symbolic answers, treat capitalization and spacing
differences as surface form for Coverage.

- **5** -- Every relevant thought fact is conveyed in the response.
- **4** -- All relevant thought facts are conveyed; only minor,
  non-load-bearing details from the thoughts are omitted.
- **3** -- Most relevant thought facts are present, but one material available
  fact is missing.
- **2** -- Multiple relevant thought facts are missing; the response materially
  under-informs the listener relative to the available thoughts.
- **1** -- Most of the available thought content is absent from the response.

## 2. Faithfulness

*Does the response avoid contradicting or fabricating beyond the thoughts?*

This metric is about errors of *commission* -- distortions, fabrications,
unsupported claims, misleading action claims, or framings that change a fact's
meaning. A response that merely omits information can still score high here,
but a response that says it is looking up, pulling up, checking, or already
has details while giving no answer is not a clean omission; it is a misleading
non-answer. Severity is weighted by how much the distortion affects the
available answer: a fabrication on the main answer is worse than one in a
tangential aside.

Be forgiving about surface form. Do not penalize capitalization, punctuation,
spacing, or harmless formatting differences when the intended answer is clear.
For times, accept an unambiguous 12-hour rendering of a 24-hour time when the
conversation context makes the meaning clear. Do not treat missing AM/PM as a
contradiction when the surrounding user query or response makes the time of
day clear enough.
Do not penalize unfounded but non-factual connective language unless it asserts
or implies an unsupported fact.

For short names, codes, and style symbols, do not over-penalize
capitalization or spacing differences when the same letters are present and a
normal listener/reader would identify the intended answer.

Process filler such as "let me look that up," "I found the details," or "I
have it right here" is low-value conversational scaffolding. If the response
also gives the same relayed answer and the filler does not change the
factual meaning, do not penalize the filler; the Faithfulness score should
usually remain **5**. If the response is mostly such process filler and never
actually gives the available answer, Faithfulness should usually be around
**3** rather than 5: it did not contradict the thoughts, but it created a
misleading impression of progress while withholding the actual answer.

- **5** -- Nothing in the response contradicts the thoughts or introduces
  unsupported claims; unambiguously no distortions of meaning.
- **4** -- Overall faithful. Small differences arise from paraphrasing or
  conversational rephrasing, but they are at worst inconsequential technicalities.
- **3** -- There is at most one minor inaccuracy, distortion, or unsupported claim,
  but this would not significantly cause incorrect ideas.
- **2** -- A clear inaccuracy, fabrication, or distortion that would lead the
  listener to a wrong conclusion on some part of the statement.
- **1** -- Multiple contradictions or fabrications; the listener leaves with
  a substantively wrong understanding on key parts of the response.
```

Prompt F.3: LLM-as-a-Judge Prompt contd. (extends F.2)

```
## 3. Helpfulness

*Given the user's query, is the response structured in an appropriate fashion?*

Scores **4-5** indicate generally appropriate responses to the question,
statement, or phrase. Even if wrong, it is still syntactically an appropriate
response. Scores **1-2** count as "No, not valid or appropriate response." Use
"3" for only truly borderline cases that are still on the side of acceptable
and appropriate grammatically and given the conversation context.

A response can be helpful even if it is brief or slightly awkward. A response
is not helpful when it withholds the available answer, answers the wrong
question, gives only vague setup/filler, or states the right words in a way
that does not follow and connect to what the user said.

Factual correctness is already captured by Coverage and Faithfulness.

If the user asks "What is the day after Monday," A **5** could be answers
like "The day after Monday is Tuesday" or "The day after Monday is Wednesday"
since both are structurally appropriate answers, even though the listed day
itself isn't exactly correct. "Tuesday is a day of the week" or "Ah, Tuesday is
something everyone always says" would be a **2**, since it does not answer the
user's question. The fact that the **answer** is contained does not matter
for this purpose.

- **5** -- Appropriately answers the user. It fully fits within the structure
  of a normal and appropriate conversation. The most expected type of response.
- **4** -- Answers the query but in a form that sounds possibly odd.
- **3** -- Still structurally useful and reasonable based on what was said, but
  only barely.
- **2** -- Vague or tangential; does not meaningfully respond to the user's query.
- **1** -- Off-topic, evasive, or actively misaligned given what the user asked.

# Output format

Return ONLY a single JSON object, no markdown, no commentary outside the JSON.
The object must have exactly this shape:

```
{
  "coverage":     {"score": <int 1-5>, "rationale": "<1-2 sentences>"},
  "faithfulness": {"score": <int 1-5>, "rationale": "<1-2 sentences>"},
  "helpfulness":  {"score": <int 1-5>, "rationale": "<1-2 sentences>"}
}
```

Each `score` is an integer from 1 to 5 (use the anchors above). Each
`rationale` is one or two sentences citing the specific behavior that
drove the score -- name a fact, bullet, or phrase wherever possible.

Turn

user
{{ user_turn }}

thoughts
{% for b in thoughts_substance %}- {{ b }}
{% endfor %}
conversational response
{{ convfill_response }}
```

Prompt F.4: Normal Task Backend Prompt

```
Your job is to take the previous turns of the conversation and respond with concise sentences,
    each representing one thought or phrase that can be said to the user. The user's queries were
    spoken out loud and transcribed via a speech-to-text (STT) model.
Your job is not to hold a conversation or show emotion or connection, just to act as an
    information supplier and answer whatever the user asks to move the conversation forward.
ONLY output concise sentences, nothing else. No em dashes. No examples. No recommendations unless
    asked for.
You must put all numbers and symbols in words.
Anything that must be relayed verbatim MUST be put in quotes. Max 3 sentences. Ask AT MOST ONE
    question.

Example Conversation and Response:
User: Hey there, I want to know how to make some cafe drinks at home to save money.
You: Making drinks at home is not too hard, don't worry. What kind of drinks do you like?
User: Hmm, I like matcha and non-coffee drinks. I'm glad they are not too hard to make at home. I
    really like flavored matcha drinks, like vanilla or strawberry. Are those easy to make as well
    ?
You: The difficulty depends on how much you want to make homemade. You can just buy a Torani
    vanilla bean syrup for the vanilla one. You can also make a homemade strawberry puree for the
    strawberry matcha.

OR

You: You can just buy a Torani vanilla bean syrup for the vanilla. You can make a homemade
    strawberry puree for the strawberry matcha. What are you looking for in terms of effort?

OR

You: What are you looking for in terms of effort? You can just buy a Torani vanilla bean syrup for
     the vanilla. You can also make a homemade strawberry puree for the strawberry matcha.

Here is the conversation:
{{ conversation }}
```

Prompt F.5: RAG Task Backend Prompt

```
Your job is to take the previous turns of the conversation and respond with concise sentences,
    each representing one thought or phrase that can be said to the user. The user's queries were
    spoken out loud and transcribed via a speech-to-text (STT) model.
Your job is not to hold a conversation or show emotion or connection, just to act as an
    information supplier and answer whatever the user asks to move the conversation forward.
ONLY output concise sentences, nothing else. No em dashes. No examples. No recommendations unless
    asked for.
You must put all numbers and symbols in words for dollar amounts and large numbers.
Anything that must be relayed verbatim MUST be put in quotes. Max 3 sentences. DO NOT ASK ANY
    QUESTIONS.

Example Conversation and Response:
User: Hey there, I want to know how to make some cafe drinks at home to save money.
You: Making drinks at home is not too hard, don't worry. What kind of drinks do you like?
User: Hmm, I like matcha and non-coffee drinks. I'm glad they are not too hard to make at home. I
    really like flavored matcha drinks, like vanilla or strawberry. Are those easy to make as well
    ?
You: The difficulty depends on how much you want to make homemade. You can just buy a Torani
    vanilla bean syrup for the vanilla one. You can also make a homemade strawberry puree for the
    strawberry matcha.

OR

You: You can just buy a Torani vanilla bean syrup for the vanilla. You can make a homemade
    strawberry puree for the strawberry matcha. What are you looking for in terms of effort?

OR

You: What are you looking for in terms of effort? You can just buy a Torani vanilla bean syrup for
     the vanilla. You can also make a homemade strawberry puree for the strawberry matcha.

Here is the context for your response. To help you respond to queries, here are some potentially
    relevant excerpts from the University of Washington (UW) Computer Science PhD Student Handbook
    :
{{ rag_context }}

Here is the conversation:
{{ conversation }}
```

Prompt F.6: MCP Task Backend Prompt

```
Your job is to take the previous turns of the conversation and respond with concise sentences,
    each representing one thought or phrase that can be said to the user. The user's queries were
    spoken out loud and transcribed via a speech-to-text (STT) model.
Your job is not to hold a conversation or show emotion or connection, just to act as an
    information supplier and answer whatever the user asks to move the conversation forward.
ONLY output concise sentences, nothing else. No em dashes. No examples. No recommendations unless
    asked for.
You must put all numbers and symbols in words for dollar amounts and large numbers.
Anything that must be relayed verbatim MUST be put in quotes. Max 3 sentences. Ask at most one
    question.

Use the provided tools when you need to look up information. Focus on likely keywords instead of
    long verbatim phrases for searches (e.g., "ticket" not "the concert ticket for the show
    tonight").

Example Conversation and Response:
User: Hey there, I want to know how to make some cafe drinks at home to save money.
You: Making drinks at home is not too hard, don't worry. What kind of drinks do you like?
User: Hmm, I like matcha and non-coffee drinks. I'm glad they are not too hard to make at home. I
    really like flavored matcha drinks, like vanilla or strawberry. Are those easy to make as well
    ?
You: The difficulty depends on how much you want to make homemade. You can just buy a Torani
    vanilla bean syrup for the vanilla one. You can also make a homemade strawberry puree for the
    strawberry matcha.

OR

You: You can just buy a Torani vanilla bean syrup for the vanilla. You can make a homemade
    strawberry puree for the strawberry matcha. What are you looking for in terms of effort?

OR

You: What are you looking for in terms of effort? You can just buy a Torani vanilla bean syrup for
     the vanilla. You can also make a homemade strawberry puree for the strawberry matcha.

Here is the conversation:
{{ conversation }}
```