

# Task-Level Instructions Induction for Audio Question Answering from Few Examples

Po-Chun Chen<sup>1</sup> Hen-Hsen Huang<sup>2</sup> Hsin-Hsi Chen<sup>1,3</sup>

<sup>1</sup>Department of Computer Science and Information Engineering,  
National Taiwan University, Taiwan

<sup>2</sup>Institute of Information Science, Academia Sinica, Taiwan

<sup>3</sup>AI Research Center (AINTU), National Taiwan University, Taiwan

pcchen@nlg.csie.ntu.edu.tw,

hhuang@iis.sinica.edu.tw, hhchen@ntu.edu.tw

## Abstract

Large audio-language models (LALMs) benefit from Chain-of-Thought (CoT) prompting for audio question answering (AQA), but acquiring audio CoT examples is particularly challenging as it requires sequential listening and careful integration of acoustic and linguistic information. Surprisingly, our experiments reveal that standard few-shot prompting yields inconsistent results compared to zero-shot CoT, with several models showing degraded accuracy. Moreover, few-shot prompting incurs substantially higher inference costs by processing multiple audio demonstrations per inference. We propose Audio-Induct, which induces reusable textual task instructions from few audio examples once per task, requiring no additional demonstrations at inference. Evaluated on 9 LALMs across two benchmarks, Audio-Induct outperforms state-of-the-art prompting methods while maintaining low inference costs. Inducted Task Instructions transfer effectively across models, enabling scalable deployment.

## 1 Introduction

Recent advances in large audio-language models (LALMs) have enabled sophisticated audio understanding across diverse tasks (Yang et al., 2025), including audio question answering (AQA). Similar to text-only models, LALMs benefit from Chain-of-Thought (CoT) prompting (Ma et al., 2025a), where providing demonstrations with explicit reasoning steps improves performance. However, audio presents unique challenges for CoT annotation: it requires sequential listening and careful integration of acoustic and linguistic information (Ma et al., 2025b), making high-quality annotation substantially more difficult than text or images. Consequently, audio CoT datasets remain scarce.

While audio CoT demonstrations are difficult to acquire, few-shot prompting (Brown et al., 2020) without explicit reasoning annotations could still be

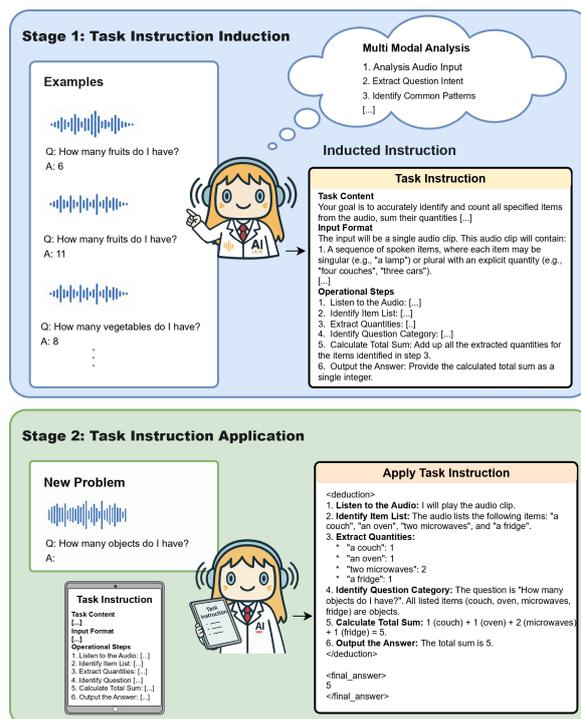


Figure 1: Overview of the Audio-Induct framework. Stage 1 analyzes few AQA examples to induce a reusable task instruction once per task. Stage 2 applies the instruction to new instances without requiring audio demonstrations at inference.

beneficial. However, our experiments reveal unexpected challenges when applying this approach to audio reasoning. First, few-shot prompting yields inconsistent results, with several models showing degraded accuracy compared to zero-shot CoT (Kojima et al., 2022) rather than the expected improvements. Second, few-shot prompting incurs substantially higher inference costs by processing multiple audio demonstrations per inference, where a 30-second audio sample costs equivalent to 20,000 text tokens in LALM APIs such as Voxtral Small.

Motivated by the success of instruction induction in text domains (Honovich et al., 2023; Chen et al., 2024b), we propose Audio-Induct, a frame-

work for audio reasoning that generates reusable task-level instructions from few audio examples. Given few AQA examples, Audio-Induct induces a purely textual instruction capturing the reasoning procedure, which can then be applied to new instances using only their audio input, requiring no demonstration audio at inference time. Across nine LALMs on two audio reasoning benchmarks, Audio-Induct outperforms zero-shot CoT (Kojima et al., 2022), few-shot prompting, and Strategic Chain-of-Thought (Wang et al., 2024) while maintaining low inference costs. Furthermore, instructions induced by stronger models transfer effectively to smaller models, enabling cost-efficient deployment at scale.

In summary, our contributions are: (1) the first instruction induction framework for audio reasoning, (2) comprehensive evaluation showing superior performance over existing prompting methods, and (3) demonstration of effective cross-model instruction transfer for cost-efficient deployment.

## 2 Method

Audio-Induct is a two-stage approach that induces reusable task instructions from few AQA examples, then applies them to new instances without requiring demonstrations at inference.

### 2.1 Task Instruction Induction

This stage analyzes a few AQA examples with ground-truth answers to induce a textual instruction capturing the task’s reasoning procedure. We define the example set as:

$$\mathcal{E} = \{(a_1, q_1, y_1), \dots, (a_N, q_N, y_N)\},$$

where  $N$  is the number of examples,  $a_i$  is the audio,  $q_i$  is the question, and  $y_i$  is the ground-truth answer.

We use an instruction induction meta-prompt  $P_M$  to guide the LALM through a two-step reasoning process within a single generation. The model first produces a multi-modal analysis  $m$  examining patterns across the audio and textual components. This analysis then guides the induction of the task instruction  $P_T$  with structured operational steps:

$$(m, P_T) = \text{LALM}(P_M, \mathcal{E})$$

The resulting  $P_T$  is a reusable, purely textual instruction applicable to all instances within the same task type. Importantly, this induction process occurs only once per task, amortizing the cost across all subsequent inferences. The complete meta-prompt template is provided in Appendix G.

### 2.2 Task Instruction Application

At inference time, the induced instruction  $P_T$  is applied to solve new instances from the same task. Given a new instance with audio  $a^*$  and question  $q^*$ , the LALM produces an answer:

$$\hat{y} = \text{LALM}(P_T, a^*, q^*)$$

where  $\hat{y}$  is the model’s prediction. Critically, this inference requires only the target audio  $a^*$  and question  $q^*$ , without any demonstration audio, substantially reducing computational costs compared to few-shot prompting.

## 3 Experiment Setup

### 3.1 Tasks

We evaluate Audio-Induct on Big Bench Audio (BBA) and Multi-Modal Audio Reasoning (MMAR). BBA is an audio adaptation of Big Bench Hard (BBH) (Suzgun et al., 2022), featuring four reasoning tasks selected by ArtificialAnalysis for AQA (Formal Fallacies, Navigate, Object Counting, and Web of Lies). MMAR (Ma et al., 2025b) is a comprehensive benchmark evaluating models’ ability to integrate acoustic and linguistic information for complex reasoning.

### 3.2 Models

We evaluate 9 LALMs from two families, including Voxtral (Mini, Small) and Gemini (2.0 Flash Lite, 2.0 Flash, 2.5 Flash Lite, 2.5 Flash, 2.5 Pro). Voxtral models are fully open-weights LALMs with weights available on Hugging Face. For Gemini 2.5 models, we evaluate both with and without Thinking mode enabled. Detailed configurations are provided in Appendix C.

### 3.3 Baselines

We compare against three prompting strategies:

**Z-CoT** Zero-shot Chain-of-Thought (Kojima et al., 2022) elicits step-by-step reasoning by appending “Let’s think step by step.”

**3-Shot** Standard few-shot prompting (Brown et al., 2020) with three audio-question-answer demonstrations without explicit reasoning steps.

**S-CoT** Automatic Strategic Chain-of-Thought (Wang et al., 2024) generates a problem-solving strategy before reasoning.

Model	BBA					MMAR				
	Z-CoT	3-Shot	S-CoT	Our	$\delta$	Z-CoT	3-Shot	S-CoT	Our	$\delta$
Voxtral Mini	58.13	52.92	<b>61.35</b>	56.15	-1.98	46.87	<b>47.55</b>	45.08	46.19	-0.68
Voxtral Small	85.42	83.23	87.29	<b>87.40</b>	+1.98	55.16	<b>56.35</b>	53.64	56.01	+0.85
Gemini 2.0 Flash Lite	77.60	67.29	<b>82.60</b>	76.15	-1.46	59.73	58.38	60.07	<b>60.91</b>	+1.18
Gemini 2.0 Flash	83.85	54.79	85.52	<b>91.35</b>	+7.50	63.62	<b>66.67</b>	63.62	63.45	-0.17
Gemini 2.5 Flash Lite	90.31	85.21	94.58	<b>95.31</b>	+5.00	56.01	51.44	56.18	<b>57.53</b>	+1.52
Gemini 2.5 Flash Lite (Thinking)	95.10	82.71	96.04	<b>97.40</b>	+2.29	56.18	50.25	<b>57.87</b>	56.18	0.00
Gemini 2.5 Flash	91.35	94.58	95.21	<b>98.33</b>	+6.98	67.34	68.02	68.02	<b>69.88</b>	+2.54
Gemini 2.5 Flash (Thinking)	96.04	93.12	95.62	<b>98.33</b>	+2.29	69.20	69.37	70.39	<b>70.90</b>	+1.69
Gemini 2.5 Pro (Thinking)	99.48	62.08	99.48	<b>99.58</b>	+0.10	80.37	79.36	79.86	<b>80.88</b>	+0.51

Table 1: Performance (%) of different models on BBA and MMAR under different prompting methods.  $\delta$  denotes Our – Z-CoT. **Bold** indicates best performance, **blue** indicates improvement, **red** indicates decline.

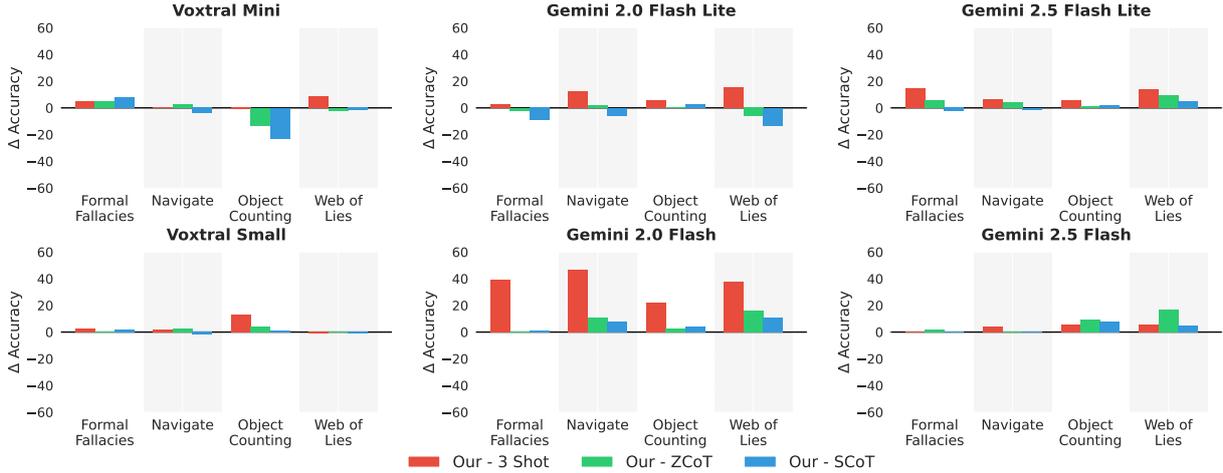


Figure 2: Head-to-head performance comparison within each model on four BBA subtasks. The accuracy delta indicates the performance difference between Audio-Induct and three baselines (Z-CoT, 3-Shot, S-CoT).

### 3.4 Implementation Details

We use  $N = 3$  examples for instruction induction across all tasks. All experiments use official API endpoints with temperature set to 0 and top-p set to 1.0 for deterministic outputs. Additional N-shot analysis is in Appendix E.1.

## 4 Results Analysis

### 4.1 Audio Instruction Induction Effectiveness

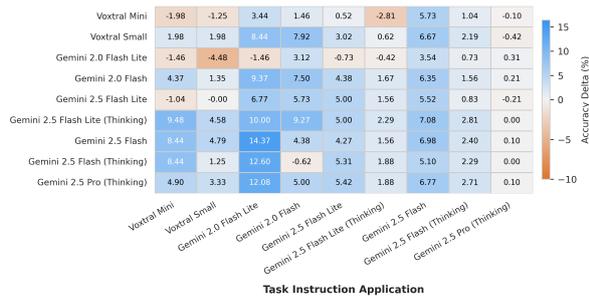
We evaluate Audio-Induct across 9 models on both benchmarks, with comprehensive results presented in Table 1. Audio-Induct outperforms all three baselines, achieving win-tie-loss rates of 13-1-4 against Z-CoT, 15-0-3 against 3-Shot prompting, and 14-0-4 against S-CoT.

The effectiveness varies with model scale and capability. Larger models (e.g., Gemini 2.5 Flash, Gemini 2.5 Pro) demonstrate stronger instruction induction performance, while smaller models show inconsistent results, likely due to limited instruction induction capabilities. These results align with findings in text-only instruction induction (Hon-

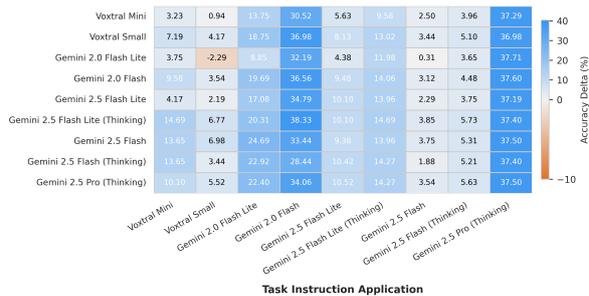
ovich et al., 2023; Chen et al., 2024b), where larger models exhibit stronger capabilities, suggesting this pattern extends to audio reasoning tasks.

Notably, standard 3-Shot prompting performs below expectations compared to Z-CoT across both benchmarks, underperforming on 8 out of 9 models on BBA with mixed results on MMAR. We speculate that in AQA, demonstrations with direct answers, rather than explicit reasoning steps, may interfere with the models’ step-by-step reasoning capabilities, leading to performance degradation.

We conducted Wilcoxon Signed-Rank Tests (Wilcoxon, 1945) across 18 model-benchmark combinations ( $9 \text{ models} \times 2 \text{ benchmarks}$ ). Audio-Induct shows statistically significant improvements over Z-CoT ( $p = 0.006$ ) and 3-Shot ( $p < 0.001$ ), with marginally significant improvement over S-CoT ( $p = 0.054$ ). Detailed inference cost analysis is provided in Appendix F.



(a) Our - Z-CoT



(b) Our - 3-Shot

Figure 3: Cross-model instruction transfer on BBA. Each cell shows the accuracy delta when instructions induced by one model (y-axis) are applied to another model (x-axis), compared to (a) Z-CoT and (b) 3-Shot prompting. Blue indicates improvement, orange indicates decline.

## 4.2 Subtask-Level Analysis on BBA

Figure 2 shows performance differences across BBA subtasks for three model pairs. Consistent with our earlier observation, stronger models demonstrate more effective instruction induction. Within each model family, smaller variants in the top row (Voxtral Mini, Gemini 2.0 Flash Lite, Gemini 2.5 Flash Lite) show inconsistent results across tasks with mixed gains and losses. In contrast, their larger counterparts in the bottom row (Voxtral Small, Gemini 2.0 Flash, Gemini 2.5 Flash) demonstrate more consistent improvements across most tasks, with Gemini 2.0 Flash showing particularly strong gains across all four reasoning tasks. Complete results for all models are in Appendix E.2.

## 4.3 Cross-Model Instruction Transfer

To evaluate whether induced task instructions transfer effectively across different LALMs, we conduct cross-model experiments on BBA. We induce task instructions with one model and apply them to another model for inference. Figure 3 shows the results compared to Z-CoT and 3-Shot baselines.

The results show that instruction transfer vastly outperforms 3-Shot prompting across nearly all model combinations, and also consistently improves over Z-CoT. Critically, instruction quality depends on source model capability. Instructions from stronger models (e.g., Gemini 2.5 Pro, Gemini 2.5 Flash) consistently improve all inference models regardless of architecture, while instructions from weaker models (e.g., Voxtral Mini, Gemini 2.0 Flash Lite) show inconsistent results, sometimes even degrading performance compared to Z-CoT. This pattern aligns with Chen et al. (2024b)’s findings in text-only settings. This enables cost-efficient deployment by inducing instructions once

with capable models, then applying them across cheaper models for large-scale inference. Detailed cross-model transfer results across all model combinations, including comparisons with S-CoT, are provided in Appendix Figure 5c.

## 4.4 Sensitivity to Example Selection

We investigate how sensitive the induced instructions are to the choice and order of examples. We conducted experiments on BBA using two representative models (Gemini 2.5 Flash and Voxtral Small) with four variants: (1) original order, (2) reversed order, (3) random selection 1, and (4) random selection 2, where random selections independently sample 3 examples from the training set.

Variant	Gemini 2.5 Flash		Voxtral Small	
	3-Shot	Our	3-Shot	Our
Original	94.58	<b>98.33</b>	83.23	<b>87.40</b>
Random 1	90.73	<b>98.02</b>	83.75	<b>86.46</b>
Random 2	92.71	<b>97.40</b>	83.85	<b>87.60</b>
Reversed	91.35	<b>97.19</b>	83.85	<b>87.29</b>
Std. Dev.	1.48	0.46	0.26	0.43

Table 2: Sensitivity analysis on BBA (%). Audio-Induct consistently outperforms 3-Shot across all variants with lower variance.

As shown in Table 2, Audio-Induct consistently outperforms 3-Shot across all tested variants, with performance advantages of 3.75-7.29% on Gemini 2.5 Flash and 2.71-4.17% on Voxtral Small. Notably, Audio-Induct exhibits consistent variance ( $\sim 0.4\%$ ) across both model families, while 3-Shot’s sensitivity varies significantly by architecture (1.48% for Gemini vs. 0.26% for Voxtral). This suggests that Audio-Induct is not only more effective but also more robust to the specific choice of examples.

### Induced Instruction: BBA Web of Lies

#### Task Content

Listen to an audio clip presenting a series of statements about individuals' truthfulness or deceit, forming a logical chain. The audio concludes with a question about whether a specific individual tells the truth. Deduce the truthfulness of the final subject and answer "Yes" or "No".

#### Input Format

A single audio file containing: (1) An initial statement about a person's truthfulness (e.g., "William lies"). (2) Subsequent statements attributing truthfulness or deceit to previous persons (e.g., "Inga says William tells the truth"). (3) A final question (e.g., "Does Crista tell the truth?").

#### Operational Steps

1. *Transcribe Audio*: Obtain a textual transcript of all spoken content.
2. *Identify Initial Premise*: Extract the first statement establishing truthfulness. Assign True/False.
3. *Parse Chained Statements*: For each statement, identify the speaker and their assertion about the previous person.
4. *Deduce Truth Values*: If B says "A tells the truth," B's value equals A's. If B says "A lies," B's value is opposite.
5. *Identify Final Question*: Locate "Does [Person] tell the truth?"
6. *Formulate Answer*: Output "Yes" if True, "No" if False.

Table 3: Induced task instruction for BBA Web of Lies, generated by Gemini 2.5 Flash. The complete output including multi-modal analysis and an additional example for MMAR music analysis are provided in Appendix G.

## 4.5 Case Study: Induced Instruction

To illustrate how Audio-Induct generates task-specific instructions, Table 3 presents an induced instruction for the BBA Web of Lies task, generated by Gemini 2.5 Flash from three audio examples.

The key insight is that Audio-Induct **makes implicit reasoning mechanisms explicit**. While Z-CoT merely prompts "think step by step" without specifying *how* to reason, and 3-Shot provides examples, leaving the model to infer patterns, the induced instruction explicitly captures task-specific reasoning procedures. For instance, Step 4 clearly articulates the truth value propagation rule—if B says "A lies," then B's value is opposite to A's—providing unambiguous guidance that baseline approaches lack. Notably, this structured procedure is automatically induced from just three examples through multi-modal analysis, not manually crafted. This illustrates how explicit procedural guidance may benefit reasoning-intensive tasks.

## 5 Related Work

### 5.1 Instruction Induction

Instruction induction enables LLMs to generate task-level instructions from few examples (Honovich et al., 2023). Related work includes automatic prompt optimization (Zhou et al., 2022; Chen et al., 2024a; Sun et al., 2023) requiring multiple iterations, self-composed reasoning structures (Zhou et al., 2024), and self-reasoning frameworks (Aswani et al., 2024). Chen et al. (2024b) proposed Induct-Learn with single-inference generation. However, existing approaches remain limited to text-only tasks.

### 5.2 Audio Reasoning Methods

Chain-of-Thought prompting has proven effective for reasoning (Kojima et al., 2022; Wei et al., 2022). Recent work explored audio CoT (Ma et al., 2025a), but few-shot prompting with audio demonstrations incurs substantially higher inference costs. Existing methods primarily rely on training: Audio-Reasoner (Xie et al., 2025) and Audio Flamingo (Kong et al., 2024) require fine-tuning on large-scale datasets, while TwS (Xiong et al., 2025) and AudioGenie-Reasoner employ additional components.

We apply instruction induction to audio reasoning, inducing reusable task instructions from few examples without model fine-tuning, and requiring no audio demonstrations at inference.

## 6 Conclusion

We presented Audio-Induct, the first instruction induction framework for audio reasoning. By inducing reusable textual instructions from few audio examples once per task, our approach requires no audio demonstrations at inference, maintaining low inference costs while outperforming state-of-the-art prompting methods. Instructions from stronger models effectively transfer to smaller models, enabling cost-efficient deployment at scale.

### Limitations

Our study has several limitations that present opportunities for future work. First, our evaluation covers both commercial API-based models (Gemini series) and open-weights models (Voxtral). While this balances state-of-the-art performance with reproducibility, the commercial models have undisclosed architectures that may limit detailed analysis of model-specific behaviors.

Second, our approach relies on models with strong instruction-following capabilities to generate responses in structured formats and induce effective task instructions. Modern LALMs are trained to handle complex instructions and produce structured outputs. However, pure audio-only models without language model components may lack the necessary capabilities to follow structured format requirements, potentially limiting the applicability of our framework to such architectures.

Third, our framework operates at the task level with predefined task categories as in BBA and MMAR. In scenarios where task boundaries are unclear or categories are not predefined, additional mechanisms such as query clustering or domain taxonomy adaptation would be needed.

## Acknowledgements

This work was supported by National Science and Technology Council, Taiwan, under grants NSTC 114-2221-E-002 -070 -MY3 and NSTC 113-2634-F-002-003 -, and by Ministry of Education (MOE), Taiwan, under grant NTU-114L900901.

## References

- Krishna Aswani, Huilin Lu, Pranav Patankar, Priya Dhalwani, Xue Tan, Jayant Ganeshmohan, and Simon Lacasse. 2024. [Auto-evolve: Enhancing large language model’s performance via self-reasoning framework](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13243–13257, Miami, Florida, USA. Association for Computational Linguistics.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, and 12 others. 2020. [Language models are few-shot learners](#). *ArXiv*, abs/2005.14165.
- Lichang Chen, Jiu-hai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. 2024a. [InstructZero: Efficient instruction optimization for black-box large language models](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 6503–6518. PMLR.
- Po-Chun Chen, Sheng-Lun Wei, Hen-Hsen Huang, and Hsin-Hsi Chen. 2024b. [Induct-learn: Short phrase prompting with instruction induction](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5204–5231, Miami, Florida, USA. Association for Computational Linguistics.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3278 others. 2025. [Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities](#). *ArXiv*, abs/2507.06261.
- Google. 2025. [Gemini 2.0: Flash, flash-lite and pro](#). FEB. 5, 2025.
- Or Honovich, Uri Shoham, Samuel R. Bowman, and Omer Levy. 2023. [Instruction induction: From few examples to natural language task descriptions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1935–1952, Toronto, Canada. Association for Computational Linguistics.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. [Large language models are zero-shot reasoners](#). *ArXiv*, abs/2205.11916.
- Zhifeng Kong, Arushi Goel, Rohan Badlani, Wei Ping, Rafael Valle, and Bryan Catanzaro. 2024. [Audio flamingo: A novel audio language model with few-shot learning and dialogue abilities](#). *ArXiv*, abs/2402.01831.
- Alexander H. Liu, Andrew S. C. Ehrenberg, Andy Lo, Clément Denoix, Corentin Barreau, Guillaume Lample, Jean-Malo Delignon, Khyathi Raghavi Chandu, Patrick von Platen, Pavankumar Reddy Muddireddy, Sanchit Gandhi, Soham Ghosh, Srijan Mishra, Thomas Foubert, Abhinav Rastogi, Adam Yang, Albert Q. Jiang, Alexandre Sablayrolles, Am’elie H’eliou, and 87 others. 2025. [Voxtral](#).
- Ziyang Ma, Zhuo Chen, Yuping Wang, Eng Siong Chng, and Xie Chen. 2025a. [Audio-cot: Exploring chain-of-thought reasoning in large audio language model](#). *ArXiv*, abs/2501.07246.
- Ziyang Ma, Yi Ma, Yanqiao Zhu, Chen Yang, Yi-Wen Chao, Ruiyang Xu, Wenxi Chen, Yuanzhe Chen, Zhuo Chen, Jian Cong, Kai Li, Keliang Li, Siyou Li, Xinfeng Li, Xiquan Li, Zheng Lian, Yu-Cheng Liang, Minghao Liu, Zhikang Niu, and 15 others. 2025b. [Mmar: A challenging benchmark for deep reasoning in speech, audio, music, and their mix](#). *ArXiv*, abs/2505.13032.
- Hong Sun, Xue Li, Yi Xu, Youkow Homma, Qin-hao Cao, Min man Wu, Jian Jiao, and Denis Xavier Charles. 2023. [Autohint: Automatic prompt optimization with hint generation](#). *ArXiv*, abs/2307.07415.

- Mirac Suzgun, Nathan Scales, Nathanael Scharli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Huai hsin Chi, Denny Zhou, and Jason Wei. 2022. [Challenging big-bench tasks and whether chain-of-thought can solve them](#). In *Annual Meeting of the Association for Computational Linguistics*.
- Yu Wang, Shiwan Zhao, Zhihu Wang, Heyuan Huang, Ming Fan, Yubo Zhang, Zhixing Wang, Haijun Wang, and Ting Liu. 2024. [Strategic chain-of-thought: Guiding accurate reasoning in llms through strategy elicitation](#). *ArXiv*, abs/2409.03271.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. [Chain of thought prompting elicits reasoning in large language models](#). *ArXiv*, abs/2201.11903.
- Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.
- Zhifei Xie, Mingbao Lin, Zihang Liu, Pengcheng Wu, Shuicheng Yan, and Chunyan Miao. 2025. [Audio-reasoner: Improving reasoning capability in large audio language models](#). *ArXiv*, abs/2503.02318.
- Zhen Xiong, Yujun Cai, Zhecheng Li, Junsong Yuan, and Yiwei Wang. 2025. [Thinking with sound: Audio chain-of-thought enables multimodal reasoning in large audio-language models](#).
- Chih-Kai Yang, Neo Ho, and Hung yi Lee. 2025. [Towards holistic evaluation of large audio-language models: A comprehensive survey](#). *ArXiv*, abs/2505.15957.
- Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc Le, Ed Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. 2024. [Self-discover: Large language models self-compose reasoning structures](#). *ArXiv*, abs/2402.03620.
- Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. [Large language models are human-level prompt engineers](#). *ArXiv*, abs/2211.01910.

## A Use of AI Assistants

We used Claude for grammar refinement and occasional assistance in code development. All language model outputs served solely as references; the final writing and code were thoroughly reviewed and authored by the authors.

## B Experimental Cost

The total experimental cost across all models, benchmarks, and methods was approximately \$500 USD. This includes all instruction induction experiments, baseline evaluations, and cross-model transfer analysis.

## C Model Implementation Details

This section provides detailed version identifiers and configurations for all models evaluated in Section 3.2. We evaluate models from two families: Voxtral (Liu et al., 2025) and Gemini (2.0 series (Google, 2025) and 2.5 series (Comanici et al., 2025)). The exact API identifiers and configurations are listed below.

**Voxtral Mini** <voxtral-mini-2507> 3B

**Voxtral Small** <voxtral-small-2507> 24B

**Gemini 2.0 Flash Lite** <gemini-2.0-flash-lite-001>

**Gemini 2.0 Flash** <gemini-2.0-flash-001>

**Gemini 2.5 Flash Lite** <gemini-2.5-flash-lite>, thinking\_budget=0

**Gemini 2.5 Flash Lite (Thinking)** <gemini-2.5-flash-lite>, thinking\_budget=8192 (default)

**Gemini 2.5 Flash** <gemini-2.5-flash>, thinking\_budget=0

**Gemini 2.5 Flash (Thinking)** <gemini-2.5-flash>, thinking\_budget=8192 (default)

**Gemini 2.5 Pro (Thinking)** <gemini-2.5-pro>, thinking\_budget=8192 (default)

For Gemini 2.5 models, the thinking\_budget parameter controls extended reasoning mode: setting it to 0 disables thinking, while 8192 enables the default thinking capacity. All experiments use official API endpoints with temperature 0 for deterministic outputs.

## D Data Preprocessing and Evaluation Metrics

### D.1 Audio Format Conversion for MMAR

The original MMAR dataset uses WAV format audio files, which are too large for few-shot prompting scenarios. Many commercial LALM API endpoints impose file size limits (e.g., 25MB for Gemini), making it infeasible to include multiple audio demonstrations in a single request. To address this limitation, we converted all MMAR audio files from WAV to MP3 format using VLC media player<sup>1</sup>, an open-source software.

The conversion parameters were configured as follows:

- Bitrate: 64 kb/s
- Channels: 1 (mono, as required by models such as Gemini)
- Sample rate: 22050 Hz

These settings substantially reduce file sizes while preserving audio quality sufficient for reasoning tasks. The BBA dataset did not require conversion as its audio files were already within acceptable size limits.

### D.2 Evaluation Metric

We use micro-averaged accuracy as our evaluation metric across all experiments. Micro-averaging calculates accuracy by counting total correct predictions across all instances, treating each instance equally regardless of task category. This provides a single aggregate performance measure suitable for comparing methods across multiple reasoning tasks within each benchmark.

## E Additional Analysis

### E.1 Impact of the Number of AQA Examples

We investigate how the number of examples  $N$  used for instruction induction affects performance. Table 8 presents results for  $N \in \{1, 3, 5\}$  across 9 models on both benchmarks.

The results reveal a pattern distinct from text-based instruction induction. While Chen et al. (2024b) observed consistent improvements with more examples in text domains, our results show that  $N = 3$  provides optimal performance. Several

<sup>1</sup><https://www.videolan.org/vlc/>

models degrade when using  $N = 5$ , suggesting diminishing or negative returns with too many audio examples.

We speculate this stems from the nature of audio processing. Unlike text examples which provide purely linguistic information, audio examples require LALMs to integrate acoustic and linguistic information simultaneously across multiple sequential inputs. Providing too many examples may introduce interference patterns that hinder rather than help instruction induction. Understanding how to effectively aggregate information across multiple audio examples represents an important direction for future research.

Based on these findings, we use  $N = 3$  for all main experiments.

## E.2 Complete Task Breakdown Results

Figure 4 presents the complete performance breakdown across all 9 models on BBA subtasks. Compared to Figure 2 in the main paper, this includes three additional models with Thinking mode enabled: Gemini 2.5 Flash Lite (Thinking), Gemini 2.5 Flash (Thinking), and Gemini 2.5 Pro (Thinking). Notably, these Thinking-enabled models show consistent improvements or maintain performance across all BBA subtasks when using Audio-Induct, with no degradation observed on any task.

## E.3 Comparison Against Strategic Chain-of-Thought

Figure 5 extends the cross-model transfer analysis by including Strategic Chain-of-Thought (S-CoT) as an additional baseline. As shown in Figure 5c(c), instructions induced by stronger models, particularly Gemini 2.5 Flash and Gemini 2.5 Pro (Thinking), consistently outperform S-CoT across nearly all inference models, achieving improvements in 17 out of 18 cross-model combinations. This demonstrates that high-quality induced instructions can surpass the state-of-the-art S-CoT baseline while eliminating the need for strategy generation at each inference.

## F Inference Cost Analysis

We provide comprehensive inference cost analysis to substantiate the claimed cost savings. All costs shown are for Stage 2 (task instruction application). Stage 1 (instruction induction) is performed once per task, and its cost is negligible when amortized across queries.

## F.1 API Pricing

Table 4 shows the pricing structure for all evaluated models. Voxtral charges audio by seconds while Gemini charges by tokens (1 second  $\approx$  32 tokens).

Model	Input	Audio	Output	Method
Voxtral Mini	\$0.04	\$0.000017	\$0.04	/sec
Voxtral Small	\$0.10	\$0.000067	\$0.30	/sec
Gemini 2.0 Flash Lite	\$0.07	\$0.075	\$0.30	/1M
Gemini 2.0 Flash	\$0.10	\$0.700	\$0.40	/1M
Gemini 2.5 Flash Lite	\$0.10	\$0.300	\$0.40	/1M
Gemini 2.5 Flash	\$0.30	\$1.000	\$2.50	/1M
Gemini 2.5 Pro	\$1.25	\$1.250	\$10.00	/1M

Table 4: API pricing for evaluated models. Voxtral charges audio by seconds; Gemini charges by tokens (1 second  $\approx$  32 tokens). Text prices are per 1M tokens.

## F.2 Input Cost Comparison

Table 5 shows the API input cost (text + audio) per 100 samples at inference. Audio-Induct achieves 53-78% input cost reduction compared to 3-Shot by eliminating demonstration audio at inference. Our input costs remain comparable to Z-CoT and S-CoT.

## F.3 Total Cost Comparison

Table 6 shows the total API cost (input + output) per 100 samples. Eight out of nine models achieve 19-78% total cost reduction compared to 3-Shot. Gemini 2.5 Pro shows comparable cost to S-CoT but achieves superior accuracy (+0.10% on BBA, +1.02% on MMAR).

## F.4 Token Usage Analysis

Table 7 provides detailed token usage per sample at inference. The format shows Input Text tokens, Input Audio tokens, and Output Text tokens separated by vertical bars. 3-Shot prompting uses approximately  $3.9\times$  more input audio tokens (demonstrations + query) compared to Our method (query only). Text input cost is generally negligible compared to audio cost, except for Gemini 2.5 Pro where text input pricing approaches audio input pricing.

## G Prompt Templates and Examples

This section provides all prompt templates used in our experiments. The prompts are organized into four color-coded categories, with detailed templates shown in tables below.

Model	BBA					MMAR				
	Z-CoT	3-Shot	S-CoT	Our	$\delta$	Z-CoT	3-Shot	S-CoT	Our	$\delta$
Voxtral Mini	0.033	0.125	0.033	0.034	<b>-73.0%</b>	0.032	0.149	0.032	0.033	<b>-77.6%</b>
Voxtral Small	0.130	0.500	0.130	0.133	<b>-73.5%</b>	0.126	0.595	0.127	0.129	<b>-78.4%</b>
Gemini 2.0 Flash Lite	0.005	0.016	0.006	0.007	<b>-59.0%</b>	0.006	0.020	0.006	0.008	<b>-61.9%</b>
Gemini 2.0 Flash	0.036	0.134	0.036	0.039	<b>-71.1%</b>	0.036	0.160	0.036	0.039	<b>-75.9%</b>
Gemini 2.5 Flash Lite	0.021	0.077	0.022	0.025	<b>-67.2%</b>	0.022	0.094	0.022	0.025	<b>-73.0%</b>
Gemini 2.5 Flash Lite (T)	0.021	0.077	0.022	0.025	<b>-66.9%</b>	0.022	0.094	0.022	0.026	<b>-72.8%</b>
Gemini 2.5 Flash	0.070	0.255	0.072	0.086	<b>-66.5%</b>	0.072	0.310	0.073	0.088	<b>-71.6%</b>
Gemini 2.5 Flash (T)	0.070	0.255	0.072	0.087	<b>-65.9%</b>	0.072	0.310	0.073	0.090	<b>-71.0%</b>
Gemini 2.5 Pro (T)	0.109	0.345	0.115	0.160	<b>-53.7%</b>	0.117	0.432	0.122	0.170	<b>-60.7%</b>

Table 5: API input cost per 100 samples at inference (USD).  $\delta$  shows percentage savings of Our method compared to 3-Shot. (T) denotes Thinking mode enabled.

Model	BBA					MMAR				
	Z-CoT	3-Shot	S-CoT	Our	$\delta$	Z-CoT	3-Shot	S-CoT	Our	$\delta$
Voxtral Mini	0.034	0.126	0.034	0.035	<b>-72.4%</b>	0.033	0.149	0.033	0.034	<b>-77.2%</b>
Voxtral Small	0.134	0.504	0.136	0.139	<b>-72.4%</b>	0.130	0.597	0.132	0.134	<b>-77.6%</b>
Gemini 2.0 Flash Lite	0.009	0.018	0.012	0.013	<b>-27.1%</b>	0.008	0.022	0.012	0.013	<b>-42.3%</b>
Gemini 2.0 Flash	0.042	0.136	0.045	0.048	<b>-65.0%</b>	0.039	0.162	0.041	0.043	<b>-73.4%</b>
Gemini 2.5 Flash Lite	0.046	0.099	0.069	0.053	<b>-46.0%</b>	0.043	0.116	0.061	0.053	<b>-54.1%</b>
Gemini 2.5 Flash Lite (T)	0.097	0.145	0.107	0.117	<b>-19.3%</b>	0.108	0.222	0.126	0.119	<b>-46.6%</b>
Gemini 2.5 Flash	0.242	0.367	0.273	0.205	<b>-44.2%</b>	0.177	0.421	0.218	0.246	<b>-41.4%</b>
Gemini 2.5 Flash (T)	0.390	0.688	0.441	0.428	<b>-37.8%</b>	0.317	0.575	0.394	0.431	<b>-25.1%</b>
Gemini 2.5 Pro (T)	1.573	1.772	1.812	1.761	-0.6%	1.324	1.453	1.634	1.635	+12.5%

Table 6: API total cost (input + output) per 100 samples at inference (USD).  $\delta$  shows percentage savings compared to 3-Shot. 8 out of 9 models achieve 19-78% total cost reduction.

## G.1 Prompt Organization

**Fixed Values (Orange)** BBA: Task information (Table 9) is shared across all four subtasks. Answer formats (Table 10) are task-specific for Formal Fallacies, Navigate, Object Counting, and Web of Lies. MMAR: Task information (Table 11) and answer format (Table 12) are both shared across all tasks, as MMAR uses uniform multiple-choice format.

**Stage 1: Instruction Induction (Blue)** Meta-prompt (Table 13) directs LALMs through structured analysis to induce task instructions. User prompt (Table 14) provides  $N = 3$  examples with task information and answer format from the orange boxes.

**Stage 2: Task Instruction Application (Green)** System prompt (Table 15) contains the induced instruction from Stage 1 and answer format from orange boxes. User prompt (Table 16) contains only the new question.

**Baseline: Z-CoT (Gray)** System prompt (Table 17) uses task description and answer format from orange boxes. User prompt (Table 18) uses “Let’s think step by step.”

**Baseline: Strategic CoT (Gray)** System prompt (Table 19) adds strategy generation before reason-

ing, using values from orange boxes. User prompt (Table 20) follows same format as Z-CoT.

**Baseline: 3-Shot Prompting (Gray)** System prompt (Table 21) uses values from orange boxes. User prompt (Table 22) includes three audio-question-answer examples before the new question.

## G.2 Placeholder Conventions

Prompts use placeholders in curly braces: {task\_information} and answer format {} are filled from orange boxes, and {newQ} is the question for inference. Complete templates are provided in the tables below.

## G.3 Example: Induced Instruction for MMAR Music Theory Task

Table 23 shows the complete output from Gemini 2.5 Flash when using the instruction induction meta-prompt (Table 13) with  $N = 3$  examples from MMAR music\_theory subtask. This task requires identifying specific musical elements (e.g., chord names, techniques, dynamics) from audio clips and selecting the correct answer from multiple-choice options. The output demonstrates how the model analyzes multi-modal patterns across audio and text to generate structured operational steps for music analysis tasks.

Model	BBA				MMAR			
	Z-CoT	3-Shot	S-CoT	Our	Z-CoT	3-Shot	S-CoT	Our
Voxtral Mini	223 453 168	261 1766 62	263 453 189	503 453 197	288 377 116	454 1502 12	328 377 153	639 377 162
Voxtral Small	223 453 151	261 1766 137	263 453 204	516 453 207	288 377 120	454 1502 84	328 377 168	542 377 164
Gemini 2.0 Flash Lite	243 479 123	292 1873 60	285 479 197	409 479 216	314 463 80	509 2214 66	356 463 189	575 463 171
Gemini 2.0 Flash	243 479 148	292 1873 53	285 479 216	522 479 223	314 463 80	509 2214 59	356 463 118	626 463 113
Gemini 2.5 Flash Lite	245 629 624	294 2465 545	287 629 1189	631 629 703	316 620 535	512 2947 554	358 620 957	667 620 696
Gemini 2.5 Flash Lite (T)	245 629 1900	294 2465 1705	287 629 2133	658 629 2291	316 620 2161	512 2947 3220	358 620 2587	687 620 2331
Gemini 2.5 Flash	245 629 685	294 2465 445	287 629 805	755 629 476	316 620 422	512 2947 442	358 620 579	871 620 632
Gemini 2.5 Flash (T)	245 629 1276	294 2465 1730	287 629 1479	809 629 1362	316 620 980	512 2947 1059	358 620 1286	932 620 1363
Gemini 2.5 Pro (T)	245 629 1463	294 2465 1426	287 629 1697	650 629 1600	316 620 1207	512 2947 1020	358 620 1512	740 620 1464

Table 7: Token usage per sample at inference. Format: Input Text | Input Audio (tokens) | Output Text. Audio duration: BBA query  $\sim$ 19s, 3-Shot demos  $\sim$ 75s; MMAR query  $\sim$ 19s, 3-Shot demos  $\sim$ 89s.

#### G.4 Example: Induced Instruction for BBA Web of Lies Task

Table 24 shows another instruction induction example from BBA web\_of\_lies subtask, generated by Gemini 2.5 Flash. This task involves logical deduction: determining whether the last person in a chain of truthfulness statements tells the truth or lies. The induced instruction differs significantly from the MMAR example, demonstrating how Audio-Induct adapts to tasks requiring sequential logical reasoning rather than perceptual analysis. Both examples use the same meta-prompt but produce task-appropriate instructions.

Model	BBA			MMAR		
	N=1	N=3	N=5	N=1	N=3	N=5
Voxtral Mini	53.96	56.15	<b>56.98</b>	43.32	<b>46.19</b>	43.99
Voxtral Small	84.58	<b>87.40</b>	84.48	54.82	<b>56.01</b>	55.67
Gemini 2.0 Flash Lite	<b>79.79</b>	76.15	78.85	<b>61.25</b>	60.91	58.71
Gemini 2.0 Flash	79.27	<b>91.35</b>	83.23	<b>63.79</b>	63.45	62.61
Gemini 2.5 Flash Lite	94.38	<b>95.31</b>	95.10	52.96	<b>57.53</b>	56.35
Gemini 2.5 Flash Lite (Thinking)	97.19	<b>97.40</b>	96.98	<b>58.04</b>	56.18	56.01
Gemini 2.5 Flash	97.50	98.33	<b>98.65</b>	66.16	<b>69.88</b>	67.34
Gemini 2.5 Flash (Thinking)	<b>98.44</b>	98.33	98.33	69.04	<b>70.90</b>	67.85
Gemini 2.5 Pro (Thinking)	<b>99.58</b>	<b>99.58</b>	<b>99.58</b>	<b>80.88</b>	<b>80.88</b>	79.86
Win $N = 1$ Rate	-	6/9	4/9	-	5/9	4/9

Table 8: Performance (%) of different models on BBA and MMAR, as a function of the number of rationales  $N$ . For each  $N > 1$ , the last row reports the number of models that outperform  $N = 1$ .

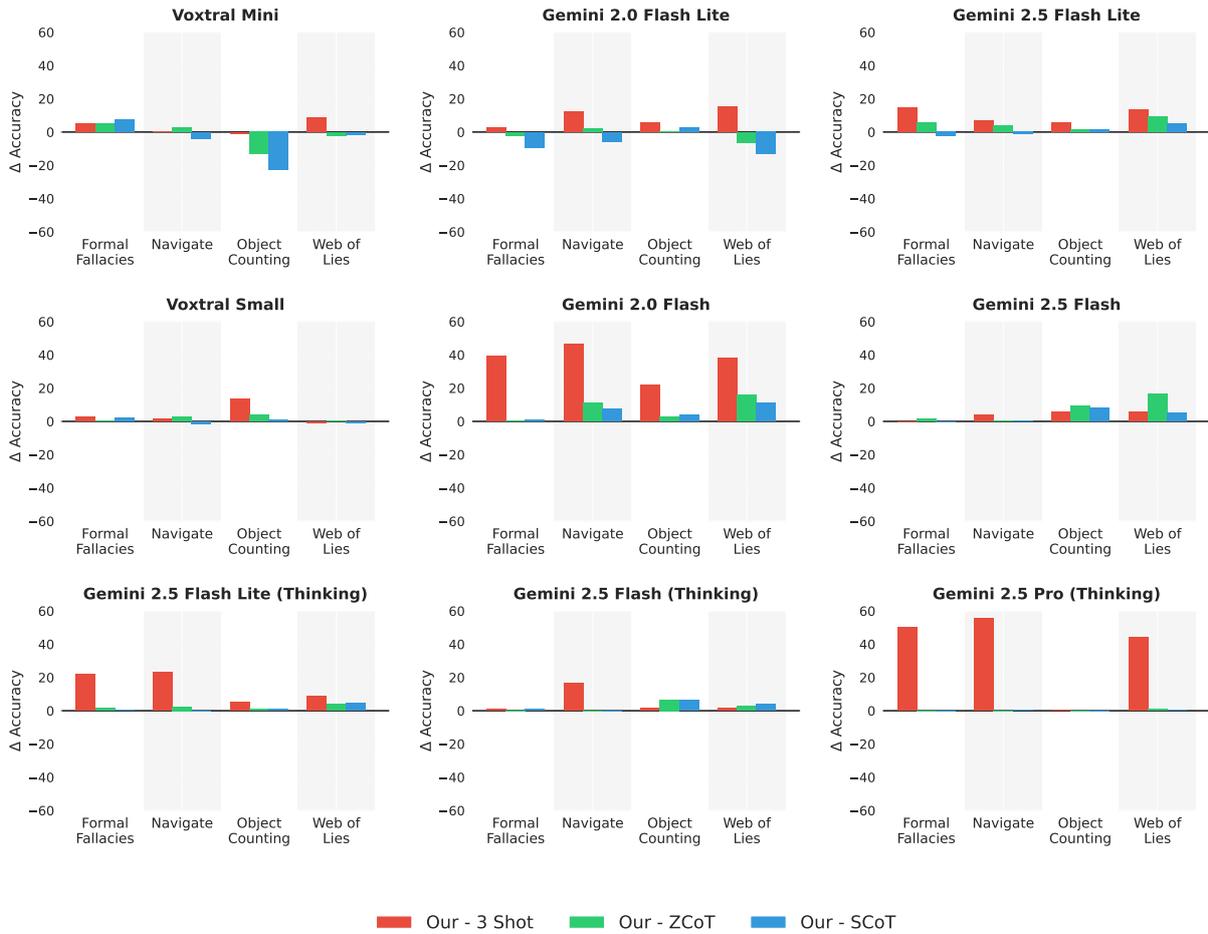
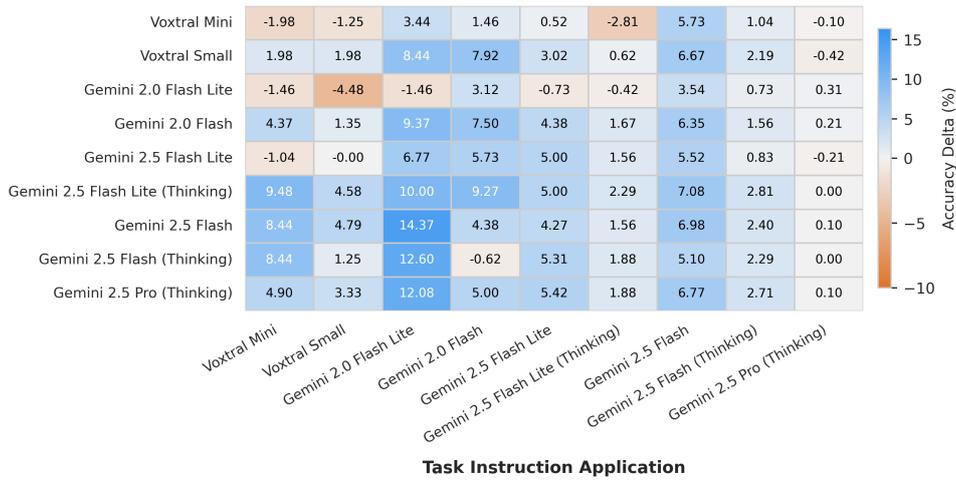
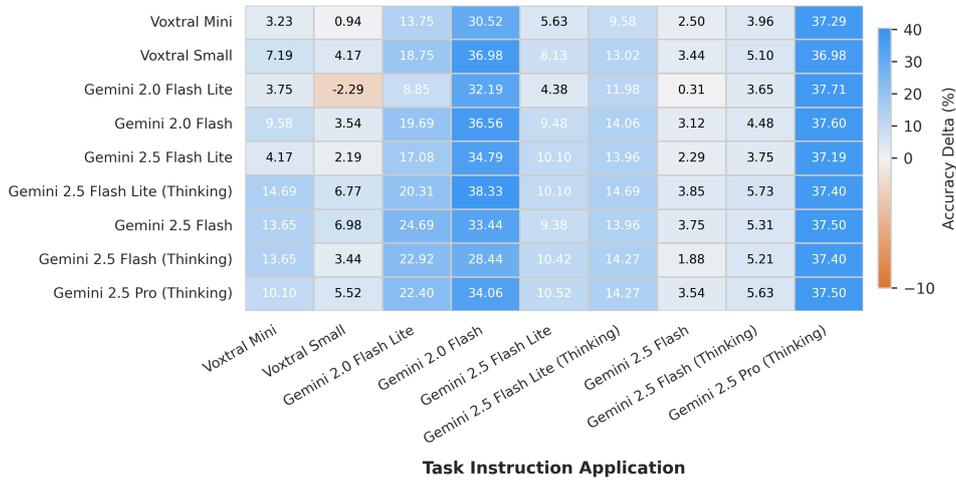


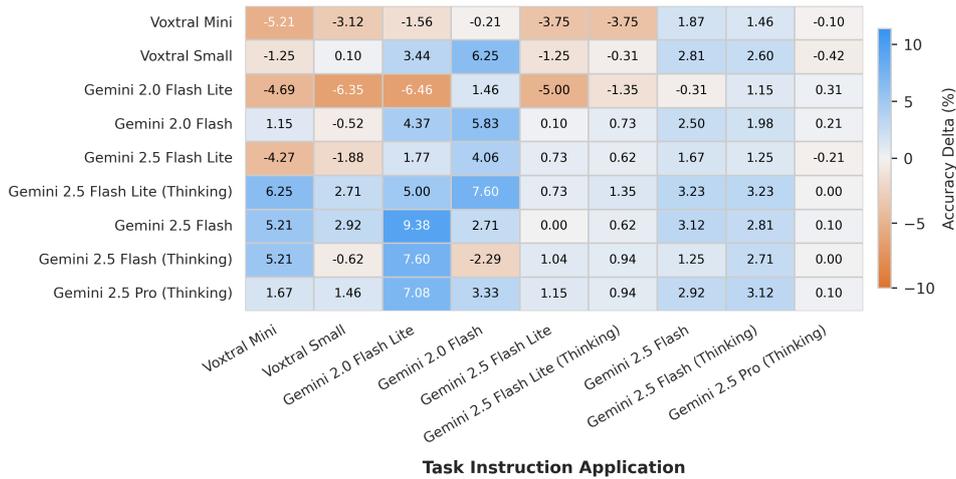
Figure 4: Complete head-to-head performance comparison across all 9 models on BBA subtasks. The accuracy delta indicates the performance difference between Audio-Induct and three baselines (Z-CoT, 3-Shot, S-CoT) for each model on the four BBA subtasks: Formal Fallacies, Navigate, Object Counting, and Web of Lies.



(a) Our - Z-CoT



(b) Our - 3-Shot



(c) Our - S-CoT

Figure 5: Complete cross-model instruction transfer results on BBA. Each cell shows the accuracy delta when instructions induced by one model (y-axis) are applied to another model (x-axis), compared to (a) Z-CoT, (b) 3-Shot, and (c) S-CoT prompting. Blue indicates improvement, orange indicates decline.

BBA Task Information (Shared across all subtasks)
You will be provided with an audio clip that contains both the question and the answer choices. Listen carefully and answer.

Table 9: Task information used for all BBA subtasks. This text fills the {task\_information} placeholder in all prompts.

BBA Answer Formats (Task-specific)
<p><b>Formal Fallacies:</b> The output should be a single word: `valid` or `invalid`.</p> <p><b>Navigate:</b> The output is a single word answer, either `Yes` or `No`.</p> <p><b>Object Counting:</b> The output is a single integer.</p> <p><b>Web of Lies:</b> The output is a single word answer, either `Yes` or `No`.</p>

Table 10: Answer format specifications for each BBA subtask. These fill the answer format placeholder {} in system and user prompts.

MMAR Task Information (Shared across all tasks)
You will be provided with an audio clip and a corresponding question with multiple-choice options. Your task is to listen carefully to the audio and analyze its content to select the most appropriate answer from the given choices.

Table 11: Task information used for all MMAR tasks. This text fills the {task\_information} placeholder.

MMAR Answer Format (Shared across all tasks)
The output is a single letter corresponding to the correct option from the list (e.g., A, B, C, etc.).

Table 12: Answer format specification for all MMAR tasks (all are multiple choice). This fills the answer format placeholder {}.

### Meta-Prompt for Audio Instruction Induction

You are an expert in both Natural Language Processing (NLP) and Audio Processing, with exceptional skills in analyzing multi-modal datasets and formulating precise task instructions. Your expertise includes extracting significant patterns and rules from complex datasets that combine textual and audio information.

Your current task is to analyze a set of examples that include both text and audio, derive insights, and formulate a clear and detailed task instruction. This instruction will guide the execution of related multi-modal tasks, such as Audio Question Answering (AQA) or other audio-language tasks.

#### Follow these steps:

##### 1. Analyze the Input:

Begin by carefully examining both the textual and audio components of the input. Wrap your analysis inside `<multi_modal_analysis>` tags in your thinking block. Include the following:

- a. Separate analysis of textual and audio components
- b. Synthesis of findings from both modalities
- c. Identification of common patterns across examples
- d. List of potential challenges in executing the task

```
<multi_modal_analysis>
```

[Your multi modal analysis here, with each element under its own subheading]

```
</multi_modal_analysis>
```

##### 2. Formulate the Task Instruction:

Based on your analysis, create a detailed task instruction that includes the following elements:

- a. *Task Content*: Clearly define the purpose of the task and the specific activities required to be completed, considering both textual and audio aspects.
- b. *Input Format*: Provide detailed descriptions of the types of data accepted (text and audio), their formats, and how to process these multi-modal data effectively.
- c. *Operational Steps*: Detail the specific step-by-step procedures required to complete the task, including how to integrate information from both textual and audio inputs.

##### 3. Format the Final Output:

Present your task instruction in the following format:

```
<task_instruction>
```

[Your task instruction here, with each element under its own subheading]

```
</task_instruction>
```

Ensure that your task instruction is concise, clear, and easily understandable by users. It should provide all necessary information for someone to successfully complete a multi-modal task involving both text and audio inputs.

Remember to focus solely on creating the task instruction based on the given information and examples. Do not attempt to complete the task itself.

Your final output should include only the “multi modal analysis” within the `<multi_modal_analysis>` tag and the “task instruction” within the `<task_instruction>` tag, without repeating or rephrasing any analysis from the thinking block.

Table 13: Meta-prompt(system prompt) for instruction induction. This prompt directs LALMs to analyze few AQA examples with ground-truth answers, identify multi-modal reasoning patterns, and induce reusable task instructions through structured analysis and formulation steps.

### User Prompt Template for Providing Examples

Here is the task\_information, answer\_format and the examples you need to create a Task Instructions for:

```
<task_information>
{}
</task_information>

<answer_format>
{}
</answer_format>

<examples>
{}
</examples>
```

Table 14: User prompt template for instruction induction. The placeholders {} are filled with task-specific information, answer format specifications, and  $N$  AQA example pairs with ground-truth answers. This prompt is combined with the meta-prompt (Table 13) to guide the instruction induction process.

### System Prompt for Task Instruction Application

```
{}
```

**\*\*Answer Format\*\*:**

```
{}
```

Follow these steps carefully:

1. Based on the `Operational Steps`, provide a step-by-step deduction that answers the question, incorporating insights from both the audio and the text:  
<deduction>  
[Your step-by-step deduction here]  
</deduction>
2. Based on your deduction, provide the final answer according to the rules specified in the "Answer Format" section. If uncertain and "Answer Format" is a list of options, select the option that best matches your reasoning. Present your final answer in the following format:  
<final\_answer>  
[Your final answer here]  
</final\_answer>

**\*\*Important Notes\*\*:**

- Do not use programming or code to solve this question.
- Ensure your final answer strictly follows the specified format

Table 15: System prompt for applying induced instructions. The first placeholder {} is filled with the induced task instruction from Stage 1, and the second placeholder contains the answer format specification. This prompt guides LALMs to follow the operational steps and provide structured reasoning.

### User Prompt for Task Instruction Application

```
[Question]
{}
[Answer]
Let's follow the "Operational Steps", think step by step, then ensure your final answer strictly follows the specified format.
```

Table 16: User prompt for applying induced instructions. The placeholder {} is filled with the question text. This prompt is combined with the system prompt (Table 15) and the audio input to solve new AQA instances.

```

Z-CoT System Prompt

[Task Instruction]
{task_information}
**Answer Format**:
{}

Follow these steps carefully:
1. Provide a step-by-step deduction that answers the question,
   incorporating insights from both the audio and the text:
   <deduction>
   [Your step-by-step deduction here]
   </deduction>

2. Based on your deduction, provide the final answer according
   to the rules specified in the "Answer Format" section.
   If uncertain and "Answer Format" is a list of options,
   select the option that best matches your reasoning.
   Present your final answer in the following format:
   <final_answer>
   [Your final answer here]
   </final_answer>

**Important Notes**
- Do not use programming or code to solve this question.
- Ensure your final answer strictly follows the specified format

```

Table 17: System prompt for Z-CoT baseline. Placeholder {task\_information} is filled with task description, and the second placeholder {} contains answer format specification (e.g., multiple choice options or numerical format).

```

Z-CoT User Prompt

[Question]
{newQ}
[Answer]
Let's think step by step, then ensure your final answer
strictly follows the specified format.

```

Table 18: User prompt for Z-CoT baseline. Placeholder {newQ} is filled with the question text for the new instance to be solved.

#### S-CoT System Prompt

```
[Task Instruction]
{task_information}
**Answer Format**:
{}

Follow these steps carefully:
1. Carefully analyze the audio and the question to generate
   the strategic knowledge that would best guide the
   problem-solving process.
   <strategy>
   [Your strategy here]
   </strategy>

2. Provide a step-by-step deduction that answers the question,
   incorporating insights from both the audio and the text:
   <deduction>
   [Your step-by-step deduction here]
   </deduction>

3. Based on your deduction, provide the final answer according
   to the rules specified in the "Answer Format" section.
   If uncertain and "Answer Format" is a list of options,
   select the option that best matches your reasoning.
   Present your final answer in the following format:
   <final_answer>
   [Your final answer here]
   </final_answer>

**Important Notes**
- Do not use programming or code to solve this question.
- Ensure your final answer strictly follows the specified format
```

Table 19: System prompt for Strategic CoT baseline. Placeholder {task\_information} is filled with task description, and the second placeholder {} contains answer format specification. S-CoT adds an explicit strategy generation step before reasoning.

#### S-CoT User Prompt

```
[Question]
{newQ}
[Answer]
Let's think step by step, then ensure your final answer
strictly follows the specified format.
```

Table 20: User prompt for Strategic CoT baseline. Placeholder {newQ} is filled with the question text for the new instance.

### 3-Shot System Prompt

```
[Task Instruction]
{task_information}
**Answer Format**:
{}
```

Follow these steps carefully:

1. Provide a step-by-step deduction that answers the question, incorporating insights from both the audio and the text:

```
<deduction>
```

```
[Your step-by-step deduction here]
```

```
</deduction>
```

2. Based on your deduction, provide the final answer according to the rules specified in the "Answer Format" section.

If uncertain and "Answer Format" is a list of options, select the option that best matches your reasoning.

Present your final answer in the following format:

```
<final_answer>
```

```
[Your final answer here]
```

```
</final_answer>
```

**Important Notes:**

- Do not use programming or code to solve this question.
- Ensure your final answer strictly follows the specified format

Table 21: System prompt for 3-Shot prompting baseline. Placeholder {task\_information} is filled with task description, and the second placeholder {} contains answer format specification.

### 3-Shot User Prompt

```
[Question]
{Q1 example}
[Answer]
{A1 example}
```

```
[Question]
{Q2 example}
[Answer]
{A2 example}
```

```
[Question]
{Q3 example}
[Answer]
{A3 example}
```

```
[Question]
{newQ}
[Answer]
```

Table 22: User prompt for 3-Shot prompting baseline. Placeholders {Q1 example}, {Q2 example}, {Q3 example} are filled with three example questions (each accompanied by audio), {A1 example}, {A2 example}, {A3 example} are filled with their corresponding ground-truth answers, and {newQ} is the new question to be solved.

## Complete Instruction Induction Output

<multi\_modal\_analysis>

### a. Separate analysis of textual and audio components

*Textual Analysis:* The questions consistently ask for specific information derived from the audio, focusing on musical theory (e.g., chord names, II-V-I progression), musical techniques (e.g., glissando, overtone, vibrato), or musical parameters (e.g., number of pitches, dynamics, speed changes). Each question is accompanied by multiple-choice options with distinct musical terminology or numerical values. The answer format is a single letter.

*Audio Analysis:* The audio clips are short musical excerpts with varying complexity. Features include clear chord progressions (likely jazz or classical influenced), sustained vocal/instrumental sounds demonstrating specific techniques (e.g., overtone singing), and melodic lines with dynamics and tempo changes. To answer questions, one must identify specific chords and qualities, musical techniques, and count specific musical elements.

### b. Synthesis of findings from both modalities

The task requires deep understanding of both musical theory/terminology (from text) and the ability to accurately perceive and interpret these elements within the audio. The textual questions guide the listener on what specific features to focus on in the audio. The options provided serve as potential interpretations of the audio, requiring careful discrimination. This is essentially an Audio Question Answering (AQA) task focused on musical content.

### c. Identification of common patterns across examples

All examples demonstrate strong multi-modal dependency between audio and text. All are firmly within the musical domain, requiring knowledge of music theory, terminology, and aural skills. Each question targets specific musical features (e.g., chords, techniques, dynamics, pitch count) using consistent multiple-choice format. Answering correctly requires precise identification of musical elements, not just general understanding.

### d. List of potential challenges in executing the task

Users without formal musical training or highly developed aural skills might struggle to identify specific chords, complex techniques, or subtle changes. Some musical elements could have slight perceptual variations. Musical terms might be interpreted differently depending on context. The multiple-choice options can be very similar, requiring fine-grained discrimination. For chord progression questions, understanding key and functional harmony is crucial.

</multi\_modal\_analysis>

<task\_instruction>

### Task Content

The task requires you to act as a music analyst. You will be presented with an audio clip containing musical content and a corresponding question with multiple-choice options. Your primary objective is to meticulously listen to and analyze the provided audio, identifying specific musical elements, techniques, or characteristics as prompted by the question. Based on your auditory analysis, you must then select the single most accurate answer from the given options.

### Input Format

1. *Audio Clip:* A short audio file containing a musical excerpt.
2. *Question:* A textual question specifying what musical aspect to focus on (e.g., chord names, musical techniques, dynamics, tempo, pitch count).
3. *Options:* A list of multiple-choice answers (typically A, B, C, D).

### Operational Steps

1. *Listen to the Audio:* Play the provided audio clip carefully. Pay close attention to all musical parameters, including melody, harmony, rhythm, timbre, dynamics, and tempo.
2. *Read the Question:* Identify the specific musical information or characteristic it is asking for.
3. *Analyze Audio in Relation to Question:* Re-listen to the audio with the question in mind. Actively seek out the specific musical elements that the question targets.
4. *Evaluate the Options:* Compare your auditory analysis with each option. Eliminate options that contradict what you hear.
5. *Select the Best Answer:* Choose the option that most accurately describes the musical content.
6. *Format the Output:* Provide your answer as a single uppercase letter (e.g., A, B, C, or D).

</task\_instruction>

Table 23: Complete instruction induction output for MMAR music\_theory subtask, generated by Gemini 2.5 Flash in a single inference. The output contains both the multi-modal analysis and the induced task instruction. This demonstrates the two-step reasoning process guided by the meta-prompt (Table 13).

Complete Instruction Induction Output
<pre> &lt;multi_modal_analysis&gt;  <b>a. Separate analysis of textual and audio components</b> <i>Textual Analysis:</i> The task information states that the audio clip contains both the question and answer choices, implying a narrative structure. The answer format specifies a binary "Yes" or "No" response. The examples show a consistent pattern: a series of statements about individuals telling the truth or lying, followed by a final question about one specific individual's truthfulness. The core task is to deduce truthfulness based on a chain of logical dependencies, where each statement links the truthfulness of one person to another's statement. <i>Audio Analysis:</i> The audio consistently presents a sequence of spoken statements, each attributing truthfulness or deceit to another person. The structure follows: "Question. [Person A] [action]. [Person B] says [Person A] [action]..." concluding with "Does [Final Person] tell the truth?" There is a clear spoken prompt "Answer the question" followed by a distinct audio cue (short, high-pitched tone) indicating the transition point.  <b>b. Synthesis of findings from both modalities</b> This is an AQA task where the entire context (premises and final question) is embedded within a single audio stream. The textual information provides the logical framework and expected output format, while the audio delivers specific instances. The audio content needs to be transcribed and logically processed to determine the final answer. The task requires understanding spoken language, extracting named entities (people), interpreting statements about truthfulness or deceit, and chaining these statements logically. The final question always asks about the truthfulness of the last person mentioned in the chain.  <b>c. Identification of common patterns across examples</b> All examples involve a chain of deduction where each statement refers to the truthfulness of the previously mentioned person. Binary truth values apply (each person either "tells the truth" or "lies"). The final question structure is always "Does [Last Person in Chain] tell the truth?" with "Yes" or "No" output. Audio cues are consistent: "Question." at the start, "Answer the question." before the final answer, and a distinct audio tone marking the end of the question segment.  <b>d. List of potential challenges in executing the task</b> Challenges include accurate speech-to-text (misinterpreting names or keywords like "lies" vs. "tells the truth"), logical deduction (correctly chaining truth values), named entity recognition (tracking each person), context window/memory (keeping track of truthfulness status as the chain unfolds), handling negation (correctly interpreting "X says Y lies" versus "X says Y tells the truth"), and real-time processing if applicable.  &lt;/multi_modal_analysis&gt;  -----  &lt;task_instruction&gt;  <b>Task Content</b> The task requires you to listen to an audio clip that presents a series of statements about individuals' truthfulness or deceit, forming a logical chain. The audio will conclude with a question about whether a specific individual tells the truth. Your goal is to deduce the truthfulness of the final subject based on the provided statements and answer "Yes" or "No".  <b>Input Format</b> The input will be a single audio file containing spoken language that sequentially presents: (1) An initial statement about a person's truthfulness (e.g., "William lies"). (2) A series of subsequent statements, each attributing truthfulness or deceit to the previously mentioned person (e.g., "Inga says William tells the truth"). (3) A final question asking about the truthfulness of the last person mentioned in the chain (e.g., "Does Crista tell the truth?"). (4) A concluding phrase "Answer the question" followed by a distinct audio cue indicating the end of the question segment.  <b>Operational Steps</b> 1. <i>Transcribe Audio:</i> Process the entire audio input to obtain a textual transcript of all spoken content. 2. <i>Identify Initial Premise:</i> Extract the first statement that establishes the truthfulness (or deceit) of the initial person. Assign a truth value (True/False) to this person. 3. <i>Parse Chained Statements:</i> For each subsequent statement, identify the speaker, the person being referred to, and the assertion made about their truthfulness. 4. <i>Deduce Truth Values:</i> Iteratively deduce the truthfulness of each new speaker based on their statement and the known truthfulness of the person they are referring to. If Person B says "Person A tells the truth," Person B's truth value is the same as Person A's. If Person B says "Person A lies," Person B's truth value is the opposite of Person A's. 5. <i>Identify Final Question:</i> Locate the final question in the transcript, which will be of the form "Does [Last Person] tell the truth?" 6. <i>Retrieve Final Truth Value:</i> Look up the deduced truth value for the "Last Person" identified in the final question. 7. <i>Formulate Answer:</i> If the "Last Person" tells the truth (deduced value is True), output "Yes". If the "Last Person" lies (deduced value is False), output "No".  &lt;/task_instruction&gt; </pre>

Table 24: Complete instruction induction output for BBA web\_of\_lies subtask, demonstrating how Audio-Induct handles logical reasoning tasks with different characteristics from MMAR.