IMPROVING ASR AND CALLSIGN DETECTION IN AIR TRAFFIC CONTROL SPEECH USING WHISPER PROMPTING

Jehan Joachim Daniel Piaget[a] Amrutha Prasad
Petr Motlicek

Idiap-RR-04-2025

JULY 2025

[a]EPFL

Improving ASR and Callsign Detection in Air Traffic Control Speech using Whisper Prompting

Robotic Semester Project

Author
Jehan Piaget

Supervised by
Amrutha Prasad

Professor
Dr. Petr Motlicek

Spring Semester 2025

Contents

1 Introduction

This report presents my semester project at EPFL, which was conducted in collaboration with the IDIAP Research Institute. This work focuses on building a robust and reproducible pipeline to perform Automatic Speech Recognition (ASR) and callsigns detection in the context of Air Traffic Control (ATC) communications. Given the noisy, accented, and high-stakes nature of ATC speech, the goal is to explore and improve ASR performance with existing models.

The work involves evaluating different ASR models on the ATCO2 dataset. Experiments are conducted on EPFL's High Performance Computing (HPC) clusters using Apptainer containers, and investigate the use of prompting techniques with the Whisper model to enhance transcription and callsign recognition accuracy.

The instructions and code to reproduce or extend the experiments can be found on the corresponding GitLab repository.

1.1 Motivation

Automatic Speech Recognition (ASR) in the domain of Air Traffic Control (ATC) presents unique challenges. ATC communications are high-stakes: controllers must coordinate multiple aircraft in real time, often under pressure. Additionally, this is a low-resource domain. It is difficult to collect large, diverse, annotated datasets due to privacy and security constraints.

The audio in ATC is domain-specific and semi-structured, following standard phraseology. It is also affected by factors such as background noise, strong accents, overlapping speech, etc. This makes the task of building reliable ASR systems extremely challenging.

There is interest in building virtual pilot agents [1] that can participate in air traffic control training simulations. These agents would be capable of understanding instructions and responding like real pilots, helping to reduce the workload of human instructors and improve the realism of training scenarios. For these systems to work correctly and be effective, ASR and named entity recognition (NER) systems are essential.

1.2 Project goals

The main objective of this project was to explore, compare, and improve the performance of ASR models in ATC speech using the ATCO2 dataset. Multiple models are benchmarked: A fine-tuned version of the XLSR-53, and Whisper models in various sizes and configurations, both fine-tuned and pre-trained only. The project also investigates the use of Whisper prompting techniques to enhance transcription quality and improve recognition of crucial entities, such as callsigns.

A key goal of the project was to ensure the reproducibility of the entire process. Due to the complexity of setting up and running ASR pipelines on large models in a high-performance environment, the experiments are done on EPFL's HPC Izar cluster infrastructure, and the system with Whisper is built using Apptainer containers. This enables future researchers or students to reuse and extend the setup efficiently.

2 Related work

Recent studies show that adapting ASR models to air traffic control speech is crucial to handle its specialized phraseology and noise. Fine-tuned Whisper on ATCO2 and ATCOSIM data achieved strong improvements in Word Error Rate (WER) [2]. Other work explores domain adaptation via prompting: [3] use prompt-conditioned fine-tuning to inject ATC-specific context, reducing WER on unseen ATC recordings. Keyword-guided prompting [4] biases the model towards expected phraseology and jargon like callsigns. Additionally, [5] improve callsign recognition by incorporating air-surveillance metadata to rescore ASR decoding lattices and bias the output toward plausible callsigns, while [6] use domain-specific language models to rescore N-best hypotheses, enhancing recognition of ATC phraseology. These approaches highlight the benefit of integrating domain knowledge to improve ASR and callsign recognition in ATC, whether by fine-tuning, rescoring, or by carefully designing the prompts.

3 Methodology

3.1 Dataset

The experiments in this project were conducted on the ATCO2 dataset, as part of the ATCO2 project [7]: a large-scale corpus of real-world ATC speech collected from various European airports. It captures live voice communications between air traffic controllers and pilots, providing realistic, high-stakes, and challenging audio data.

The dataset includes over 5'000 hours of unlabeled training data and two manually annotated test sets. A 4h test set, manually transcribed, and a 1h subset used in this project for evaluation, containing high-quality gold transcriptions and full NER tags.

ATC audio presents several challenges, with noisy data, overlapping speech, and diverse accents. Even though ATC speech is partly structured and follows a standardized phraseology, it is domain-specific and is underrepresented in classic ASR training data. Callsigns, in particular, are critical for understanding the utterance but are rarely seen in pre-trained models.

The dataset was preprocessed through a pipeline involving segmentation, noise filtering, speaker diarization, English language detection, etc. The 1h evaluation set, with high-quality annotations, enables robust benchmarking for ASR performance and downstream entity recognition tasks.

3.2 ASR models

This project investigates two strong ASR models: XLSR-53, and Whisper. Both models are pre-trained on large-scale datasets and used here in fine-tuned variants adapted to domain-specific ATC data.

3.2.1 XLSR-53

XLSR-53 [8] is an encoder-only model based on the wav2vec 2.0 model developed by Facebook AI [9]. Its architecture consists of two main components: a convolutional neural network (CNN) feature encoder that transforms raw audio waveforms into latent speech representations, followed by a Transformer network that captures long-range context across the input sequence.

It is pre-trained in a self-supervised way on more than 50,000 hours of unlabeled, multilingual speech data. During pre-training, parts of the encoded audio representations are masked, and the model learns to predict the masked segments from surrounding context.

For this project, a version of the XLSR-53, fine-tuned on 190 hours of supervised ATC speech data is used [1]. It allows the model to adapt to the domain-specific vocabulary, accents, and phraseology of ATC communications.

XLSR-53 serves as a strong baseline for evaluating the impact of Whisper's fine-tuning and prompting strategies in this specialized setting.

3.2.2 Whisper

Whisper is a sequence-to-sequence encoder-decoder model released by OpenAI [10]. It consists of a Transformer-based encoder processing log-Mel spectrograms inputs, and a Transformer decoder that generates transcriptions, similar in approach to models like GPT.

Unlike XLSR, Whisper is pre-trained in a supervised manner on a massive corpus of 680,000 hours of labeled audio, with multiple languages and diverse recording conditions. This makes it robust to noise, accents, and challenging environments.

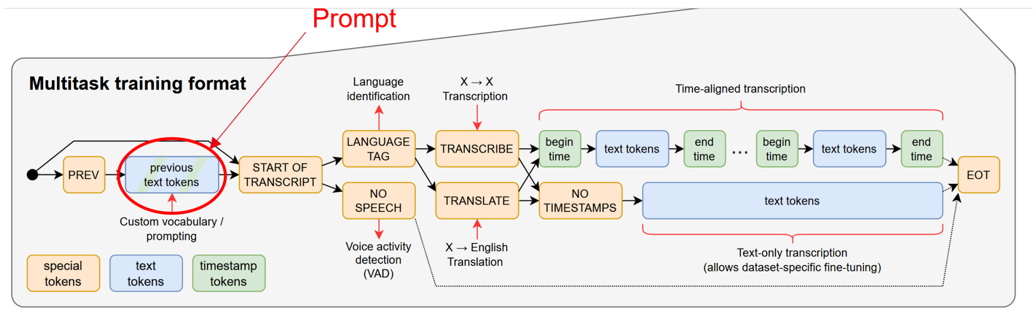

Figure 1: Whisper pipeline. Prompting step is circled in red [12]

In this project, original Whisper models in small, medium, and large (v2) configuration are evaluated. Moreover, a fine-tuned Whisper-medium model, trained on 190 hours of ATC speech (the same data as the XLSR fine-tuned model) using the Icefall toolkit [11], is included. It allows comparison of baseline performance with and without fine-tuning, and study the potential benefits of prompt-based conditioning in ATC speech recognition.

3.3 Prompting Whisper

The Whisper model supports prompting as a way to inject prior textual context into the transcription process, and bias the output using this information. This is done by providing an initial text segment, the **prompt**, which is tokenized and passed alongside the audio input. The decoder then attends to both the audio and the prompt during generation, through the attention mechanism of the Transformer, effectively biasing the transcriptions by adjusting the decoder's output probabilities according according to the provided prompt. Figure 1 shows the prompting in the pipeline of decoding.

In the project, prompting is used to inject domain-specific knowledge into the ASR process. For example, prompts include:

- Known callsigns: "oscar kilo charlie alfa papa", "sky travel four three two papa"
- Examples of ATC transcripts following standard phraseology, including the callsign: "charlie zulu searching hotel sierra for landing fato two five"
- Simple contextual information: "Air traffic control communication"

These techniques aim to help the model better recognize entities like callsigns, that are poorly represented in Whisper's pretraining data, and often misrecognized with non fine-tuned models.

One limitation of Whisper prompting is its fixed prompt length of 224 tokens [13, 14], restricting how much context can be injected, requiring to format prompts in a concise and efficient way.

3.4 Working environment

3.4.1 EPFL HPC clusters

All experiments in this project were conducted on EPFL's HPC Izar clusters. It provided the necessary compute capacity for evaluating large ASR models. The clusters offer access to high-memory CPU nodes and GPUs, and they rely on the SLURM workload manager for job scheduling and resource allocation.

Working in such an environment presents several challenges. As a shared-resource system, multiple users submit jobs concurrently, which can lead to queuing delays and limited availability of specific hardware configurations. Additionally, users do not have root or administrative privileges, which complicates the installation of system-level dependencies or configure the environment freely.

To adress these constraints and ensure reproducibility, the project adopts Apptainer containers.

Full details on the EPFL Izar hardware and usage can be found in the official documentation [15, 16].

3.4.2 Apptainer

Apptainer is a containerization tool designed specifically for High Performance Computing (HPC) environments. Unlike Docker, which requires root privileges to run and manage containers, Apptainer allows users to build and execute containers without administrative access, making it well-suited for shared computing environments like EPFL's HPC clusters.

An Apptainer container allows to package an entire software environment, including libraries, tools, scripts, and dependencies, into a single and portable image file. This enables experiments to run in a consistent and isolated environment, and can be run from any machine. It eliminates dependency conflicts and ensures reproducibility.

It consists of:

- **Image**: A template that defines the environment. For example, an image could include Python, Whisper, and all necessary libraries.
- **Container**: A running instance of an image. When launched, it behaves like a lightweight virtual environment where code can be executed with control over dependencies.
- **Volume**: Any file or folder can be mounted with the container, and used with the environment.

Apptainer definition files (.def) are used to specify the base image and installation steps. The container is built locally and requires the necessary files, folders, and models.

In this project, Apptainer was used to encapsulate all components required to run Whisper ASR model, including Python environments and libraries, toolkits like K2 and Icefall, the fine-tuned Whisper models, and the inference scripts. Only the normalization and evaluation tools for extracting callsigns and computing callsign accuracy and WER are mounted from outside the containers.

3.5 Experiments

All experiments are conducted on the ATCO2 1-hour test set, which includes high-quality gold transcriptions and named entity annotations. The focus is on assessing transcription quality, particularly for callsign recognition, using the evaluation metrics described in the next section.

The following five ASR models are evaluated: XLSR-53 fine-tuned, Whisper medium fine-tuned, and base small, medium, large-v2 Whisper models.

Prompting strategies are applied to the four Whisper models. Several strategies are explored:

- **Zero-shot**: No prompt provided. This serves as a baseline.
- **Domain prompt**: Context sentence "Air traffic control communication"
- **Transcription examples (1, 5, 15)**: Random gold ATC utterances from the test set, including callsigns and phraseology
- **Callsign prompt (15)**: Random list of callsigns from the test set.
- **Ground truth callsign**: The exact gold callsign is injected as prompt for each utterance decoding. Conversely to other methods, utterances are decoding with a batch size of 1, with a different prompt containing the exact callsign of the utterance at decoding time. Although it is not realistic for real-time systems, this serves as a potential upper-bound to evaluate the benefit of almost perfect prior knowledge.

These prompting strategies are designed to assess whether Whisper's decoder attention can leverage domain context to improve performance on challenging ATC data.

3.6 Evaluation

3.6.1 Normalization

Normalization is essential in ASR evaluation because it ensures that reference and hypothesis transcriptions from artificially increasing the measured error rate. In this project, normalization is applied to both the reference and hypothesis files. It first uses a list of known corrections for callsigns and abbreviations (e.g. airfrans $\rightarrow$ air france, air frans $\rightarrow$ air france). It standardizes number expression by converting digits to text (e.g. 360 $\rightarrow$ three six zero, in headings or degrees), while rendering altitude contexts as words (e.g. 4600ft $\rightarrow$ four thousand six hundred feet). It also handles contractions (e.g. it's $\rightarrow$ it is) and splits hyphenated and underscore forms to separate words.

Impact on the results is provided in the Appendix.

3.6.2 Metrics

The models and prompting methods are evaluated using the following metrics:

- **Word Error Rate (WER)**:
$$WER(\%) = \frac{S + D + I}{N} \cdot 100\%$$
where S, D, and I are the number of substitutions, deletions, and insertions respectively, and N is the number of words in the gold reference. This provides a general measure for the quality of transcription.
- **Callsign WER**: Same formula as WER, but computed only on the callsign segments, extracted from both the hypothesis and reference transcriptions. Focusing on callsigns only is critical for evaluating the ability of hte model to recognize key entities reliably, which are essential in the ATC domain.
- **Callsign Accuracy**: The percentage of utterances where the entire callsign is predicted correctly. This requires an **exact match** between the predicted and reference callsign after normalizing both to the International Civil Aviation Organization (ICAO) format (e.g., "hotel bravo kilo echo x-ray" $\rightarrow$ HBKEX) This metric is crucial for callsign precision, essential for downstream applications like virtual pilot agents or automatic flight tracking, to ensure the system can be trusted.

4 Results

In this section, we discuss the evaluation of ASR models and prompting strategies. Overall trends in WER, callsign WER, and callsign accuracy are reported. Detailed result tables are provided in the Appendix.

For the **Global WER**, we observe the following:

- Both fine-tuned models, XLSR and Whisper outperform original ones. Without prompting, the XLSR and Whisper (medium) fine-tuned achieve a substantially lower WER of 20.3% and 17.8% respectively compared to the 44.6% of the base medium Whisper, confirming the importance of domain adaptation.
- Prompting improves performance for all the models. The best performance (except for ground-truth "cheating" decoding) is achieved providing 15 transcripts examples, leading to a WER of 15.9%. We observe that prompting leads mainly to a huge difference on original models. The best prompting method (providing 15 trancripts examples) for small, medium, and large models help reduce by 17.6%, 14.2%, and 13.3% respectively. Even providing a single example of transcript helps models to reduce by 11.4%, 4.4%, and 2.9%. This highlights the importance of examples for original models, as they have never seen this kind of data and are clearly not adapted.
- Richer prompts yield better gains. Performance consistently improves with more examples in the prompt (1, 5, 15). These provide more domain context and guide the model towards the correct phraseology.

- Domain-only prompt has limited effect. Simply specifying "Air traffic control communication" helps slightly, but far less than examples-based prompts. This is also due to how Whisper attends to prompts. Similarly, asking to transcribe is not advised. The model is more focused on the style of the prompt, and the attention mechanism. As a result, examples are better to improve performance than global context or description.
- Providing ground-truth transcripts is an upper bound for the fine-tuned model. For the other models, providing 15 transcripts is as important as providing the ground truth callsign to reduce the WER.

For the **callsign WER**:

- Fine-tuned models again outperform original ones.
- Prompt size matters. Increasing from 1 to 15 transcripts examples reduces callsign WER for all the models.
- 15 transcripts vs 15 callsigns. Interestingly, providing 15 full transcripts performs about the same as providing 15 callsign-only prompts.
- Ground-truth callsign prompts yields best results, but are again unrealistic.

For the **callsign accuracy**:

- Accuracy is generally low. Even best realistic configuration achieves 42.4% accuracy. This highlights the challenge of correctly recognizing callsigns.
- Low WER does not guarantee high accuracy. Even a small mistake in 6-word callsign results in 0 accuracy for that utterance.
- Prompting improves accuracy. As with WER, more prompt context yields better accuracy, but gains remain limited.
- Ground-truth callsign prompts upperbound is relatively low. Even when feeding the exact callsign, accuracy gains are not huge, showing the difficulty of the task.

To summarize, across all metrics, results show:

- Fine-tuning is essential for strong performance on ATC speech.
- Prompting is an effective and lightweight domain adaptation strategy, with a bigger impact for non-fine-tuned models. Even minimal context helps significantly.
- More context is better. Prompting with many examples (15 transcripts) gives the best realistic improvements.
- Callsign recognition remains difficult. Even with low callsign WER, accuracy remains low.
- Prompting is extremely sensitive. Small changes in prompt design (e.g. punctuation, other callsign examples) can shift results by a few percent. This indicates the need for careful prompt engineering.

5 Conclusion

This project addressed the challenge of robust ASR and callsign detection in ATC communications. The first contribution was a systematic benchmarking of multiple ASR models, including fine-tuned versions of XLSR-53 and Whisper, on the ATCO2 dataset. It provided clear insights into the strengths and weaknesses of different models and the importance of fine-tuning for ATC speech.

A second contribution was the exploration of prompt-based conditioning with Whisper, demonstrating that injecting relevant in-domain phraseology or known callsigns can improve transcriptions, notably for models that have not been fine-tuned on ATC data. This shows the advantage of prompting as a lightweight adaptation technique for domain-specific ASR tasks.

Finally, a significant outcome of the project was the development of a fully portable, reproducible environment for Whisper experimentation on EPFL's HPC clusters. Using Apptainer containers, the entire pipeline, including models, dependencies, data and scripts, can easily be deployed in a shared environment without root access. This ensures that future researchers and students can reproduce and build on this work with minimal setup overhead.

5.1 Challenges & Limitations

Despite the progress made in this project, several limitations remain:

- **Prompt length constraints**: Whisper's prompting mechanism has a strict limit of 224 tokens. This restricts the amount of context that can be provided for decoding and makes prompt design sensitive and challenging.
- **Challenging speech data**: As shown by the results, ATC speech remains a highly difficult domain for ASR. Heavy noise, strong accents, and specific phraseology lead to frequent errors in the transcriptions, even with performant models.
- **Complex engineering setup**: Integrating the models, toolkits, scoring frameworks, and the HPC cluster environment was difficult and required a significant effort. Future work can now easily build on this project.
- **HPC resource constraints**: The shared nature of clusters led to practical limitations. Queuing delays, limited node availability, and numerous students working on GPU resources slowed experimentation and number of iterations.

These limitations highlight the complexity of building a clean, reproducible, and performant ASR research pipeline.

5.2 Future work

Several directions could be explored to extend this work:

- **Prompt design and optimization**: Initial experiments showed that prompting can improve performance. Future work could explore different prompt formulations, lengths, and context to find the most effective strategies for adaptation to ATC data.
- **Beam rescoring**: The current setup uses beam search decoding with Whisper and selects the top hypothesis directly. Incorporating context for rescoring could improve transcription quality.
- **Dynamic, contextual prompts**: Since the project already implements ground-truth prompting for each utterance, another direction is to generate dynamic prompts based on available metadata. For instance, providing a list of likely callsigns based on aircraft in the area of the control tower at decoding time. This could make the system more realistic and effective in operational settings.

Acknowledgements

I would like to thank Amrutha for her time, the multitude of questions she has answered, and the ideas she has suggested, as well as Petr for his trust and guidance on the project.

References

[1] Juan Zuluaga-Gomez, Amrutha Prasad, Iuliia Nigmatulina, Petr Motlicek, and Matthias Kleinert. A virtual simulation-pilot agent for training of air traffic controllers, 2023.

[2] Jan van Doorn, Junzi Sun, J. M. Hoekstra, Patrick Jonk, and Vincent de Vries. Whisper-atc: Open models for air traffic control automatic speech recognition with accuracy. In *Proceedings of the International Conference on Research in Air Transportation (ICRAT)*, 2024.

[3] Feng-Ting Liao, Yung-Chieh Chan, Yi-Chang Chen, Chan-Jan Hsu, and Da shan Shiu. Zero-shot domain-sensitive speech recognition with prompt-conditioning fine-tuning, 2023.

[4] Aviv Shamsian, Aviv Navon, Neta Glazer, Gill Hetz, and Joseph Keshet. Keyword-guided adaptation of automatic speech recognition, 2024.

[5] Juan Zuluaga-Gomez, Iuliia Nigmatulina, Amrutha Prasad, Petr Motlicek, Karel Veselý, Martin Kocour, and Igor Szöke. Contextual semi-supervised learning: An approach to leverage air-surveillance and untranscribed atc data in asr systems, 2021.

[6] Mrinmoy Bhattacharjee, Iuliia Nigmatulina, Amrutha Prasad, Pradeep Rangappa, Srikanth Madikeri, Petr Motlicek, Hartmut Helmke, and Matthias Kleinert. Contextual biasing methods for improving rare word detection in automatic speech recognition. ICASSP 2024, 2024.

[7] Juan Zuluaga-Gomez, Karel Veselý, Igor Szöke, Alexander Blatt, Petr Motlicek, Martin Kocour, Mickael Rigault, Khalid Choukri, Amrutha Prasad, Seyyed Saeed Sarfjoo, Iuliia Nigmatulina, Claudia Cevenini, Pavel Kolčárek, Allan Tart, Jan Černocký, and Dietrich Klakow. Atco2 corpus: A large-scale dataset for research on automatic speech recognition and natural language understanding of air traffic control communications, 2023.

[8] Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. Unsupervised cross-lingual representation learning for speech recognition, 2020.

[9] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020.

[10] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022.

[11] k2-fsa contributors. Icefall: Speech recognition toolkit using k2-fsa and lhotse. `https://github.com/k2-fsa/icefall`, 2024. Accessed: 2025-06-27.

[12] David Cochard (Medium). Prompt engineering in whisper.

[13] Preston Tuggle. Whisper prompting guide. OpenAI Cookbook, 2023.

[14] OpenAI. Speech to text: Prompting. `https://platform.openai.com/docs/guides/speech-to-text/prompting`, 2024. Accessed: 2025-06-27.

[15] EPFL SCITAS. Izar cluster hardware and configuration. `https://www.epfl.ch/research/facilities/scitas/hardware/izar/`, 2025. Accessed: 2025-06-20.

[16] EPFL SCITAS. Scitas user documentation. `https://scitas-doc.epfl.ch`, 2025. Accessed: 2025-06-20.

Appendix: Complete Results

Prompting Abbreviations:

N = No prompt
GT = Ground truth, provide callsign ground truth at each utterance
AC = List of callsigns (not necessarily from test set, up to 224 tokens)
C = Contextualize simply with "Air traffic control communications"
OS = Single example of transcript from test set
FS = Few examples (5) of transcripts from test set
RT = Realistic transcriptions (15 transcripts from test set)
RC = Realistic callsigns (15 callsigns from test set)

Table 1: Global Word Error Rate (WER, %)

Model	N	GT	AC	C	OS	FS	RT	RC
XLSR	20.3							
W-medium-ft	17.8	14.7	17.3	17.9	17.6	16.4	15.9	16.2
W-small	78.6	59.0	69.4	71.2	67.2	60.0	61.0	63.9
W-medium	44.6	31.5	36.1	45.0	40.2	33.0	30.4	32.7
W-large_v2	43.8	32.1	37.7	41.3	40.9	35.4	30.5	33.9

Table 2: Callsign Word Error Rate (WER, %)

Model	N	GT	AC	C	OS	FS	RT	RC
XLSR	21.8							
W-medium-ft	18.3	9.1	18.6	19.2	18.3	16.9	14.7	14.8
W-small	85.3	50.0	67.4	83.8	79.1	66.9	60.1	60.6
W-medium	60.5	19.3	42.1	60.9	53.9	43.0	38.0	36.4
W-large_v2	58.5	34.7	46.5	55.3	55.2	44.7	37.3	37.2

Table 3: Callsign Accuracy (%)

Model	N	GT	AC	C	OS	FS	RT	RC
XLSR	38.6							
W-medium-ft	39.2	46.0	39.0	38.6	39.5	40.0	42.5	42.0
W-small	13.1	29.3	11.7	13.3	13.1	14.3	18.3	16.5
W-medium	19.6	44.0	22.2	19.1	18.2	21.9	24.4	24.3
W-large_v2	20.9	31.3	25.0	20.6	20.9	24.1	28.9	26.6

Table 4: Comparison of Global WER and Callsign WER (%) for **non-normalized vs normalized** hypotheses across N, GT, and RT prompting methods, evaluated on the Whisper-medium fine-tuned model.

Metric	N	GT	RT
Global WER (non-normalized)	20.4	16.9	18.4
Global WER (normalized)	17.8	14.7	15.9
Callsign WER (non-normalized)	22.0	12.2	17.9
Callsign WER (normalized)	18.3	9.1	14.7