Structured Command Extraction from Air Traffic Control Communications Using Large Language Models

MSc Thesis Aerospace Engineering

Ana Maria Mekerishvili

TU Delft

nlr

Structured Command Extraction from Air Traffic Control Communications Using Large Language Models

MSc Thesis Aerospace Engineering

by

Ana Maria Mekerishvili

to obtain the degree of Master of Science
at Delft University of Technology
to be defended publicly on September 15, 2025 at 13:00

Thesis committee:

Chair:	Dr. F. Yin
Supervisors:	Dr. J. Sun (TU Delft) & P. Jonk (NLR)
External examiner:	Dr. M. J. Ribeiro
External member:	V. De Vries (NLR)
Place:	Faculty of Aerospace Engineering, TU Delft
Date:	6 August, 2025
Thesis Duration:	January, 2025 - September, 2025
Student number:	5322065

Cover picture by Gorodenkoff, sourced from stock.adobe.com

Delft University of Technology · Royal Netherlands Aerospace Centre (NLR)


TU Delft
Delft
University of
Technology


nlr

Contents

1

Scientific Paper

Structured Command Extraction from Air Traffic Control Communications Using Large Language Models

Ana Maria Mekerishvili
Faculty of Aerospace Engineering
Supervisors: Junzi Sun (TU Delft) and Patrick Jonk (NLR)

September 1, 2025

Abstract

Radiotelephony (RT) remains the primary medium for pilot-controller communication, yet extracting structured information from spoken exchanges is challenging. Deep learning approaches often depend on large annotated datasets, limiting use in data-scarce environments. This study evaluates open-source large language models (LLMs) for Structured Information Extraction (SIE) from ATC communications, with applications in assisting or automating pseudo-pilot tasks. We evaluate Llama 3.3 (70B) with baseline prompting and Gemma-3 (4B) with baseline and fine-tuned variants on 500 utterances from NLR's ATM simulator. Performance is assessed on human transcripts and ASR outputs from Whisper models, with varying prompt contexts. Cross-sector generalization is tested across two ATC sectors. Using manual scoring, Llama 3.3 achieves micro-F1 0.95 on human transcripts and 0.86 on fine-tuned Whisper outputs. While Gemma-3 performed weaker in its baseline form, fine-tuning on a small sample led to notable improvements. Results demonstrate the potential of LLMs for ATC applications without the need for large annotated datasets.

1 Introduction

The exchange of operational information between pilots and controllers is primarily conducted through voice-only radiotelephony (RT) communications. Automating structured information extraction from these exchanges supports applications such as performance assessment, safety monitoring, scenario analysis, and simulator training. In simulator environments, automatic parsing of Air Traffic Controller (ATCO) commands assists pseudo-pilots and potentially automates a subset of their tasks, which is particularly valuable given the typically high workload and need for specialized expertise [5]. Recent advances in large language models (LLMs) have opened new opportunities for such systems, as these models have been shown to generalize effectively from very limited data [1], [22]. Applying and evaluating these capabilities to the ATC domain requires a clearly defined target task, which in this work is referred to as Structured Information Extraction (SIE).

SIE entails extracting structured entities from ATC instructions according to an ontology adapted from SESAR PJ16-04 [9]. For each ATCO utterance, the parsed information includes the callsign, and for each instruction: category, command, value, unit, qualifier, and condition. Unlike traditional Natural Language Processing (NLP) tasks such as Named Entity Recognition (NER) and Slot Filling (SF), SIE must handle complete Air Traffic Control (ATC) commands and capture relationships between entities, including situations where multiple instructions appear in a single utterance.

Previous studies have researched using BERT-family encoders for NER and SF tasks in the ATC domain [13], [26]. BERT, or Bidirectional Encoder Representations from Transformers, is a transformer encoder pre-trained with masked language modeling and, in its original form, next-sentence prediction [4], [22]. It is typically fine-tuned on labeled data for downstream tasks. BERT-based approaches therefore have required domain-specific labeled datasets, a challenge in ATC where annotated open data is scarce [24], [26]. Moreover, previous studies have evaluated instruction parsing modules in isolation rather than in an end-to-end pipeline with Automatic Speech Recognition (ASR), even though this is how an SIE module would be deployed in practice for most of its applications. A high-level overview of the combined ASR and SIE pipeline is shown in Figure 1.

The aim of this study is to evaluate the ability of open-source LLMs to extract structured ATC commands from both human transcripts and ASR outputs, incorporating varying levels of contextual information. We examine: (i) few-shot prompting of LLMs on human

transcripts to accelerate labeling and estimate an upper bound on clean-text performance, and (ii) few-shot prompting of LLMs on ASR outputs to assess realistic end-to-end performance. Most experiments use a large model (Llama-3.3[1] [7]), and we further compare with a smaller architecture (Gemma-3[2] [20]), evaluated both in its baseline form and after fine-tuning on approximately 350 labeled samples. The effect of including different levels of contextual information in prompts is also examined.

The ASR component uses both a baseline and a fine-tuned Whisper model [6], [17]. Experimental data are drawn from NARSIM, NLR's real-time ATM simulator[3]. Performance is measured using F1 scores, and sector generalizability is assessed by evaluating results on two different sectors separately. Finally, a qualitative analysis of the LLM outputs is performed.

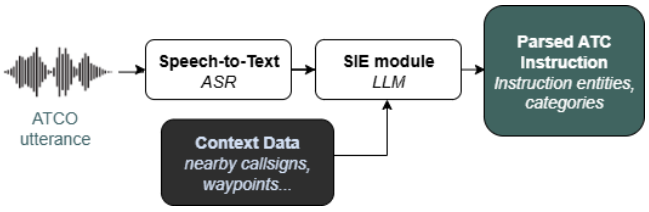

Figure 1: Simple overview of ASR and context-enhanced Structured Information Extraction from Air Traffic Controller utterances

2 Related Work

In the ATC domain, previous NLP studies have primarily focused on NER, i.e., detecting and classifying key elements such as callsigns, commands, and numerical values in transcripts. NER differs from structured information extraction (SIE), which aims to combine these entities into complete, structured commands, but it still provides valuable in-domain benchmarks for this work. BERT for NER was trained on the ATCO2 dataset, and most recent work has relied on deep learning approaches. Using four hours of labeled data with 5-fold cross-validation [26] resulted in mean F1 scores of 0.97, 0.82, and 0.87 for callsign, command, and value, respectively. An open-source BERT NER trained on the 1-hour open subset of ATCO2 reported an F1 of 0.66[4], illustrating BERT's sensitivity to training sample size.

Furthermore, another study training a BERT model for the NER task, using the same ontology as in this paper [9], showed that input data variability can affect F1. This study found that excluding utterances containing two or fewer entities from the training set, while reducing the training set size, improved performance, raising overall F1 from 0.80 using 2000 samples to 0.84 using 1400 samples [14].

Other recent relevant studies include a model using RoBERTa-Attention-BiLSTM-CRF as a more complex architecture, achieving an F1 of 0.88 on 9,200 manually labeled ATC transcriptions in Mandarin [2]. Another study focused on limited training data [12] proposed SLKIR, which extended a Transformer-XL backbone by inserting category "prompt" tokens (e.g., "Callsign") to guide a prompt classification head. The model was trained on 1,200 utterances from the commercial flight dataset and achieved an F1 of 0.85 on a commercial dataset and 0.79 on simulator training data, with each test set containing 200 utterances.

Moreover, a very recent study investigated a shallow Artificial Neural Network (ANN) with two hidden layers of 10 and 8 neurons using ReLU activation and dropout, followed by a softmax layer over eight intent classes to categorize instructions. This classifier was trained on a large, manually labeled corpus of ATC instruction transcripts, each annotated with one of eight intent categories (e.g., Radar Contact, Climb, Descend). They reported an F1 score of 0.98 for predicting the instruction category on ASR outputs, after post-processing and normalization. The ASR module used here was trained on a large dataset (including a private transcribed dataset of 32 hr) and had a Word Error Rate (WER) of 13.6% on the ATCOsim dataset [18].

Similar work outside the ATC domain has demonstrated that fine-tuning on relatively small samples (400–650 per dataset) and applying LLMs for SIE from scientific texts can be effective [3]. Another study introducing GPT-NER found that GPT-3 achieved performance comparable to BERT-based models on NER tasks. The paper also concluded GPT-NER to be advantageous in low-resource and few-shot settings, outperforming supervised baselines when labeled data was scarce [23].

As for the ASR module, end-to-end models have shown improved performance over previously used hybrid approaches in the ATC domain [25]. The open-source OpenAI Whisper model [17], a multi-layer transformer trained via large-scale weak supervision on a large corpus of diverse data, has also been tested on the ATC domain and fine-tuned. The fine-tuned Whisper model was found to achieve a lower WER than a previously tested Wav2Vec-fine-tuned model [6], [25].

Most prior work on ATC command extraction has focused on the NER task using BERT-family models, which achieved F1 scores ranging from 0.66 to 0.88 depending on the dataset and training conditions. The

[1] https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct
[2] https://huggingface.co/google/gemma-3-4b-it
[3] https://www.nlr.org/newsroom/facility/narsim/
[4] https://huggingface.co/Jzuluaga/bert-base-ner-atc-en-atco2-1h

ATCO2 benchmark remains among the strongest results, with F1 scores of 0.97 for callsigns, 0.82 for commands, and 0.87 for values, while other studies reported overall F1 scores in the 0.84–0.88 range. However, these approaches were found to be sensitive to the quantity and quality of labeled data and therefore rely on large, high-quality annotated corpora. In contrast, relatively little research has addressed end-to-end pipelines or scenarios with very limited data. To address this gap, our work evaluates open-source large language models for structured extraction from both human and ASR transcripts, using Whisper as the ASR component given its strong performance in the ATC domain and its availability as an open-source system.

3 Dataset

The dataset used in this research is collected from NARSIM during airspace restructuring[5] experiments. These simulations involve operational participants, including LVNL (Luchtverkeersleiding Nederland), the Dutch air navigation service provider, and MUAC (Maastricht Upper Area Control Centre), which manages upper airspace over parts of northwest Europe. The dataset includes recordings of controller–pilot exchanges on separate radio frequencies, as well as scenario metadata such as flight entry times, flight routes, radio frequency assignments, and airline designators.

Specifically, two frequencies are used in this study: one from LVNL's area control center and another from MUAC, responsible for the Delta sector. Table 1 summarizes the dataset used for evaluation (in the case of few-shot prompting) or for training and testing (in the case of fine-tuning Gemma-3). The audio segments from MUAC-Delta and LVNL-ACC1 are transcribed in approximately equal proportions. A separate dataset, LVNL-dev, is used exclusively for prompt engineering and is described in Table 2.

Table 1: Overview of the main NARSIM dataset number of instructions and the total duration of the transcribed audio segments per frequency

Dataset	Sector	Instructions	Duration
MUAC	Delta (Upper Area Control)	251	16 min
LVNL	ACC1 (Area Control)	263	15 min

[5]https://www.eurocontrol.int/press-release/optimising-airspace-above-netherlands-and-north-west-germany

Table 2: LVNL-dev set used for prompt design and validation, sampled separately from the main LVNL dataset with no shared audio segments between the two.

Dataset	Sector	Instructions	Duration
LVNL-dev	ACC1 (area control)	93	6 min

3.1 Transcription and Annotation

Audio segmentation is performed based on audio energy thresholds to isolate individual utterances. ATCO commands are transcribed by first running a Whisper model on the segments and then manually correcting them using Prodigy[6]. The audio recordings also include occasional communication between ATCOs (usually in Dutch), but these are not included in the dataset for this study. The utterances used contain exclusively English speech (with occasional Dutch phrases such as *goede dag*) and relevant simulator communications.

3.2 Context Data

Next to the recorded audio from NARSIM, complementary context data from the experiments is also used for this study. In particular, this includes the callsigns of all flights, along with their associated flight plans, consisting of scheduled entry times and route waypoints. The context data used in this study corresponds to scenario information that is typically available prior to the start of ATM simulations. No real-time or in-simulation incorporation of context data is performed. For some aircraft, the available route data is incomplete, meaning that certain utterances lack corresponding waypoint information. To address this, synthetic augmentation is applied to five utterances from the combined LVNL and MUAC datasets by inserting the missing waypoint into the list of possible waypoints at a random position.

4 Experimental Setup

All experiments are conducted on data obtained from NARSIM. The Structured Information Extraction (SIE) module is evaluated on both human and ASR transcriptions. A small development subset (LVNL-dev) is used solely to refine prompts, and performance on the LVNL-ACC1 and MUAC-Delta sets is not used for prompt iteration.

[6]https://prodi.gy/

4.1 ASR Module

We evaluate two ASR systems: the baseline Whisper `large-v3` [7] and a fine-tuned Whisper large-v2 model [6]. Performance of both these models is shown in Table 3. The word error rates (WER) shown in this table are calculated after normalization, including case folding, number normalization, and mapping NATO phonetic alphabet tokens to letters [6]. In some outputs of the fine-tuned model, a spurious fixed string occasionally appears at the start, which is easily removed in post-processing.

It is worth noting that, although the fine-tuned model is not trained on NARSIM data, its training set includes operational data from LVNL, which may explain its stronger performance on data from the simulated LVNL sector compared to the MUAC sector.

Table 3: Word error rate of each Whisper model on the LVNL-ACC1 and MUAC-Delta NARSIM sets.

Dataset	Whisper Baseline (large-v3)	Whisper Fine-tuned (large-v2)
LVNL	40.7%	12.2%
MUAC	28.3%	17.2%
Total	34.8%	14.6%

4.2 Structured Information Format and Ontology

The SIE module first identifies the commands in transcribed ATCO utterances, then determines their *category* and extracts command *entities* in a structured JSON format. An example of a SIE input-output pair is shown in Figure 2. The utterance in this example contains two instructions: a climb command and a heading command. The SIE returns the command category along with all instruction entities for each command. In this case, neither command includes a qualifier or condition, so the module returns *null* for these fields.

Command category, or *Instruction category*, can be regarded as a multi-class prediction task with labels *vertical command*, *horizontal command*, *speed command*, *changing frequency command*, or *other*, which is a subset of the categories mentioned in SESAR PJ16-04 [9]. An overview of the frequency at which each of these categories *appears* in the LVNL and MUAC datasets is shown in Figure 3. Overall, vertical commands (altitude changes, vertical speeds) *make up* 48.7% of all commands, followed by horizontal commands *making up* another 22.6% (direction commands, heading changes),

[7] https://huggingface.co/openai/whisper-large-v3

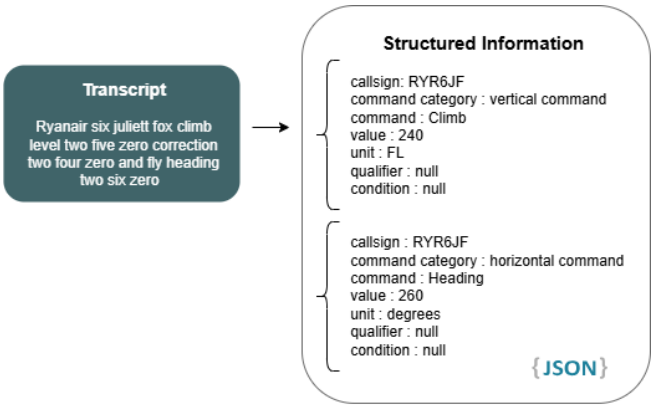

Figure 2: Example of structured information extraction (SIE) from a human-annotated ATC transcript.

frequency change commands 12.8%, and speed changes being the least frequently occurring category with only 4.5% of total commands. There *is* also a difference in distribution between the LVNL and MUAC datasets, with the LVNL dataset *having* more vertical and speed commands and MUAC *having* more frequency and horizontal commands. These differences likely *reflect* the operational roles of the respective sectors, with lower airspace *having* more climb and descent phases or commands, while upper airspace control *emphasizes* cruise-level coordination.

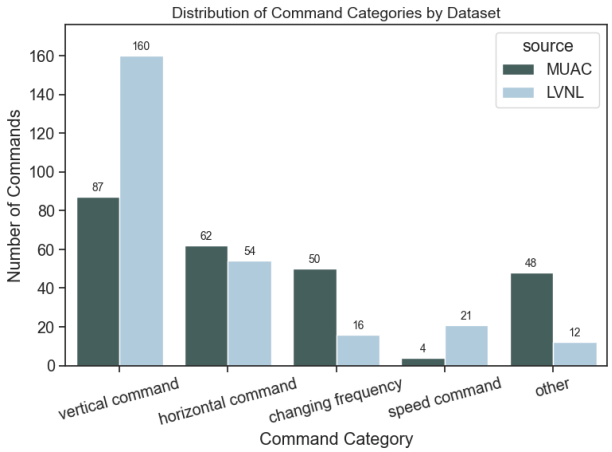

Figure 3: Instruction frequencies per sector by category.

Instruction entities follow the same ontology [9]: each instruction is parsed to *command* (type), *value*, *unit*, *qualifier*, and *condition*. Examples of these fields appear in Figure 4.

Callsigns are always extracted in their ICAO format [15] for standardization. For example, "KLM1999" may be spoken as "KLM one niner niner niner" or "KLM one triple niner," yet both map to the same ICAO string.

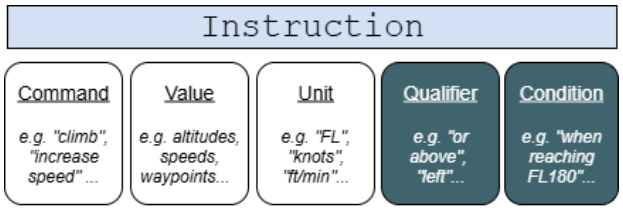

Figure 4: ATC instruction entities, based on ontology from SESAR PJ16-04 [9].

Not all instructions contain all entities. The prevalence of each field in the combined LVNL and MUAC datasets is shown in Figure 5. Callsigns and commands appear in every instruction, but not all of them include a value or unit, an example of such an instruction is "KLM123, continue climb." Furthermore, only a small fraction of commands include a qualifier or a condition.

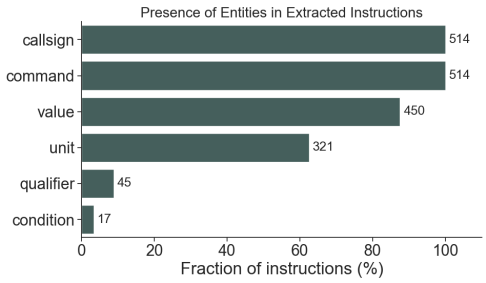

Figure 5: Occurrence frequency for each entity in the combined LVNL and MUAC dataset.

4.3 Prompt Design and Variants

LLM performance on information extraction is sensitive to how the prompt is formulated. Clear task description, consistent output formatting, and structured examples improve accuracy and reproducibility of the outputs [1]. Prompt development is guided by small-scale trials on LVNL-dev, which is excluded from final evaluation. All prompts share the same template:

- Brief description of the entities
- Desired JSON output specification
- Optional context information
- Five examples (retrieved exclusively from the LVNL-dev dataset)

We construct three prompt variants, differing in context information provided. They are summarized in Table 4. Each family is applied to both human and ASR transcriptions. ASR prompts include a brief note on potential transcription errors and use noisy examples reflecting typical artifacts, but are otherwise the same as the prompts used for human transcripts.

4.4 Inference and Hardware Specifications

We use 2xNVIDIA A100 and V100 GPUs. A100 GPUs are used to run the Llama 3.3 70B model, while V100s are used for Whisper inference and for running/fine-tuning Gemma-3. Unless stated otherwise, the decoding temperature for LLM inference is set to 0.0, in order to improve reproducibility and reduce the randomness of the outputs.

4.5 LLM Fine-tuning

Fine-tuning is applied to the smaller Gemma-3 (4B) model to test whether limited domain-specific data can boost performance, as its lightweight architecture makes it feasible to fine-tune within hardware constraints compared to the larger Llama 3.3. Gemma-3 (4B, instruction-tuned) is further fine-tuned using `Unsloth`[8] on input–output pairs constructed with the Callsigns-Only (CS) prompt *without* embedded examples. Five-fold cross-validation is performed on the combined LVNL-ACC1 and MUAC-Delta datasets with a split of 70/10/20 for training/validation/testing, respectively. For each fold, the model is trained for one epoch. Parameter-efficient fine-tuning (PEFT) via Low-Rank Adaptation (LoRA) is applied [8], [10]. A fixed random seed is selected to ensure reproducibility.

4.6 Evaluation Metrics

Manually annotated labels (JSON-formatted, with entities and categories) are used as reference for evaluating the performance of the SIE module. Since utterances may contain multiple commands, a greedy one-to-one matching algorithm is used to align human annotated and LLM predicted commands. For each human annotated command, all unmatched predictions are scored based on the number of identical fields after normalization (command category, command, value, unit, qualifier, condition, and callsign). Normalization consists of lowercasing and collapsing extra whitespace to ensure consistent string comparison. The prediction with the highest score is selected as the match, with each prediction used at most once.

For each matched command pair, entity fields are compared individually. A True Positive (TP) is counted when the manual annotation and the prediction contain the same value. A False Negative (FN) occurs when the human annotation specifies a value but the prediction either omits it or gives a different value. Conversely, a False Positive (FP) occurs when the prediction provides a value that is absent from the human annotation

[8] https://docs.unsloth.ai/

Table 4: Prompt variants and included context. All prompts also include five examples and JSON output format requirements.

Prompt family	Context included
No Context (NC)	Airline designators only, no nearby callsigns or any sector-specific information.
Callsigns Only (CS)	Nearby callsigns written in ICAO format (2-hour scenario window) and relevant airline designators only.
Additional Context (CX)	Two-stage prompting: extract callsign and command category, then add targeted context based on category (route waypoints for horizontal, sector frequencies for frequency change, plausible speed ranges for speed, sector altitude ranges for vertical).

or differs from it. Thus, FN reflects missing information relative to the human annotation, whereas FP reflects additional or incorrect information introduced by the prediction.

Unmatched human annotated commands contribute to FN counts for each present entity field, while unmatched predictions contribute to FP counts for each present entity field.

F1 and Micro-F1 Computation

We evaluate the command category assignment performance using the F1 score, defined as the harmonic mean of precision and recall:

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (1)$$

where Precision $= \frac{TP}{TP+FP}$ and Recall $= \frac{TP}{TP+FN}$.

The entity extraction task covers six structured entity fields: `callsign`, `command`, `value`, `unit`, `qualifier`, and `condition`. Micro-F1 is chosen as the primary metric because it reflects overall extraction accuracy by weighting each prediction equally, regardless of the entity type. This is particularly important in our setting, where the dataset is imbalanced (e.g., callsigns are much more frequent than conditions), and we aim to measure the model's aggregate performance across all entities. Micro-F1 is calculated by aggregating true positives (TP), false positives (FP), and false negatives (FN) across all six fields and all utterances within an evaluation split, as shown in Equation 2.

$$\text{Micro-F1} = \frac{2 \cdot TP_{\text{total}}}{2 \cdot TP_{\text{total}} + FP_{\text{total}} + FN_{\text{total}}} \quad (2)$$

Finally, entity frequency in this study is defined as the number of times a specific entity appears in human annotations (TP + FN).

5 Results

First, we evaluate the performance of SIE using the Llama model on different prompts and transcripts. Figure 6 summarizes micro-F1 scores across the three different prompt strategies and across human and Whisper transcripts. Performance increases consistently with both higher-quality transcripts and richer prompts. Moving from *No Context* (NC) to *Callsigns Only* (CS) yields gains of approximately 0.05-0.07 F1 across all transcript types, with the largest improvement observed for baseline Whisper transcriptions. Adding *Additional Context* (CX) provides a further but smaller improvement of 0-0.02, with the effect diminishing as transcription quality increases. When using human transcripts, the model reaches a ceiling of **0.91** micro-F1 under both CS (0.912) and CX (0.914) prompts.

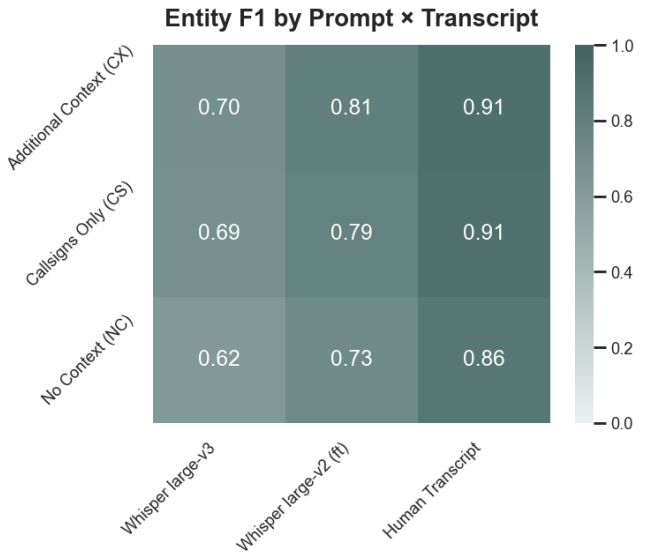

Figure 6: Micro-F1 by prompt family and transcript source. Each cell reports the micro-averaged F1 score for one prompt (rows) and transcription condition (columns).

Transcript quality seems to be more impactful than context-enhanced prompting: fine-tuned Whisper-v2

significantly outperforms Whisper-v3 by around 0.09–0.11 by micro-F1 scoring, with gold transcripts making an additional difference of 0.10–0.13. The impact of transcript quality on the SIE output can also be visualized in Figure 7, where the three data points corresponding to the three different transcripts indicate a strong correlation between the transcript WER and the SIE Micro-F1. This relation seems to be somewhat stronger when going from golden transcripts to fine-tuned Whisper outputs than going from fine-tuned to baseline Whisper. Furthermore, CS and CX prompts show a somewhat less steep decline in quality between human transcripts and baseline Whisper models, which can be explained by the fact that these prompts enhance the recognition of certain callsigns, waypoints, and values even when they are not transcribed fully correctly.

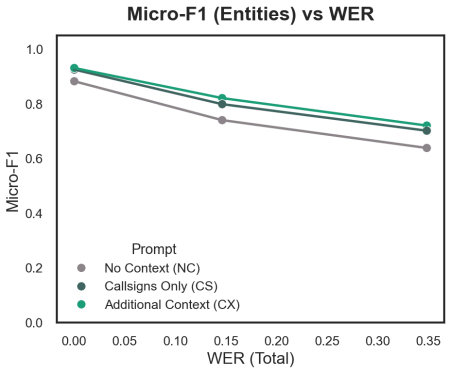

Figure 7: Effect of ASR WER on SIE micro-F1 per prompt family. The three data points correspond to gold transcriptions (WER assumed to be $\tilde{0}\%$), fine-tuned Whisper model (WER = 14.6%), and baseline Whisper model (WER = 34.8%)

We also evaluate the performance of the Llama 3.3 model per entity to understand how these micro-F1 scores are constructed. Table 5 shows the performance on human transcripts using the CS prompt. In general, the LLM output contains more false positives than false negatives, explained by the fact that the LLM overpredicts the number of commands present in the transcriptions. Performance is generally higher for frequent entities, such as unit (F1 = 0.96), callsign (F1 = 0.94), value (F1 = 0.94), and command category (F1 = 0.88). In contrast, rare entities perform worse: qualifier (F1 = 0.69) and especially condition (F1 = 0.38), where both false positives and false negatives outnumber true positives. This is explained by the fact that conditions tend to be more complex and longer than other entities, and they may have multiple ways of wording, making them less likely to be an exact string match with the human annotation. For qualifiers, false positives strongly dominate false negatives, even more so than for other entities, indicating that the LLM tends to overpredict what counts as a qualifier.

Table 5: Entity-level F1 scores for the Llama model using the *CS* prompt and human transcripts. Micro-F1 = 0.912

Entity	TP	FP	FN	Frequency	F1
callsign	489	40	25	514	0.94
command	460	72	54	514	0.88
value	426	27	24	450	0.94
unit	313	19	8	321	0.96
qualifier	37	25	8	45	0.69
condition	6	9	11	17	0.38

Sector Differences

Furthermore, sector generalizability of the Llama-based SIE is evaluated by calculating evaluation metrics on the LVNL and MUAC datasets separately, with the results displayed in Figure 8. It seems that although prompts are tailored according to the LVNL-dev dataset, only a difference of 0.03–0.04 in F1 score is observed on human transcription and baseline Whisper-large-v3 model performances across sectors. There is a larger discrepancy in the fine-tuned Whisper output performance, but this can also partially be attributed to the difference in WER between the sectors as shown in Table 3. The sector difference in the fine-tuned Whisper performance is also evident from the fact that callsign recognition is better for the MUAC-Delta sector than for LVNL-ACC1 in human transcriptions, but the opposite is true for the fine-tuned Whisper SIE, where there are likely more transcription errors in callsigns (see Tables 6 and 7).

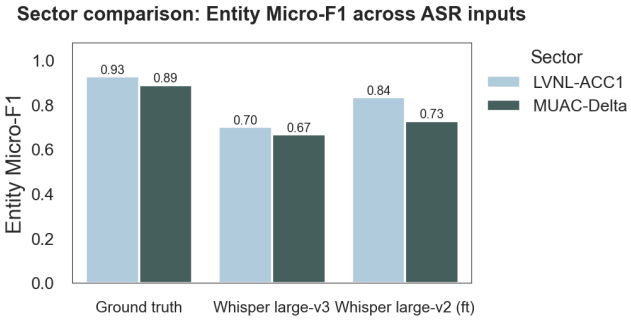

Figure 8: Entity micro-F1 scores per sector. CS prompt used.

Table 6: Entity-level micro-F1 scores by sector for the CS prompt and human transcripts.

Entity	LVNL-ACC1	MUAC-Delta
callsign	0.93	0.97
command	0.89	0.81
condition	0.67	0.00
qualifier	0.95	0.64
unit	0.97	0.94
value	0.94	0.93

Table 7: Entity-level micro-F1 scores by sector for the CS prompt and the fine-tuned Whisper transcripts.

Entity	LVNL-ACC1	MUAC-Delta
callsign	0.76	0.71
command	0.86	0.76
condition	0.42	0.00
qualifier	0.83	0.53
unit	0.95	0.92
value	0.82	0.66

Latency Evaluation

To evaluate the computational cost of the different prompting strategies, we measured inference latency using Llama 3.3 on two A100 GPUs. Table 8 reports the average prompt length (in tokens) and average inference time for the three schemes: no context (NC), callsigns only (CS), and additional context (CX).

The results show that NC yields the shortest prompts (828 tokens on average) and therefore the fastest inference time (2.6 s). In contrast, the CS scheme significantly increases prompt length (1956 tokens) and correspondingly latency (3.3 s). The CX scheme, which uses a two-step process, results in a combined length of 1560 tokens in the first step and 882 tokens in the second, with a total inference time of 3.8 s.

Overall, the differences of approximately 1–1.2 s between the simplest (NC) and most complex (CX) schemes are not prohibitive, but they highlight the trade-off between accuracy and latency when designing prompt structures for real-time ATC applications.

Table 8: Average input length (tokens) and total inference time for different prompt schemes using Llama 3.3 on two A100 GPUs. Tested schemes are No Context (NC), Callsigns Only (CS), and Additional Context (CX).

Prompt	Avg. Length (tokens)	Avg. Time (s)
NC	828	2.6
CS	1956	3.3
CX	1560 (step 1), 882 (step 2)	3.8

Gemma-3 (4B) Baseline Performance

To evaluate the effect of LLM model size and family, the Gemma-3 4B model is evaluated with the CS prompt on both human transcriptions and Whisper-large-v3 transcriptions. Gemma is a smaller model and, while it is newer than Llama 3.3, the expectation is that due to the difference in the number of parameters it still performs worse.

Both entity micro-F1 and category extraction F1 scores are shown and compared with Llama 3.3 in Table 9. It is evident that the Gemma model performs significantly worse than the Llama model on both category and entity extraction. Gemma-3 4B reaches an entity micro-F1 score of 0.70 under the CS prompt.

Table 9: Gemma vs Llama on CS prompt for two transcript sources.

Transcript	Metric	Gemma-3 4B (baseline)	Llama 3.3 70B
Human Transcript	Entity F1	0.70	0.91
	Category F1	0.80	0.98
Whisper large-v3	Entity F1	0.58	0.70
	Category F1	0.65	0.88

Gemma-3 (4B) fine-tuned Performance

While the performance of the baseline Gemma-3 model is lower than that of Llama 3.3, it has the advantage of being a lighter and faster model. This applies both to inference and fine-tuning. The effects of fine-tuning and testing this 4B model on the combined LVNL/MUAC dataset are shown in Table 10. It is evident that fine-tuning improves the entity extraction performance significantly, with the entity micro-F1 increasing after fine-tuning on fewer than 400 samples from 0.70 (baseline) to 0.81 (fine-tuned). On the other hand, the command classification does not perform significantly better. This suggests a steeper learning curve for the Gemma model to understand the task and what each extracted field entails, rather than understanding the broader context of the utterances.

Table 10: Performance of different LLMs on human transcripts, using the Callsigns Only prompt.

Model	Entity F1	Category F1
LLaMA 3.3	0.91	0.98
Gemma-3 (baseline)	0.70	0.80
Gemma-3 (fine-tuned)	0.81	0.84

6 Qualitative Error Analysis

While F1 scores provide a good overview of the SIE module's overall performance, it's also important to qualitatively examine the LLM outputs to understand the nature of the errors being made, not just their frequency. This is particularly relevant when considering real-world deployment, such as using the model to assist or automate tasks for a pseudo-pilot.

6.1 Beyond Exact String Matching

It is important to note that exact string matching tends to underestimate performance, as many predicted outputs are semantically equivalent to the human labels despite lexical differences (e.g., *"Climb"* vs. *"Climb to"*, *128.4* vs. *128.400*, *"maintain speed"* vs. *"keep speed"*, *"Amsterdam sector 2"* vs. *"Amsterdam sector two"*, *"call"* vs. *"contact"*).

Moreover, the boundaries between certain entities are not always clearly defined, even for human annotators. For example, an instruction like *"expedite your climb"* could be interpreted as a single command (`expedite climb`) or as a command with an associated value (`expedite`, `climb`). Such ambiguity is particularly common in the `qualifier` and `condition` fields, which partly explains why these fields consistently have the lowest accuracy and recall across experiments.

To better reflect semantic understanding, a subset of experiment outputs was manually re-scored with semantic matching in mind. In this evaluation, semantically equivalent entities and commands were counted as correct, resulting in higher micro-F1 scores. For CX on human transcriptions, the micro-F1 increased from 0.91 to 0.95 (see Table 11). For fine-tuned Whisper + CX, it increased from 0.81 to 0.86. As expected, per-entity scores, particularly for fields with more ambiguous definitions such or complex structure such as `qualifier` and `condition`, were also notably higher than in the exact matching evaluation (see Table 5).

Table 11: Field-level scores with manual semantic matching (CX on human transcripts).

Field	Precision	Recall	F1
command category	0.99	0.99	0.99
command	0.96	0.96	0.96
value	0.98	0.96	0.97
unit	0.98	0.98	0.98
qualifier	0.72	0.89	0.80
condition	0.67	0.59	0.62
callsign	0.96	0.96	0.96

6.2 Representative Error Types

Even after adjusting for semantically correct outputs that were penalized by exact string matching, errors remain in the SIE outputs, even for gold transcriptions. After a manual inspection, certain types of recurring errors and trends emerge which are described below.

A. Logical/format errors (numbers and corrections). The model occasionally misinterprets verbal numerals or fails to apply correction phrases as intended. These errors typically affect callsign and value extraction and are traceable to specific patterns in the input. Table 12 shows examples where the predicted callsign was incorrect despite the correct callsign being evident from the transcript, and the predicted one not appearing in the list of nearby callsigns. A mitigation strategy here would be to add an extra check ensuring the retrieved callsign is in the list of nearby callsigns.

Table 12: Logical handling of numerals and corrections.

Utterance	Human Label	Prediction
KLM one triple nine climb FL two one zero	CS=`KLM1999`	CS =`KLM139`
KLM eight eight, correction KLM one one eight eight, descend FL seven zero	CS =`KLM1188`	CS = `KLM88`

B. Hallucinations. Occasionally, the model generates values that are not supported by the input text, even when it seems to parse the rest of the instruction perfectly correctly. An example of this type of error is given in Table 13. This instruction is parsed completely correctly, except for the value (frequency). Although this was the only such hallucination to be observed in the CX prompt/human transcript outputs, this type of errors particularly problematic because they lack any apparent grounding in the input. These errors cannot be explained by transcription noise or misinterpretation, and therefore represent true model hallucinations.

Large language models are known to exhibit this behavior [11], [16]. While difficult to eliminate, some mitigation strategies include applying post-processing steps to LLM outputs: (i) checking if the output was mentioned in the utterance as well, (ii) lexicon/range validation for frequencies, headings, and flight levels, and (iii) a brief self-verification step that prompts the model to check and revise its own answer, which can reduce the rate of hallucinations significantly [21].

Table 13: Example of a hallucinated value.

Utterance	Human label	Prediction
Ryanair six juliett fox call Amsterdam one two three seven zero five	Val = 123.705	Val = 123.770 (altered)

C. Waypoints Or other Unfamiliar Words It was a curious observation that the waypoint *OMELO* was parsed incorrectly on all three occasions it appeared in the dataset, across all prompt families and ASR inputs. In the full-context prompt, it was rendered as *OMELLO* or *OMELo*, while in callsign-only prompts it appeared as *OMEL* or *OMELON*. This behavior is likely due to out-of-vocabulary (OOV) effects [19]: *OMELO* is a domain-specific term that likely does not appear in Llama's training data and is tokenized into rare or unfamiliar subword units, resulting in unstable output. These errors can be mitigated by applying a post-processing step that checks against a known list of valid waypoints.

D. Entity ambiguity. As mentioned earlier, the LLM struggled with some instructions that have a more ambiguous mapping to structured entities. A common example involves climb or descent instructions that also specify a vertical speed. These were sometimes parsed as two separate commands, and other times as a single `Climb`/`Descend` command with the vertical speed included as a `condition`. For instance, the instruction *"Sunexpress seven tango tango, descend flight level two six zero, two thousand feet per minute or greater"* could be parsed either as:

(i) two commands: a `Descend` command with value `FL260`, and a separate `Vertical Speed` command with value `2000 fpm or greater`, or

(ii) a single `Descend` command with value `FL260` and the vertical speed expressed as a `condition`.

Such inconsistencies are not critical and can likely be mitigated by incorporating clearer descriptions or representative examples into the prompt.

6.3 ASR Error Propagation

Finally, SIE output errors that can be traced back to ASR mistakes are considered. Table 14 presents examples of SIE errors that arise directly from transcription mistakes. Waypoints, in particular, tend to cause issues: they are often misrecognized by the ASR system, likely due to their rarity and absence from general training data. Even when the list of waypoints is provided as context in the prompt, the SIE module often fails to recover them correctly. In some cases, these errors also affect the extraction of other entities within the same utterance if they are transcribed incorrectly.

7 Study Limitations

This study has several limitations that may affect the generalizability of its findings. First, the dataset used was relatively small: around 500 annotated samples, which limits the statistical robustness of both the fine-tuning and evaluation. All annotations were created manually by the author. While every effort was made to ensure accuracy, this introduces potential for human error and subjective interpretation. Additionally, the dataset was limited to area control scenarios from the NARSIM simulation environment. As a result, findings may not generalize to other simulation setups or to different air traffic management (ATM) domains such as tower control.

Model training and evaluation were also constrained by hardware. Larger models such as Llama 3.3 could not be fine-tuned with the available hardware for this study. Furthermore, no extensive hyperparameter tuning was performed for Gemma-3.

Moreover, while prompting was used to improve the performance of the LLM-based SIE module, the ASR module (Whisper) was used without prompting or conditioning. This may have negatively impacted transcription quality and contributed to downstream errors. The study also only evaluated open-source LLMs due to confidentiality and accessibility constraints. As a result, stronger proprietary models, potentially better suited to the task, were not considered.

Finally, the prompts provided to the SIE module only included static context information, such as a broad list of nearby callsigns known in advance. In real-time ATC systems, where simulator metadata is available dynamically, more precise prompts could be constructed (e.g., with a much smaller list of plausible callsigns or routes), potentially improving both extraction precision and reducing the computational cost associated with large prompt lengths.

8 Conclusions

This thesis set out to evaluate the feasibility of using open-source large language models (LLMs) for structured information extraction (SIE) from air traffic control (ATC) communications, considering both gold-standard transcripts and automatic speech recognition (ASR) outputs. The findings show that accurate structured parsing is achievable with few-shot prompting, and that such systems can be built without reliance on large annotated datasets.

A central conclusion is that transcription quality appears to be the most influential factor for down-

Table 14: Representative ASR to SIE error propagation examples. Fine-tuned Whisper and CX prompt used

Reference utterance	ASR output	Downstream extraction error
”eight one november hello climb flight level ah two five zero”	”eight two november hello climb flight level ah two five zero”	CS = EWG82N instead of KLM81N (not in nearby list)
”KLM three three yankee hello climb to flight level two five zero”	”KLM three three yank hello cleared to fly at flight level two five zero”	Command = Cleared to instead of Climb to
”one zero hotel good day continue on the arrival landing runway one eight center”	”one zero hotel good day equal to you ahm only ryanair four landing runway will be one eight center”	Command = Landing runway, Callsign = RYR
”...resume own navigation direct RAVLO”	”...resume own navigation to the right hello”	Qualifier = to the right, waypoint missed
”...direct to BUREK”	”...direct to BUREQ”	Value = BUREQ instead of BUREK

stream SIE. Improvements in ASR word error rate tend to translate into better extraction accuracy, highlighting the value of continued development of domain-adapted ASR. Prompting with targeted context, especially nearby callsigns, generally yields improvements and can be regarded as a practical default. Additional context provides only modest benefit, except in noisier transcripts. Model size plays a secondary role: while Llama 3.3 achieves the strongest performance, compact models such as Gemma-3 can reach comparable accuracy when fine-tuned on a few hundred domain-specific examples, offering a potentially more resource-efficient alternative.

Taken together, these results provide evidence that an end-to-end speech-to-instruction pipeline can reach operationally meaningful performance. At the same time, qualitative analysis highlights recurring error types, over-prediction, difficulties with rare entities, and occasional hallucinations, that would need to be mitigated through verification steps before deployment. The discussion has shown that such measures, combined with latency-aware design and targeted prompting, could make LLM-based SIE suitable not only for research and training environments but also for integration into real-time pseudo-pilot systems.

The contributions of this work are threefold. First, it establishes a benchmark for open-source LLM-based SIE on both clean and noisy transcripts in ATC. Second, it demonstrates the effectiveness of lightweight prompting and fine-tuning strategies, pointing to clear trade-offs between accuracy, latency, and computational cost. Third, it identifies concrete directions for improving reliability, such as verification layers, dynamic context, and tighter coupling with ASR. Together, these findings position LLMs as a flexible and scalable foundation for structured ATC command extraction, with applications ranging from automated training support to future safety-monitoring tools.

9 Discussion and Future Work

While this study focused primarily on the structured extraction (SIE) performance of LLMs, practical deployment, particularly in real-time systems, requires consideration of computational cost and inference latency. Llama 3.3 (70B), used in this work for few-shot prompting, is a large and resource-intensive model. Few-shot prompting also necessitates longer inputs, further increasing inference time. In real-time applications such as an automated pseudo-pilot, where outputs must be generated promptly, these delays become significant. This is especially true when the SIE module is paired with an ASR system, as latency compounds across the pipeline. Although tools like Whisper-Live[9] or Whisper-Turbo[10] offer faster ASR, this often comes at the cost of higher word error rates.

On the output side, modern text-to-speech (TTS) tools are already fast enough and are unlikely to contribute further to latency bottlenecks. Our latency evaluation confirmed that more complex prompt schemes introduce additional delay, which, when compounded with ASR latency, could limit real-time applicability. In practice, prompts could be shortened by using targeted context, such as shorter callsign lists. Fine-tuning

[9] https://pypi.org/project/whisper-live/0.0.4/

[10] https://huggingface.co/openai/whisper-large-v3-turbo

a smaller LLM or using more advanced hardware can further help to reduce latency.

Furthermore, practical deployment would likely also have stringent requirements on accuracy. To mitigate hallucinations or spurious outputs from the LLM, several lightweight verification strategies can be added. These include self-verification through a second prompting step, verifying that extracted values actually occur in the utterance, or ensuring consistency with external knowledge such as a list of nearby callsigns, valid waypoints, or typical ranges for speeds and altitudes. Such checks can reduce the risk of false positives, often more harmful than null predictions in a safety-critical setting, though they may increase inference time. Furthermore, prompting the model to abstain (i.e., return a null output) when uncertain is a viable approach to improve reliability and reduce false positives, especially when the model is used in conjunction with imperfect ASR transcriptions.

Beyond structured extraction, large language models offer potential for broader use in pseudo-pilot systems. Their ability to maintain long context and flexibly handle diverse input makes them suitable for conversational or interactive behavior. While uncommon, controller-pilot interactions occasionally include clarification requests or negotiation of infeasible instructions, cases where traditional rule-based systems or fall short. LLMs could enable more dynamic and context-aware interaction in these scenarios.

In settings where real-time inference is not essential, LLMs can also be used to automatically generate labeled data for downstream tasks. For instance, they could be leveraged to create large-scale labelled training datasets for smaller, faster models, such as BERT-based classifiers or fine-tuned LLM variants. These models could then be used in real-time settings with significantly lower latency.

Finally, improvements in context use can enhance both performance and efficiency. In this study, context was static, based on pre-defined lists like nearby callsigns. However, real-time access to simulator metadata could enable dynamically targeted prompts, with shorter, more relevant lists. For example, narrowing the set of expected callsigns to those active in the sector could help the model make better predictions and reduce prompt size.

Future work should also explore combining LLM-based SIE with prompted ASR models. Since errors in callsign and waypoint transcription were found to be a major source of downstream mistakes, improving the ASR output itself, especially for domain-specific terminology, could yield substantial gains for the entire system.

References

[1] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[2] S. Chen, W. Pan, Y. Wang, S. Chen, and X. Wang, "Research on the Method of Air Traffic Control Instruction Keyword Extraction Based on the Roberta-Attention-BiLSTM-CRF Model," *Aerospace*, vol. 12, no. 5, 2025, ISSN: 2226-4310, doi: 10.3390/aerospace12050376.

[3] J. Dagdelen, A. Dunn, S. Lee, *et al.*, "Structured information extraction from scientific text with large language models," en, *Nature Communications*, vol. 15, no. 1, p. 1418, Feb. 2024, Publisher: Nature Publishing Group, ISSN: 2041-1723, doi: 10.1038/s41467-024-45563-x.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] K. Dönmez, S. Demirel, and M. Özdemir, "Handling the pseudo pilot assignment problem in air traffic control training by using NASA TLX," *Journal of Air Transport Management*, vol. 89, p. 101 934, Oct. 2020, ISSN: 0969-6997, doi: 10.1016/j.jairtraman.2020.101934.

[6] J.-W. van Doorn, J. Sun, J. M. Hoekstra, P. Jonk, and V. de Vries, "Whisper-ATC," en, 2024.

[7] A. Grattafiori, A. Dubey, A. Jauhri, *et al.*, *The Llama 3 Herd of Models*, arXiv:2407.21783 [cs], Nov. 2024, doi: 10.48550/arXiv.2407.21783.

[8] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, *Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey*, arXiv:2403.14608 [cs], Sep. 2024, doi: 10.48550/arXiv.2403.14608.

[9] H. Helmke, M. Slotty, M. Poiger, *et al.*, "Ontology for Transcription of ATC Speech Commands of SESAR 2020 Solution PJ.16-04," in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, ISSN: 2155-7209, Sep. 2018, pp. 1–10, doi: 10.1109/DASC.2018.8569238.

[10] E. J. Hu, Y. Shen, P. Wallis, *et al.*, *LoRA: Low-Rank Adaptation of Large Language Models*, arXiv:2106.09685 [cs], Oct. 2021, doi: 10.48550/arXiv.2106.09685.

[11] Z. Ji, N. Lee, R. Frieske, *et al.*, "Survey of Hallucination in Natural Language Generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, Dec. 2023, arXiv:2202.03629 [cs], ISSN: 0360-0300, 1557-7341, doi: 10.1145/3571730.

[12] P. Jiang, C. Zeng, W. Pan, B. Han, and J. Zhang, "SLKIR: A framework for extracting key information from air traffic control instructions Using small sample learning," en, *Scientific Reports*, vol. 14, no. 1, p. 9791, Apr. 2024, Publisher: Nature Publishing Group, ISSN: 2045-2322, doi: 10.1038/s41598-024-60675-6.

[13] J. Li, A. Sun, J. Han, and C. Li, "A Survey on Deep Learning for Named Entity Recognition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 50–70, Jan. 2022, arXiv:1812.09449 [cs], ISSN: 1041-4347, 1558-2191, 2326-3865, doi: 10.1109/TKDE.2020.2981314.

[14] A. K. Y. Low, L. Nimrod, S. Alam, and L. C. K. Poh, "Deep neural network-based automatic speech recognition for atc-pilot audio transcription," 2024.

[15] *Manual of radiotelephony*, 4th ed., Doc 9432 AN/925, International Civil Aviation Organization, Montréal, Canada, 2007, ISBN: 92-9194-996-5.

[16] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, *On Faithfulness and Factuality in Abstractive Summarization*, arXiv:2005.00661 [cs], May 2020, doi: 10.48550/arXiv.2005.00661.

[17] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, *Robust Speech Recognition via Large-Scale Weak Supervision*, arXiv:2212.04356 [eess], Dec. 2022, doi: 10.48550/arXiv.2212.04356.

[18] A. Sarhan, R. Fathy, and H. Ali, "Intelligent air traffic control using NLP-enhanced speech recognition and natural language generation," *Journal of Electrical Systems and Information Technology*, vol. 12, Jul. 2025, doi: 10.1186/s43067-025-00234-9.

[19] R. Sennrich, B. Haddow, and A. Birch, *Neural Machine Translation of Rare Words with Subword Units*, arXiv:1508.07909 [cs], Jun. 2016, doi: 10.48550/arXiv.1508.07909.

[20] G. Team, A. Kamath, J. Ferret, *et al.*, *Gemma 3 Technical Report*, arXiv:2503.19786 [cs], Mar. 2025, doi: 10.48550/arXiv.2503.19786.

[21] S. Tonmoy, S. Zaman, V. Jain, *et al.*, "A comprehensive survey of hallucination mitigation techniques in large language models," *arXiv preprint arXiv:2401.01313*, vol. 6, 2024.

[22] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[23] S. Wang, X. Sun, X. Li, *et al.*, *GPT-NER: Named Entity Recognition via Large Language Models*, arXiv:2304.10428 [cs], Oct. 2023, doi: 10.48550/arXiv.2304.10428.

[24] J. Zuluaga-Gomez, I. Nigmatulina, A. Prasad, *et al.*, "Lessons Learned in Transcribing 5000 h of Air Traffic Control Communications for Robust Automatic Speech Understanding," en, *Aerospace*, vol. 10, no. 10, p. 898, Oct. 2023, Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2226-4310, doi: 10.3390/aerospace10100898.

[25] J. Zuluaga-Gomez, A. Prasad, I. Nigmatulina, S. Sarfjoo, *et al.*, "How does pre-trained wav2vec2. 0 perform on domain shifted asr? an extensive benchmark on air traffic control communications," *IEEE Spoken Language Technology Workshop (SLT), Doha, Qatar*, 2022.

[26] J. Zuluaga-Gomez, K. Veselý, I. Szöke, *et al.*, *ATCO2 corpus: A Large-Scale Dataset for Research on Automatic Speech Recognition and Natural Language Understanding of Air Traffic Control Communications*, arXiv:2211.04054 [cs], Jun. 2023, doi: 10.48550/arXiv.2211.04054.

A

Supplementary Figures

This appendix presents supplementary figures that complement the Results section. All scoring follows the exact-match evaluation procedure described in Section 4 of the paper. No additional methods or datasets are introduced beyond those outlined in the main text.

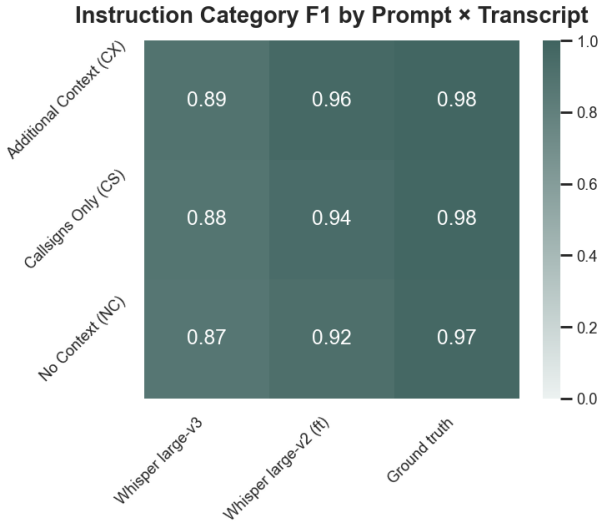

Figure A.1: Category extraction by prompt family and transcript source. Each cell reports the instruction category score for one prompt (rows) and transcription condition (columns), darker shades indicate better performance.

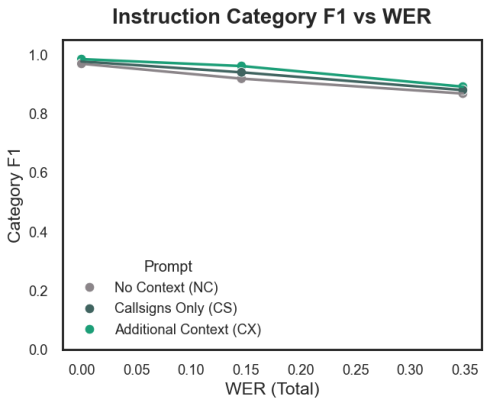

Figure A.2: Effect of ASR WER on instruction category F1 per prompt.

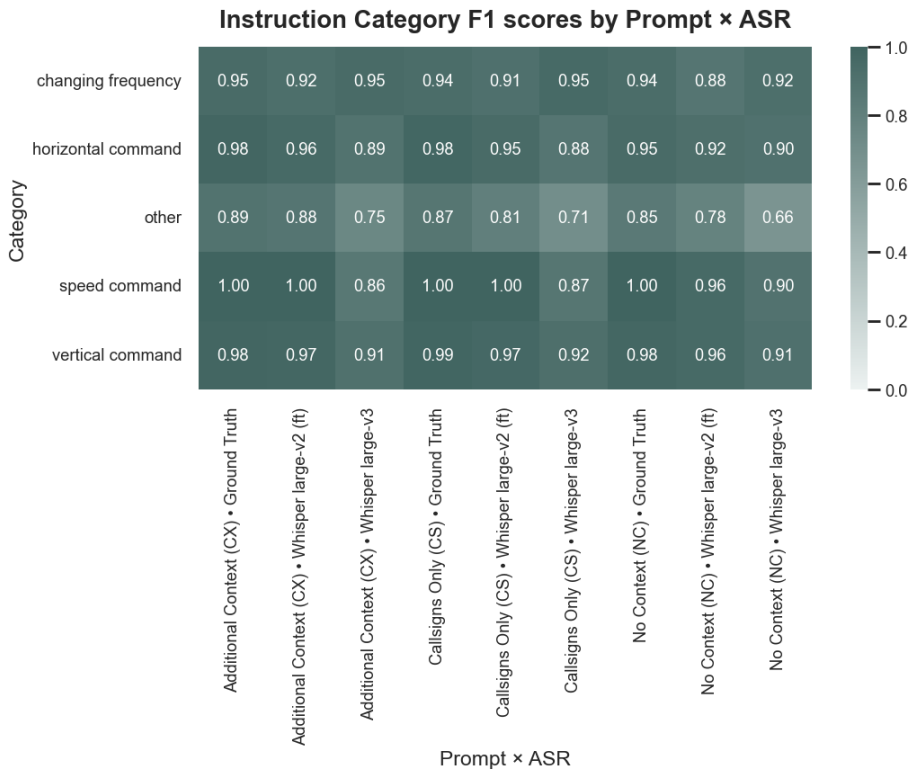

Figure A.3: F1 scores per instruction category by prompt schemes and trancsripts.

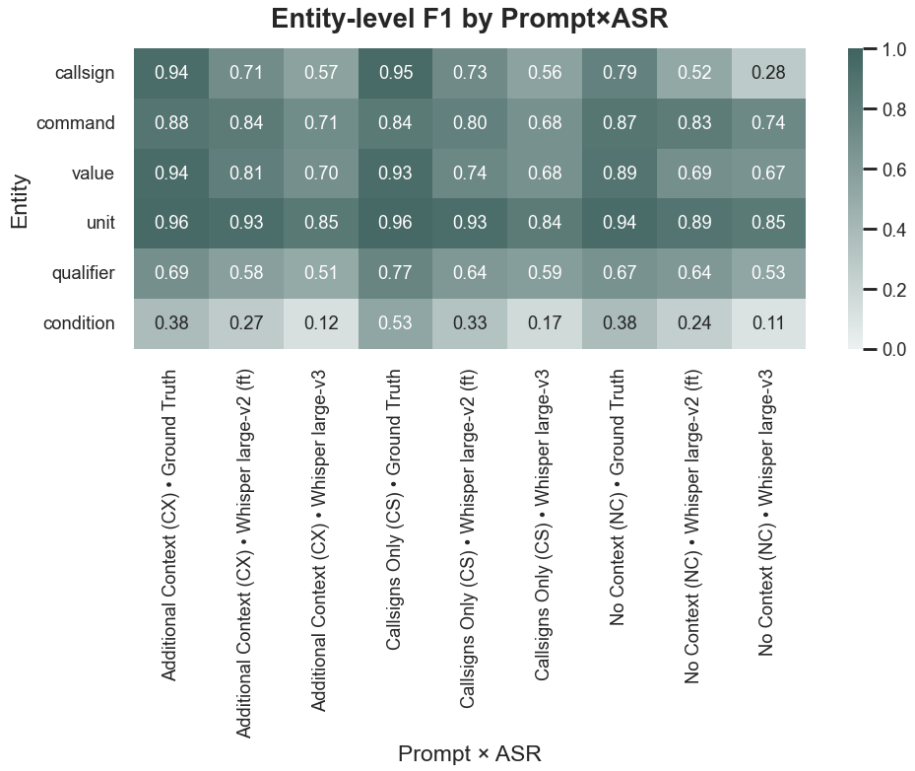

Figure A.4: Entity extraction accuracy per prompt and transcription input. Strict matching scores.

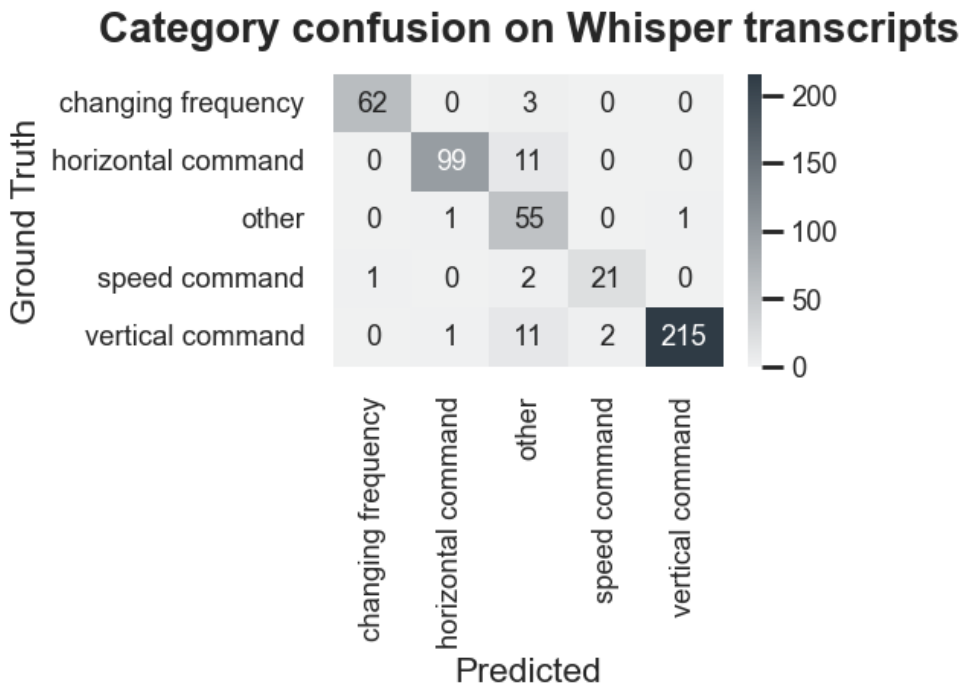

Figure A.5: Category classification errors per category, with CS prompt scheme on the baseline Whisper large-3 transcripts.

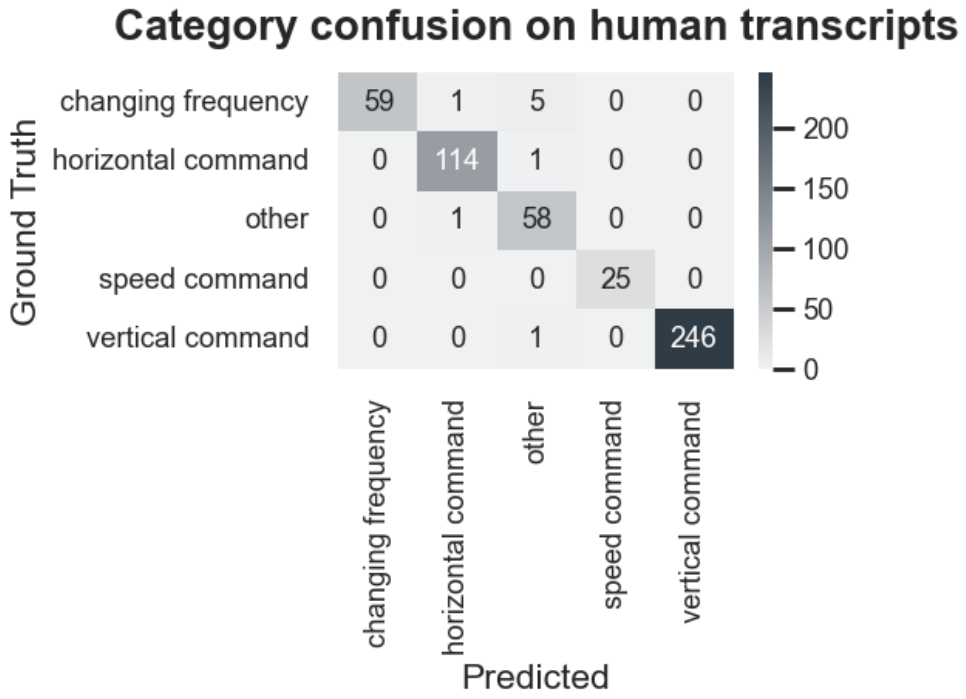

Figure A.6: Category classification errors per category, with CS prompt scheme on the human transcripts.

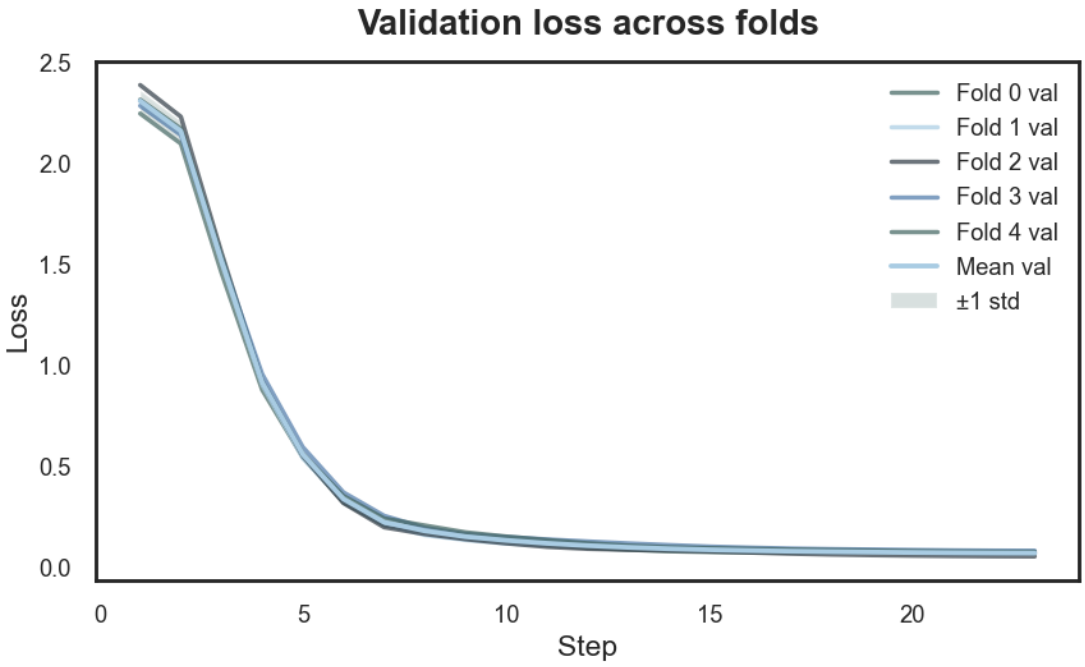

Figure A.7: Validation losses per fold for fine-tuning Gemma-3 (4B)

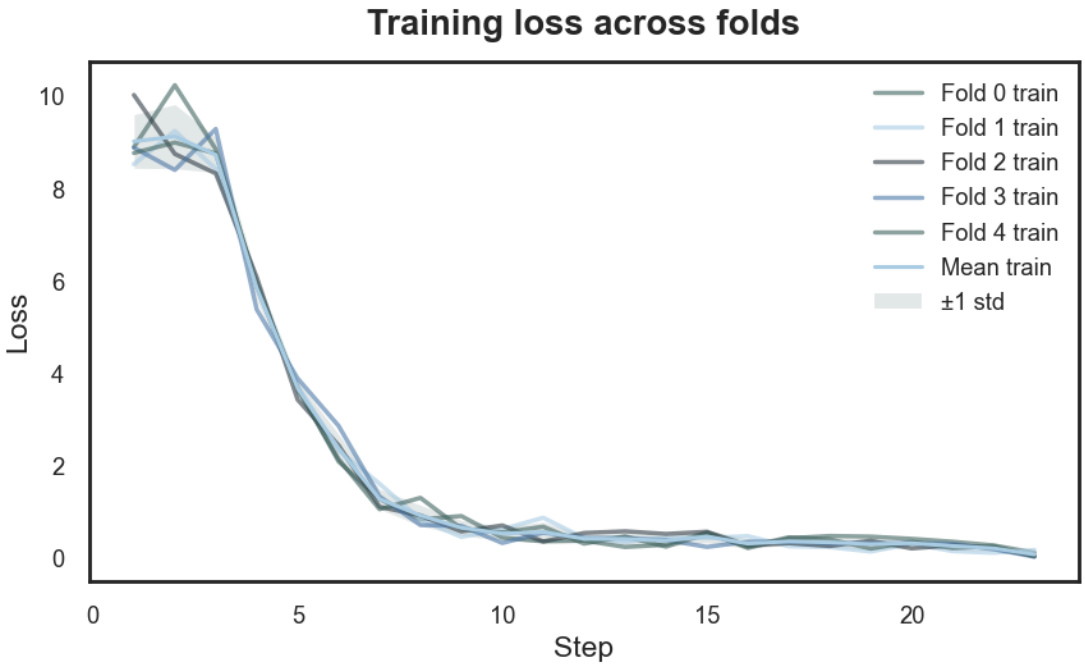

Figure A.8: Training losses per fold for fine-tuning Gemma-3 (4B)

Table A.1: 5-fold cross-validation scores for fine-tuned Gemma-3 (4B), with the indices indicating separate runs. Entity Micro-F1 scores are given.

	Entity Extraction F1	Command classification F1
0	0.843	0.836
1	0.782	0.829
2	0.781	0.826
3	0.822	0.867
4	0.799	0.827
mean	0.805	0.837

B

Finetuning Parameters

Unsloth framework is used for memory-efficient finetuning of Gemma 3[1].

```
model = FastModel.get_peft_model(
    model,
    finetune_vision_layers     = False,
    finetune_language_layers   = True,
    finetune_attention_modules = True,
    finetune_mlp_modules       = True,

    r = 8,
    lora_alpha = 8,
    lora_dropout = 0.00,
    bias = "none",
    random_state = 3407,
)
```

```
from trl import SFTTrainer, SFTConfig
trainer = SFTTrainer(
    model = model,
    tokenizer = tokenizer,
    train_dataset = df,
    eval_dataset = df_val,
    args = SFTConfig(
        dataset_text_field = "text",
        per_device_train_batch_size = 4,
        gradient_accumulation_steps = 4,
        warmup_steps = 5,
        num_train_epochs = 1,
        learning_rate = 2e-4,
        logging_steps = 1,
        optim = "adamw_8bit",
        weight_decay = 0.01,
        lr_scheduler_type = "linear",
        seed = 3407,
        eval_strategy = "steps",
        eval_steps = 1,
        report_to = "none",
    ),
)
```

[1] https://unsloth.ai/blog/gemma3

C

Prompts

Few-shot prompt used for ground truth transcriptions, aiding with a list of nearby callsigns

One example is removed from the prompts due to data confidentiality

```
    You are an ATC transcript parser.

Extract essential information from each instruction. For every instruction in the transcript,
    return a list of structured JSON objects with the following fields:

- "callsign": [in ICAO format, has to be matching one of the nearby callsigns provided]
- "command_category": [one of: "vertical command", "horizontal command", "speed command", "
    changing frequency", "other"]
- "command": [e.g., "Climb", "Descend", "Maintain", "Turn", "Contact", "Direct to", "Speed", etc
    .]
- "value": [e.g., "350", "180", "WOODY", "101225", or null]
- "unit": [e.g., "FL", "knots", "degrees", "MHz", or null]
- "qualifier": [e.g., "left", "or above", or null]
- "condition": [e.g., "when reaching FL200", "if able", or null]

Strict output rules:
- Output must be **valid JSON only**.
- Wrap output in a **JSON array**, even for a single instruction.
- Do **not** include Markdown formatting, comments, or explanations.
- Do **not** return line-separated outputs, just a single well-formed JSON array.

---

### Examples

#### Input:
Transcript: corendon eight lima echo good day climb flight level two five zero
Operator names: CAI: CORENDON, DAL: DELTA, KLM: KLM, RYR: RYANAIR, SAS: SCANDINAVIAN, TRA:
    TRANSAVIA
Callsigns nearby: CAI66JF, CAI8LE, DAL161, KLM1755, KLM7910, RYR8JF, SAS1555, TRA55L

json(s):
[
  {
    "callsign": "CAI8LE",
    "command_category": "vertical command",
    "command": "Climb",
    "value": "250",
    "unit": "FL",
    "qualifier": null,
```

```
    "condition": null
  }
]

---

#### Input:
Transcript: Ryanair six juliett fox climb level two five zero correction two four zero and fly
    heading two six zero
Operator names: CAI: CORENDON, CND: DUTCH CORENDON, DAL: DELTA, KLM: KLM, RYR: RYANAIR, SAS:
    SCANDINAVIAN, TRA: TRANSAVIA, WZZ: WIZZAIR
Callsigns nearby: CAI1129, CND6514, DAL161, KLM1755, KLM7910, RYR6JF, RYR8JE, SAS1555, TRA1223,
    WZZ943

json(s):
[
  {
    "callsign": "RYR6JF",
    "command_category": "vertical command",
    "command": "Climb",
    "value": "240",
    "unit": "FL",
    "qualifier": null,
    "condition": null
  },
  {
    "callsign": "RYR6JF",
    "command_category": "horizontal command",
    "command": "Heading",
    "value": "260",
    "unit": "degrees",
    "qualifier": null,
    "condition": null
  }
]

---

#### Input:
Transcript: KLM four one hotel speed two fifty or less
Operator names: CAI: CORENDON, DAL: DELTA, EJU: ALPINE, KLM: KLM, RYR: RYANAIR, SAS: SCANDINAVIAN
    , TRA: TRANSAVIA
Callsigns nearby: CAI66JF, CAI8LE, DAL161, KLM10H, KLM136E, KLM1755, KLM41H, KLM63J, KLM7910,
    RYR8JF, SAS1555, TRA55L

json(s):
[
  {
    "callsign": "KLM41H",
    "command_category": "speed command",
    "command": "Speed",
    "value": "250",
    "unit": "knots",
    "qualifier": "or less",
    "condition": null
  }
]

---

#### Input:
```

```
Transcript: two zero delta hello continue on the arrival runway is three six right
Operator names: CAI: CORENDON, CND: DUTCH CORENDON, DAL: DELTA, KLM: KLM, RYR: RYANAIR
Callsigns nearby: CAI1129, CAI66JF, CND6514, DAL161, KLM20D, KLM33Y, KLM7910, RYR8JF

json(s):
[
  {
    "callsign": "KLM20D",
    "command_category": "other",
    "command": "Continue on arrival",
    "value": "Runway 36R",
    "unit": null,
    "qualifier": null,
    "condition": null
  }
]

---

Now extract and return the instructions in strict JSON format from the following input:

% \end{verbatim}
```

Few-shot prompt used for ASR transcriptions, no additional context information needed

One example is removed from the prompts due to data confidentiality

```
You are an ATC transcript parser.

Extract essential information from each instruction. For every instruction in the transcript,
    return:

- "callsign": [in ICAO format]
- "command_category": [one of: "vertical command", "horizontal command", "speed command", "
    changing frequency", "other"]
- "command": [e.g., "Climb", "Descend", "Maintain", "Turn", "Contact", "Direct to", "Speed", etc
    .]
- "value": [e.g., "350", "180", "WOODY", "101225", or null]
- "unit": [e.g., "FL", "knots", "degrees", "MHz", or null]
- "qualifier": [e.g., "left", "or above", or null]
- "condition": [e.g., "when reaching FL200", "if able", or null]

Keep in mind that the transcript may have some automatic speech recognition errors in them.

Strict output rules:
- Output must be **valid JSON only**.
- Wrap output in a **JSON array**, even for a single instruction.
- Do **not** include Markdown formatting, comments, or explanations.
- Do **not** return line-separated outputs, just a single well-formed JSON array.

---

### Examples

#### Input:
Transcript: Corinne eight lima echo good day climb flight level two five zero

json(s):
[
```

```
  {
    "callsign": "CAI8LE",
    "command_category": "vertical command",
    "command": "Climb",
    "value": "250",
    "unit": "FL",
    "qualifier": null,
    "condition": null
  }
]

---

#### Input:
Transcript: rhein air six juliett fox climb level two five zero correction two four zero and fly
    heading two six zerra

json(s):
[
  {
    "callsign": "RYR6JF",
    "command_category": "vertical command",
    "command": "Climb",
    "value": "240",
    "unit": "FL",
    "qualifier": null,
    "condition": null
  },
  {
    "callsign": "RYR6JF",
    "command_category": "horizontal command",
    "command": "Heading",
    "value": "260",
    "unit": "degrees",
    "qualifier": null,
    "condition": null
  }
]

---

#### Input:
Transcript: KELOM for one hotel speed two five or less

json(s):
[
  {
    "callsign": "KLM41H",
    "command_category": "speed command",
    "command": "Speed",
    "value": "250",
    "unit": "knots",
    "qualifier": "or less",
    "condition": null
  }
]

---

#### Input:
Transcript: KLM eight one papa fly heading zero five five
```

```
json(s):
[
  {
    "callsign": "KLM81P",
    "command_category": "horizontal command",
    "command": "Heading",
    "value": "055",
    "unit": "degrees",
    "qualifier": null,
    "condition": null
  }
]

---

Now extract and return the instructions in strict JSON format from the following input:
```

D

Transcript Correction

Prior to SIE, research was conducted on improving ASR transcripts using LLMs. Three models were evaluated: GPT-4o, Llama-3.3, and Gemma-2, with the results shown in Figure D.1. ASR performance was assessed using Word Error Rate (WER) on a subset of the ATCO2 test dataset [1].

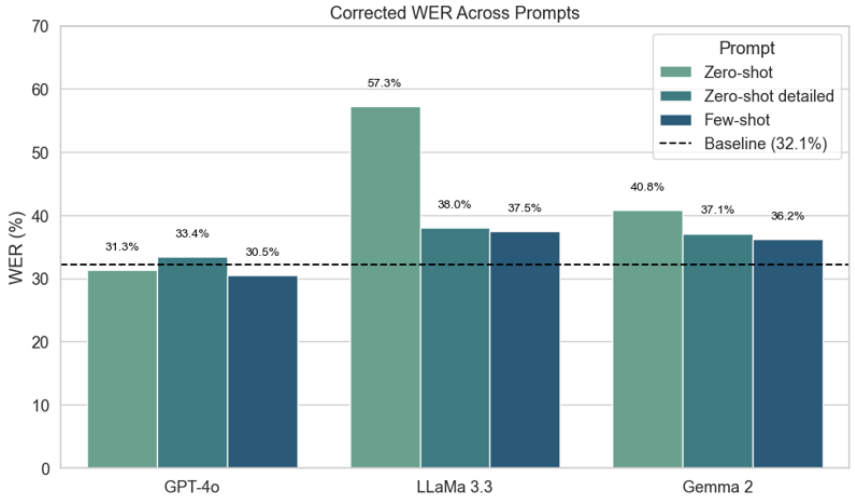

Figure D.1: Word Error Rate improvement on the ATCO2 test set (146 utterances)

To fine-tune the model, we applied a LoRA (Low-Rank Adaptation) approach on the ATCO2 training set, which contains approximately 550 transcribed air traffic control utterances. Training was limited to a single epoch and carried out on free T4 GPUs available through Kaggle, ensuring a cost-effective and accessible setup.

The fine-tuning procedure followed an instruction-tuning strategy in a zero-shot setting. Each training instance consisted of an instruction prompt, the corresponding Whisper-generated transcription, and the target output. Fine-tuning was performed with Hugging Face's Supervised Fine-Tuning (SFT) Trainer, which provided an efficient pipeline for adapting the baseline model. The baseline performance was 31.0%, and this setup established the foundation for measuring improvements achieved through fine-tuning.

Dataset	Baseline Gemma-2-9B	Finetuned Gemma-2-9B
ATC-Test Set	40.8%	38.2%

Table D.1: Performance on the ATC-TestSet before and after fine-tuning.

E

Initial Research Proposal

List of Figures

List of Tables

Nomenclature

List of Abbreviations

ACC	Area Control Center
AI	Artificial Intelligence
ANSP	Air Navigation Service Providers
APP	Approach Control
ASR	Automatic Speech Recognition
ASRU	Automatic Speech Recognition and Understanding
ATC	Air Traffic Control
ATCo	Air Traffic Controller
ATM	Air Traffic Management
AUC	Area Under Curve
CER	Command Error Rate
CPU	Central Processing Unit
CSER	Callsign Error Rate
CTA	Control Area
CTR	Control Zone
CUDA	Compute Unified Device Architecture
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
ICAO	International Civil Aviation Organization
KER	Keyword Error Rate
LLM	Large Language Model
LoRA	Low-rank Adaptation
NER	Named Entity Recognition
NLP	Natural Language Processing
ROC	Receiver Operatic Characteristic
SNR	Signal-to-Noise Ratio
TMA	Terminal Control Area
TWR	Tower Control
UTA	Upper Control Area
VAE	Variational Auto-Encoder
WER	Word Error Rate

Introduction

The application of automatic speech recognition (ASR) and natural language processing (NLP) in air traffic control (ATC) has significant potential to enhance safety and efficiency in the field. One key area is safety monitoring and incident detection, where ASR is employed to identify communication errors, such as readback mistakes, and detect anomalies in ATC-pilot exchanges. Projects such as the HAAWAII initiative have demonstrated the feasibility of both rule-based and machine learning approaches for identifying miscommunications [2]. Similarly, research is being conducted to explore methods for detecting abnormal, safety-critical situations within ATC transcripts [3].

Beyond safety applications, ASR also plays a crucial role in speech-to-text logging and incident investigation, enabling automated transcription of ATC communications to support post-incident analysis and risk assessment. Additionally, ASR contributes to ATC training through its potential to be integrated into simulation environments, facilitating automated pseudo-pilot interactions and improving post-training performance evaluations. Studies have shown that ASR-based training tools can reduce the costs associated with human pseudo-pilots by either assisting them with easily accessible information or potentially replacing them altogether [4, 5, 6, 7, 8]. These use cases underscore the importance of developing high-accuracy ASR models tailored to ATC communications, both for enhancing safety and lowering operational costs.

Despite recent breakthroughs in ASR technologies, speech-to-text transcription within ATC remains prone to errors and often requires manual verification to ensure sufficient accuracy. OpenAI's Whisper model has shown promising results in speech-to-text transcription for ATC, particularly when fine-tuned with domain-specific data [9]. However, due to the limited availability of large-scale annotated ATC datasets, accuracy remains a challenge, and human oversight is still necessary for many critical applications.

Recent breakthroughs in large language models (LLMs), driven by the rise of transformer architectures, increased data availability, and advancements in computational power, have opened new possibilities for improving ASR post-processing [10, 11]. LLMs are highly versatile and capable of processing contextual data while adhering to ICAO-standard phraseology and ATC communication protocols. This thesis aims to evaluate the use of LLMs in refining ASR-generated transcripts, with the goal of improving transcription accuracy and usability in operational ATC settings. It will do so by leveraging existing transcribed data, standard phraseology rules, and other contextual data such as radar information.

This report presents the research proposal for the study. Chapter Appendix E provides a review of relevant literature, including background on ASR, LLMs, ATC communications, and their use cases. The research objective and key research questions are then outlined in Appendix E, followed by a discussion of the project scope and timeline in Chapter 4.

Literature Review

This chapter aims to provide background information on the topics of the thesis. The literature study is divided in three main parts: applications and use cases of speech-to-text in ATC are discussed in Appendix E, backgroud in ASR is given in Appendix E, large language models (LLMs) in Equation E, and finally, a background in air traffic control is given in Equation E.

E.1. Use Cases of Speech Recognition and Understanding in ATC

First, an overview is provided of some of the use cases which make it useful to have a model or a pipeline to understand and process speech in ATC. It is important to be aware of different use cases, both to evaluate the utility in developing such models, but also to have an idea of different accuracy thresholds that are needed for the models to be useful. Some potential use cases are explored below, as well as background information on whether or not these have already been implemented.

Safety Monitoring & Incident Detection

ASR systems can be used to enhance safety in ATC. One approach to this is to compare the ATC instructions or what the pilot says, to actual real-time actions of the pilot to make sure they align. In most cases this would require multi-modal data (e.g. radar data) to compare with the communications, certain safety hazards can be flagged purely based on the communication as well.

One such example is checking in real time whether there are any incorrect readbacks. This can be done by comparing the statements from the the air traffic controller (ATCo) with the statements repeated from the pilot [12], [2].

The HAAWAII project aimed to improve ATC safety by developing a Readback Error Detection Assistant (REDA) using Automatic Speech Recognition and Understanding (ASRU) [2]. Readback errors, where pilots misrepeat ATC instructions, occur in 1–2% of transmissions, with 80

The project explored rule-based and machine-learning approaches. Rule-based methods extract key details using predefined rules, offering high accuracy when ASR quality is good but requiring manual updates. ML-based models, like RoBERTa, generalize better but demand large annotated datasets and are harder to interpret. A hybrid system combining both achieved the best results, detecting 81

However, ASR accuracy remained a challenge, with up to 10% WER for pilots. affecting REDA's overall performance. The study underscores REDA's potential to enhance ATC safety but calls for improvements in false alarm reduction, real-time processing, and ASRU integration into ATC workflows [2].

Another study by Fox et al. [3] looks into leveraging LLMs to detect anomalies (such as e.g. a pilot noticing something on fire or a sudden engine failure) in ATC communications. The study combined publicly available transcript datasets with synthetic LLM-generated data to train a Variational Auto-Encoder (VAE) to be able to recognize abnormal, safety-critical situations from communications. The study found promising results, achieving Area Under Receiver Operatic Characteristic curve (AUC-ROC) of 88.4% when processing entirety of conversational exchanges.

These studies provide good examples of transcriptions of ATC for safety analysis that only require transcripts as input, making a highly accurate ASR being a prerequisite for their utility. Other use cases in safety also include displaying transcriptions to pilots or ATCos to reduce chances of misscommunications to begin with.

Logging and Incident Investigation

ASR can play a crucial role in speech-to-text logging and incident investigation by transcribing and storing all ATC communications in real-time. This can ensure that every interaction between air traffic controllers and pilots is accurately documented, creating a searchable and time-stamped archive. In the event of an incident, safety investigators can quickly access past conversations, pinpoint specific exchanges, and

analyze communication patterns to understand what went wrong. By eliminating the reliance on manual audio reviews, ASR significantly speeds up investigations, allowing for a more efficient and data-driven approach to incident analysis.

Beyond immediate investigations, long-term transcription archives enable pattern analysis and systemic risk detection. Aviation authorities can examine historical records to identify trends in operational issues, such as repeated misunderstandings of clearance instructions, common phraseology inconsistencies, or frequent communication breakdowns in certain airspace regions. This data-driven approach helps refine ATC procedures, enhance training programs, and improve standard phraseology adherence, ultimately contributing to a more robust and error-resistant aviation communication framework. By maintaining comprehensive logs, the aviation industry ensures that lessons from past incidents are thoroughly analyzed and applied, reinforcing safety across the entire air traffic system.

Air Traffic Controller Training

ASR can have multiple other applications in controller training as well, and can be of use both in post-operation and real-time applications. For post-operation, ASR can be utilized for event recognition and logging, enabling a detailed assessment of controllers' performance after training sessions. By transcribing ATC communications, ASR facilitates the identification of areas for improvement, contributing to enhanced training outcomes. Studies have explored automatic transcription of ATC communications as a means to improve system safety and operational efficiency [4].

In real-time training scenarios, ASR supports the role of a pseudo-pilot: individuals who simulate pilot communications during controller training exercises. Integrating ASR into simulators allows for the automation of pseudo-pilot functions, leading to more efficient and realistic training environments. Research has demonstrated the development of virtual simulation-pilot agents capable of processing spoken communications from trainee controllers, generating appropriate pilot responses, and thereby reducing the need for human pseudo-pilots who tend to be quite expensive [5].

A study by Prasad et al. presents a potential pipeline for such an agent acting as a pseudo-pilot. The proposed pipeline consists of four modules: (i) an ASR module, (ii) a named entity recognition (NER) module to categorize ATC speech, (iii) a repetition generator to produce the pilot's response, and (iv) a text-to-speech module for verbalizing the response [6].

Their proof-of-concept study highlights that a well-performing ASR system is crucial for the model's effectiveness. They found that the NER module achieved promising results, with F1-scores above 0.80 on the ATCO2 dataset. For generating the pseudo-pilot's speech, they used a rule-based dialogue system to convert text into appropriate pilot responses, combined with an out-of-the-box text-to-speech model FastSpeech2 [7]. The model's output remained relatively simple, demonstrating proof-of-concept functionality.

Moreover, ASR technology has been applied to support simulation pilots during Human-in-the-Loop experiments by recognizing verbal clearances from air traffic controllers and forwarding the information to visual interfaces, enhancing the realism and effectiveness of ATC training simulations [8].

E.2. Automatic Speech Recognition

After considering the various applications of automatic speech recognition and processing, it is important to research the state-of-the-art technologies behind them. This section aims to provide an overview of the various technical elements of ASR, as well as how to evaluate them Appendix E.

Automatic Speech Recognition Models

ASR models are technologies that converts spoken language into text without the need of a human in the loop. They are widely used in applications such as voice assistants and transcription services. This section aims to give a background on the theory behind ASRs, as well as the most state-of-the-art ASR models.

Traditional ASR and Probability Theory

The fundamental goal of an ASR model can be written as follows [13]:

$$\hat{W} = argmax P(W|O) \quad for\, W \in L \tag{E.1}$$

Where W is a given word sequence and O is an accoustic input sequence and L is a set of possible word sequences. Using Bayes' theorem, this can be rewritten as:

$$\hat{W} = argmax \frac{P(O|W)P(W)}{P(O)} = argmax P(O|W)P(W) \quad for\, W \in L \tag{E.2}$$

Since P(O) remains constant across all candidate sequences, it does not affect the optimization process and can be ignored [13].

Traditional ASR Architecture

A typical speech recognition system consists of several key components, including the acoustic front-end, acoustic model, lexicon, language model, and decoder, as illustrated in Figure E.1. The acoustic front-end is responsible for processing the speech signal and extracting meaningful features that aid in recognition. During this feature extraction process, the raw audio waveform captured by a microphone is transformed into a sequence of fixed-size acoustic feature vectors. These vectors are then used to estimate the parameters of word or phoneme models based on the training data.

The decoder plays a crucial role in ASR by searching through all possible word sequences to determine the most probable sequence corresponding to the input speech. This probability is calculated using an acoustic model, which represents the likelihood of observing a given sequence of sounds for a particular word (P(O|W)), and a language model, which defines the probability of different word sequences (P(W)) [13].

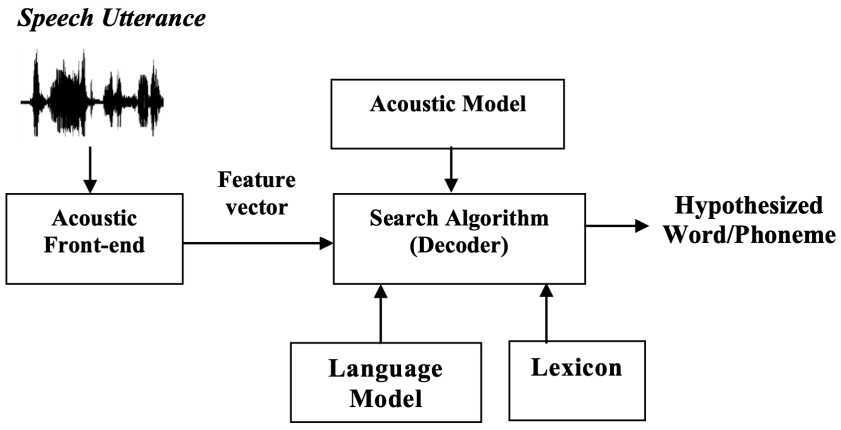

Figure E.1: Main components of a typical speech recognition architecture [13]

Modern Approaches in ASR
Recent advances in ASR have seen the emergence of end-to-end models like Whisper and Wav2Vec, which represent a shift from traditional ASR systems. These models streamline the speech recognition process by integrating the key components, such as the acoustic and language models, into a single neural network framework. Whisper, developed by OpenAI [14], processes raw audio input and generates text output without the need for separate feature extraction or explicit language modeling. By learning both acoustic and language representations jointly, Whisper eliminates the need for complex pipelines, significantly simplifying the ASR process while achieving impressive accuracy across various languages and acoustic conditions.

Similarly, Wav2Vec [15], particularly in its latest iteration, Wav2Vec 2.0, leverages a self-supervised learning approach to pretrain on raw audio data. Unlike traditional ASR systems, Wav2Vec also does not require handcrafted feature extraction and can directly learn powerful representations from the waveform itself. These modern approaches have revolutionized ASR by reducing reliance on domain-specific feature engineering and enabling more robust, scalable solutions for real-world applications.

Previous studies have looked into finetuning the Whisper and Wav2Vec models for the specific application of ASR in ATC, and demonstrated reaching WERs 19.8 % and 13.46% on the ATCO2 test set, respectively [9, 16]. With OpenAI's Whisper demonstrating one of the best performances with finetuning, it can be considered a state-of-the-art ASR method the ATC application. OpenAI has released a few versions of Whisper, including large-v2, on which the 13.46% WER was obtained [9], large-v3 and large-v3-turbo. Comparison between the models can be seen in Table E.1.

Feature	**Whisper Large-v2**	**Whisper Large-v3**	**Whisper Large-v3-Turbo**
Release Date	December 2022	November 2023	October 2024
Architecture	Transformer (Encoder-Decoder)	Transformer (Encoder-Decoder) with 128 Mel frequency bins	Optimized Transformer with reduced decoder layers (4 instead of 32)
Training Data	680,000 hours of multilingual audio	1 million hours of weakly labeled audio; 4 million hours of pseudo-labeled audio	Based on Whisper Large-v3, optimized for speed and efficiency
Performance	Baseline for comparison	10–20% error reduction over Large-v2	Preliminary reports suggest significantly improved speed over Large-v3, while maintaining comparable accuracy.
Language Support	Multilingual	Multilingual with improved language identification	Multilingual
Use Case Suitability	General-purpose ASR	Enhanced accuracy, especially in noisy environments	Applications requiring rapid transcription with minimal resource usage
Source	OpenAI Whisper Large-v2	Hugging Face: Whisper Large-v3	OpenAI Whisper Large-v3-Turbo

Table E.1: Comparison of Whisper Model Variants

Challenges in modern ASR in Air Traffic Communications
Previous research on ASR for analyzing air traffic communication has typically focused on specific contexts, such as a particular airport or en-route/approach scenarios. Fine-tuning machine learning models across different airports or control zones often requires new, domain-specific data, which can be difficult to collect and annotate. For example, ATC audio data from one airport (e.g., airport X) may not generalize well to another airport (e.g., airport Y) [17].

The collection and transcription of ATC data are both costly and time-consuming. The collection phase

involves capturing and preprocessing data, a task that can be automated. However, the transcription phase, which requires creating word-for-word transcripts of ATC speech, is typically done manually. This process can be expensive, as transcribing even one hour of ATC audio without silence requires significant human labor. For solutions targeting smaller airports, these costs can be prohibitive, raising the question of how to efficiently collect and process large amounts of ATC audio data [17].

Additionally, ATC audio data is often noisier compared to standard ASR corpora when captured through very-high-frequency (VHF) receivers. The signal-to-noise ratio (SNR) can range from 5 to 20 dB, which presents challenges in developing effective ASR systems and using their outputs for downstream tasks. While higher SNR data, sourced from operation rooms with close-mic recordings and reduced noise, are available from air navigation service providers (ANSPs), they are often limited to private use [17].

Measures of Performance

The performance of speech recognition systems is usually assessed in terms of accuracy and speed. Accuracy is often quantified using the Word Error Rate (WER), while speed can be measured by units such as the real-time factor (RTF), which compares the audio processing time to the length of the audio itself. A lower RTF (closer to 1.0 or below) indicates that the ASR system can transcribe speech at or faster than real-time, making it suitable for live ATC applications, whereas a higher RTF (>1.0) suggests that processing takes longer than the audio length.

Word errors are classified into three categories: insertions, substitutions, and deletions. The computation for WER is given by (E.3) [13]:

$$\text{Word Error Rate (\%)} = \frac{\text{Insertions (I)} + \text{Substitutions (S)} + \text{Deletions (D)}}{\text{Number of Reference Words (N)}} \times 100 \tag{E.3}$$

Where N is the total number of words, C is the number of correctly transcribed words, S denotes the number of substitutions, I number of insertions and D the number of deletions.

While WER is a generally useful tool for evaluating ASR performance, specifically in ATC, other metrics can convey important information as well. Such a metric is Callsign Recognition Rate (CRR), which measures the amount of correctly transcribed callsigns as a percentage of the total (ground truth) callsigns [18], or Callsign Error Rate (CSER) which is simply $1 - CRR$. Another relevant metric can be the Keyword Error Rate (KER), which is calculated similarly to WER and CSER but focuses exclusively on critical ATC-specific keywords. These include terms essential for command comprehension, such as “right”, “left”, “heading”, “descend”, and “climb”. Since misrecognition of these words can lead to safety risks, KER provides a more targeted evaluation of ASR reliability in command recognition.

Finally, the Command Error Rate (CER) can also be used in ATC, assessing the accuracy of full command recognition rather than individual words. Unlike KER, which evaluates isolated keywords, CER evaluates whether an entire ATC instruction has been transcribed correctly. This metric is particularly important for determining the operational reliability of ASR systems in real-world ATC environments, where even a small misinterpretation can lead to critical misunderstandings. By incorporating units such as CSER, KER, and CER, ASR performance can be more effectively assessed in the context of ATC communications, ensuring both safety and operational efficiency.

E.3. Large Language Models

LLMs can have multiple roles in the development of a post-processing pipeline in ATC transcripts. Due to their versatility, they can be applied for things such as synthetic data creation, NER, and direct correction of transcripts themselves. The field of LLMs has seen massive growth in recent years, making it even more important to have an overview of what different types of model there are and what they can be used for.

General Background on LLMs

Large language models are neural networks trained on massive text corpora to understand and generate human-like language. In recent years, transformers, in combination with the increase in computational resources and datasets, have revolutionized the realm of natural language processing [10], [11]. Transformers utilize attention mechanisms, which allow the training of transformers to be parallelized. Using the transformer architecture, researchers and engineers have been able to train large language models with up to hundreds of billions of parameters. These models process text by breaking it into smaller units (tokens), using attention mechanisms to understand relationships between words, and (in case of encoder-decoder or decoder LLMs) generating contextually relevant responses.

Training of large language models typically consists of two parts: pre-training and fine-tuning. Pre-training is the foundational phase in developing LLMs, where models learn from vast amounts of text data to acquire linguistic patterns, factual knowledge, and contextual understanding. This phase involves training on diverse corpora using self-supervised learning, allowing the model to predict missing words or generate coherent text. LLMs are designed using different architectures, ranging from encoder-decoder to decoder-only models, each incorporating distinct building blocks and loss functions [19]. Pretraining enables LLMs to develop a strong general-purpose foundation, which can then be adapted for specific tasks through fine-tuning methods.

Fine-tuning involves adapting pretrained models to specific tasks through various approaches: *Transfer learning* fine-tunes a general-purpose LLM using task-specific data, enhancing its performance for a particular application [20], [21]. *Instruction-tuning* improves the model's ability to follow user prompts by training it on structured instruction-response data, enabling better generalization across multiple tasks [22]. Often, during fine-tuning developers also makes use of reinforcement learning or reinforcement learning with human feedback (RLHF) to iteratively improve model behavior [23].

In recent years, many different large language models have emerged with their unique strengths and limitations. What sets them apart from each other are among others: architecture, the data they are trained on, the number of parameters, pre-training and finetuning methods. An overview of LLM agents deployed up until April fo 2024 can be seen on Figure E.2

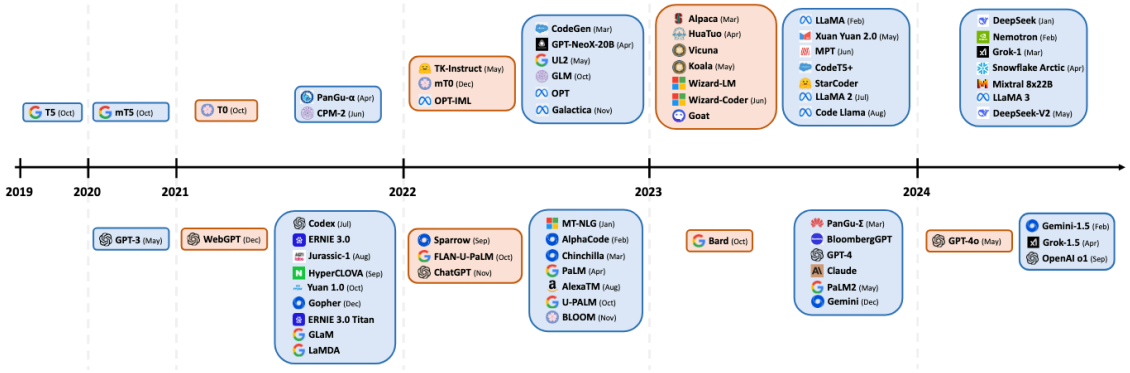

Figure E.2: Timeline of large language models. Blue represents pre-trained models and orange represents instruction-tuned models. Upper half corresponds to open-source models, whereas the bottom are closed-source [19].

Classification of LLMs

The selection of Large Language Models (LLMs) for post-processing ASR transcripts in ATC depends on their underlying architecture, openness, customization potential, computational efficiency, and multimodal capabilities.

Open-Source vs. Closed-Source

LLMs can be broadly classified based on their accessibility and customization potential:

- Open-Source Models (e.g., LLaMA, Mistral, Gemma): These models allow fine-tuning and local deployment, making them preferable for domain-specific applications where adaptation to ATC-specific terminology is required [24, 25, 26].
- Closed-Source Models (e.g., GPT-4, Claude, Gemini): These proprietary models offer high performance out-of-the-box with minimal setup. However, their restricted customization and reliance on API access can limit their applicability in specialized ATC workflows [27].

Model Size and Computational Cost

The size of an LLM directly impacts its performance, inference speed, and deployment feasibility:

- Small Models (e.g., Mistral 7B, Llama3.2 1B & 3B): These models are efficient in terms of speed and resource consumption, making them suitable for real-time applications. However, their reduced capacity may limit their ability to generate highly accurate or context-aware corrections [25].
- Large Models (e.g., GPT-4, LLaMA 65B): Larger models achieve higher accuracy and contextual understanding but come with higher computational and cost constraints [27, 24].

In theory some LLMs could also take in the raw audio next to the transcripts to improve transcription. Considering a separate ASR module is used to perform the speech-to-text transcription, only speech processing and generation is relevant for the thesis. Therefore, whether an LLM can process multimodal data or not is not considered.

Overview of Some State-of-the-Art LLMs

A high-level overview of some of the most state-of-the-art large language models are given below. While extensive, this is not an exhaustive list. Comparing LLMs is not straightforward as many of them have different strengths and use cases. Many have tried to summarize key differences between these models to generate better overview in amidst constant developments and growing number of models [28]. An example of such classification is displayed on Table E.2[29].

GPT (Generative Pre-trained Transformer) is a series of large language models (LLMs) developed by OpenAI, starting with GPT-1 in 2018. GPT-1 introduced the decoder-only transformer architecture, leveraging self-attention and pre-training on unlabeled text, followed by fine-tuning for downstream NLP tasks. GPT-2 (2019) significantly expanded the model size and training data, achieving state-of-the-art results in language modeling but requiring fine-tuning for specific tasks. GPT-3 (2020) further scaled up to 175 billion parameters, trained on a vast, high-quality dataset, and demonstrated impressive zero-shot and few-shot learning capabilities, making it widely adopted for applications like chatbots, content generation, and software development. This progression of GPT models paved the way for LLMs to become general-purpose AI tools, powering a range of real-world applications and influencing subsequent advancements in AI [30]. GPT-3.5, an intermediate model between GPT-3 and GPT-4, improved efficiency and response quality, addressing limitations like coherence and instruction-following. GPT-4 (2023) further enhanced reasoning, contextual understanding, and multimodal capabilities, while the latest models, including GPT-4-turbo, optimize efficiency, cost, and real-time interaction [31].

A downside to GPT models is that they can not be run locally and can only be accessed via cloud-based APIs or through the web interface [32]. This reliance on cloud-based access introduces potential drawbacks, such as privacy concerns, latency issues, and ongoing usage costs. Additionally, users have limited control over data processing and model customization, making local deployment alternatives, like open-source LLMs, an appealing option for those prioritizing data security and offline access [33]. In the context of the thesis, open models like GPT mcan only be used when working with open-sourced data.

The LLaMA series of models, published by Meta AI, has gained significant attention for its open-source nature and strong performance. With continuous updates from LLaMA [34] to LLaMA 3 [35], the models

have improved in scale and capability. It integrates advanced security and safety tools while achieving competitive performance against leading closed-source LLMs like GPT-4o and Claude 3.5 Sonnet on benchmarks such as MMLU [36], GSM8k [37], and HumanEval [38]. Llama is accessible and straightforward to run locally through Ollama [39] if local hardware allows it.

DeepSeek R1, released in January 2025 [40] is an advanced AI model designed for math and coding, demonstrating strong reasoning capabilities. With 671 billion total parameters, it achieves state-of-the-art performance on benchmarks like MATH-500 and AIME 2024, surpassing or matching OpenAI's o1 model. Notably, DeepSeek R1 was developed with an emphasis on cost-effective training, making it significantly more efficient compared to other leading LLMs. Upon its release, it quickly gained attention, driving a surge in website traffic [41]. As an open-source model, it is also freely accessible through an API, DeepSeek website, or to run locally through Ollama [39]. DeepSeek-R1 also integrates Chain-of-Thought (CoT) reasoning to enhance its problem-solving capabilities. By explicitly outlining its step-by-step thought process within <think> tags before delivering the final answer in <answer> tags, DeepSeek-R1 provides transparency in its reasoning. This structured approach allows users to follow the model's logical progression, which can sometimes lead to more accurate and interpretable responses.

Mistral 7B is a relatively compact large language model (LLM) developed by the Mistral AI team, featuring 7.3 billion parameters [42]. Despite its smaller size, it outperforms larger models such as LLaMA 2 (13B) and LLaMA 1 (34B) on multiple benchmarks, particularly in reasoning and coding tasks. Mistral 7B is fully open-source, allowing for unrestricted modification and commercialization.

However, its reduced size also presents limitations, particularly when compared to larger models like GPT-4. One major drawback is its smaller context window, which restricts the amount of text it can process at once. Additionally, it is generally less accurate and more prone to hallucinations than state-of-the-art, larger LLMs. As a result, Mistral 7B is best suited for applications where efficiency is prioritized over absolute accuracy. It is, however, an attractive choice if having low computational cost is of importance.

Model	Performance Metrics	Primary Applications	Parameter Size	Training Sources	Model Architecture	Distinct Features
OpenAI GPT-4	High accuracy, Low perplexity	Content creation, chatbots, code generation	Over 175 billion	Trained on a diverse range of internet sources	Transformer-based autoregressive model	Excels at generating fluent text, Adaptable for human-like interaction, Strong API support
Google Gemini	Strong logical reasoning	Tasks involving both text and images	Varies by version (up to 1 trillion)	Multimodal datasets including textual and visual information	Transformer-based with multimodal functionality	Merges textual and visual input, Designed for advanced problem-solving
Meta AI LLaMA	Competitive benchmark results	Research and academic studies	Up to 65 billion	Primarily trained on publicly accessible data	Transformer-based	Open-source for customization, Benefits from community contributions
Anthropic Claude	High responsiveness to user intent	Applications emphasizing safety and ethics	Varies by version	Specifically curated to minimize biases and harmful outputs	Transformer-based	Prioritizes ethical AI interactions, Implemented safeguards to limit harmful outputs
Mistral	Strong efficiency	Real-time use cases, software development	Up to 123 billion	Trained on multilingual datasets	Transformer-based with GQA and SWA	Advanced attention mechanisms (GQA), Improved function execution capabilities

Table E.2: Comparison of AI Language Models, retrieved from [29]

Google's *Gemini* models are developed by Google DeepMind [43], designed to enhance performance across a range of applications. When compared to OpenAI's GPT models, Gemini, with Google Search integration, excels in factual accuracy and source citation, making it more reliable for up-to-date and precise information. In contrast, ChatGPT prioritizes conversational fluency and creativity, generating more engaging, nuanced, and diverse responses, including storytelling, coding, and poetry. While Gemini favors conciseness and precision, ChatGPT enhances user interaction with personality and expressiveness. Some say that the choice betweem the two models depends on whether the user values accuracy and transparency (Gemini) or creativity and natural dialogue (ChatGPT) [44]. Like GPT, Gemini is closed-source. Google DeepMind has not released the model's weights or full architecture details to the public. While Gemini is accessible via Google's API and cloud-based services, it cannot be run locally or modified like open-source models such as LLaMA 3.3 or Mistral 7B.

On the other hand, Google has also released a series of open-source LLMs called *Gemma*. Gemma is developed to provide lightweight yet high-performing AI solutions. Initially released in February 2024, Gemma offers models with 2 billion and 7 billion parameters, optimized for efficient deployment on consumer-grade hardware [45]. Despite their compact size, Gemma models achieve notable benchmark results, often outperforming larger open models. However, their reduced size may limit performance

on tasks requiring deep contextual understanding or complex reasoning. In December of 2024, Google released Gemma 2, performing up to 10% better on certain benchmarks, compared to the earlier released Gemma models of comparable size [46]. Furthermore, when compared to Mistral 7B model and Llama 8B models, Gemma 2 2B and 9B models performed somewhat worse and somewhat better, respectively, on many benchmarks conventionally used for LLM evaluation. Gemma 27B performed better than both Mistral 7B and Llama 8B models, which is to be expected due to being a significantly large model [46].

Groq's AI hardware offers several advantages and trade-offs when considering it for real-world applications. A key strength is ultra-fast inference speed, making it highly effective for latency-sensitive tasks such as real-time AI interactions and high-frequency financial analysis [47]. However, a notable drawback is that Groq's hardware is optimized for inference rather than training, making it less suitable for applications where finetuning is needed [48]. These factors make Groq suitable for deployment in applications prioritizing speed and efficiency but less suited for flexible, research-intensive AI development.

Prompting

Prompt engineering is the process of crafting structured inputs to optimize the output of LLMs. In the context of ATC communications, this can include designing prompts that help AI models refine ASR transcripts as well as correcting errors while maintaining standard phraseology and accuracy. Since LLMs process text as tokens which represent words, subwords, or characters, the structure and clarity of prompts can sometimes directly influence the model's ability to generate accurate corrections. A well-designed prompt provides explicit instructions and, when necessary, contextual data to guide the model in reconstructing ATC communications with minimal distortion.

With the growth of LLMs, a lot of studies have been done on prompt engineering and classification of prompts [49], [50], [51], [52]. A prompt can be broken down to components, which are described in Table E.3. Not all components are always nescessary to include in the prompt (e.g. context), but they all have the potential to aid to guiding the LLM to desired output.

Component	Description
Instruction	Giving the LLM a *task* to guide its behavior, e.g.: `"Give me the corrected transcript"`
Context	External information to give the LLM context, e.g.: `"These transcripts are from an air traffic control centre in the Netherlands"`
Input data	The data that the LLM needs to process, e.g.: `"Ryan heir five five zero six"`
Output indicator	To indicate what format the output needs to be, e.g.: `"Return only the corrected transcription, with numbers written numerically instead of words"`

Table E.3: Components for LLM prompt

Fine-tuning LLMs

Fine-tuning a large language model (LLM) is highly effective for adapting general-purpose models to specialized tasks, improving both accuracy and efficiency. Pretrained LLMs possess extensive linguistic knowledge, but their performance in domain-specific applications, such as ATC transcription correction or other aviation-related tasks,is often limited due to a lack of exposure to specialized terminology and structured communication patterns [53]. Fine-tuning allows these models to learn domain-specific nuances, correct systematic errors, and enhance contextual understanding, leading to more reliable outputs in specialized fields [54].

Furthermore, fine-tuning enables LLMs to leverage task-specific objectives, such as error correction, summarization, or information extraction, refining their ability to process domain-specific data with greater precision [55]. This is particularly valuable in high-stakes environments like air traffic control (ATC), where ASR-generated transcriptions must be both accurate and compliant with standard phraseology.

Multiple approaches can be adopted when fine-tuning LLMs: self-supervised learning (where simply additional database, e.g. specifically from the ATC domain is fed to the LLM for training), supervised learning (where prompt-output pairs are fed for training) and reinforcement learning (using reward/penalty functions). Reinforcement learning can be done both by assigning rewards based on comparing to ground truth (when available), or by having a human who manually marks responses good or bad. This latter is also called Reinforcement Learning from Human Feedback (RLHF)[23]. Self-supervised, supervised and reinforcement learning methods are often used in combination when training LLMs. Further information on these three methods is given in Table E.4.

Table E.4: Fine-Tuning Approaches for LLMs Used in Correcting ATC Transcripts

Fine-Tuning Approach	**Description**	**Input Data and Examples**
Self-Supervised Learning (SSL)	The LLM is trained on large amounts of raw ATC transcripts without explicit corrections. It learns patterns, phraseology, and common structures in ATC communication.	Feeding raw ATC conversations (e.g., pilot-controller exchanges) into the LLM to help it recognize and reconstruct phraseology like “Cleared for takeoff” or “Turn left heading 270.”
Supervised Learning (SL)	The LLM is fine-tuned with labeled (input-output) data, learning to correct specific transcription errors by comparing mis-transcribed and corrected text.	Mis-Transcribed: ”Turn left at honey heading two seven zero.”; Corrected: ”Turn left at one-eighty, heading 270.”
Reinforcement Learning (RL)	The LLM is optimized using rewards for correct corrections and penalties for maintaining transcription errors, refining its ability to correct ATC transcripts over time.	If the LLM suggests ”Descend to flight level 200” instead of the incorrect ”Decent to fly level 200,” it receives a reward. If it fails to correct errors, it is penalized.

When finetuning LLMs, it is not just the input data that must be considereed, but also the parameters that are being adjusted. Retraining an entire LLM (all its layers) requires a huge amount of computational power. Another, more computationally light finetuning strategy is feature extraction, which freezes transformer Layers, and only affects output layers. The idea is that this strategy keps the pretrained model’s general language understanding while fine-tuning only specific layers.

Another strategy is Low-Rank Adaptation (LoRA). LoRA is a parameter-efficient fine-tuning (PEFT) method that inserts small trainable low-rank matrices into the transformer layers of an LLM instead of modifying all parameters. This approach significantly reduces computational costs and memory usage while allowing the model to adapt to new tasks effectively [56].

LoRA modifies the weight updates in a transformer’s attention layers by introducing low-rank matrices. Given an original weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA approximates the update ΔW using two smaller matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$, where $r \ll \min(d, k)$. Instead of directly training ΔW, the update is expressed as:

$$\Delta W = BA \tag{E.4}$$

The adapted weight matrix during fine-tuning is then:

$$W = W_0 + \Delta W = W_0 + BA \tag{E.5}$$

where W_0 remains frozen, and only A and B are trainable, significantly reducing the number of parameters. During inference or training, for an input feature $X \in \mathbb{R}^{k\times n}$, the transformation is computed as:

$$Y = WX = (W_0 + BA)X = W_0X + BAX \tag{E.6}$$

Since AX first projects the input into a lower-dimensional space r, and B maps it back to the original space, the number of trainable parameters is reduced from $d \times k$ to $r(d + k)$. This makes LoRA an efficient alternative to full fine-tuning, minimizing memory and computational overhead while preserving model performance [56].

E.4. Air Traffic Control

This section presents an overview of air traffic management (ATM) and air traffic control (ATC) to give general background and help place context for communications between air traffic controllers (ATCos) and pilots. It begins with a description of ATM, followed by a discussion on airspace classification, and concludes with an examination of ATC phraseology.

Air Traffic Management

ATM is responsible for the overall coordination of air traffic, and is further divided into Air Traffic Services (ATS) for safe and orderly movement of aircraft, Airspace Management (ASM) for efficient airspace allocation, and Air Traffic Flow & Capacity Management (ATFCM) for balancing traffic demand with capacity constraints. An overview of the structure of ATM is given in Figure E.3.

Air Traffic Services (ATS) ensure the safe and efficient movement of aircraft within controlled airspace by managing incoming, outgoing, and en-route traffic. ATS is provided within designated Flight Information Regions (FIRs), with control handed over at FIR boundaries. Upper airspace is typically divided into sectors, facilitating organized traffic flow between major international hubs. Surveillance is maintained through radar networks, allowing for radar-based separation of aircraft. ATS routes, often described as ”highways in the sky,” connect navigational beacons (e.g., VOR/DME), with traffic separated into designated lanes or altitudes. Altitude-based separation follows the semi-circular rule, assigning odd flight levels to eastbound flights and even levels to westbound flights to maintain safe vertical separation. ATS encompasses Air Traffic Control (ATC), Advisory Service, Flight Information Service (FIS), and Alerting Service.

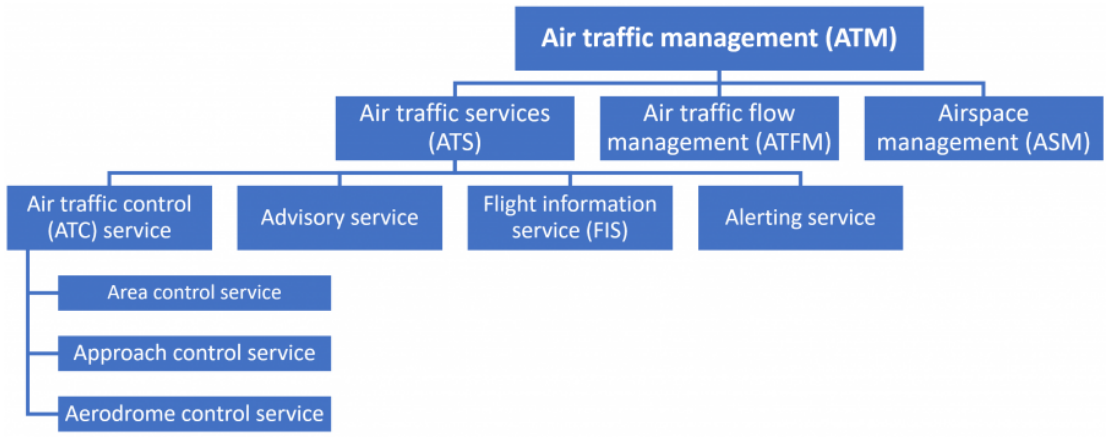

Figure E.3: Air Traffic Management Breakdown [57]

Flight Information Service (FIS) is responsible for collecting, managing, and distributing flight-related information to assist pilots in conducting their flights safely and efficiently. An example of FIS is ATIS (Automatic Terminal Information Service), which provides continuous VHF radio broadcasts to assist pilots operating within the Terminal Control Area (TMA) or Control Zone (CTR). It includes details on the runway in use, transition level (QNE to QNH), weather conditions (wind, visibility, precipitation, clouds, temperature), QNH, and operational updates. At Amsterdam Schiphol Airport (EHAM), ATIS is divided into Arrival Information (AI) and Departure Information (DI) to ensure pilots receive phase-specific updates for safe and efficient flight operations.

As part of Air Traffic Services (ATS), the Alerting Service (AL) works alongside the Flight Information Service (FIS) to ensure flight safety by notifying and coordinating with search and rescue (SAR) organizations in case of a potential emergency. While FIS provides pilots with essential information to enhance situational awareness, AL intervenes when communication is lost or distress signals arise.

Air Traffic Control (ATC) is a ground-based service in which air traffic controllers manage aircraft movements both on the ground and within designated controlled airspace, while also offering advisory services to aircraft operating in uncontrolled airspace. ATC is provided by air traffic controllers (ATCos) who issue clearances, instructions, and advisories to pilots to prevent collisions and manage air traffic flow.

Controlled Airspace Organization

Controlled airspace is geographically divided into different controlled zones, each managed by a specific ATC unit responsible for ensuring safe and efficient aircraft movement. In Table E.5 is an overview of UTA, CTA, TMA, and CTR, along with the ATC facilities that control them.

Airspace	Area	Managed By	Example (Netherlands)
UTA (Upper Control Area)	Upper airspace (approx. FL 195 and higher)	Area Control Center (ACC)	Maastricht Upper Area Control (MUAC)
CTA (Control Area)	Lower airspace (approx FL 195 or lower	ACC	Amsterdam CTA
TMA (Terminal Control Area)	Incoming/outgoing flights between CTR and CTA	Approach Control (APP)	Schiphol TMA (EHAM)
CTR (Control Zone)	Circular area around airport	Tower Control (TWR)	Amsterdam Schiphol CTR (EHAM)

Table E.5: Summary of Controlled Airspace Divisions and Their Management

Aircraft transitions between these airspace zones follow Standard Instrument Procedures, ensuring structured and efficient traffic flow.

Standard Terminal Arrival Routes (STARs) guide aircraft from en-route airspace into the Terminal Maneuvering Area (TMA) and finally to the Control Zone (CTR). Managed by Area Control Centers (ACC), STARs streamline arrivals by reducing pilot-controller communication, ensuring terrain clearance, and minimizing noise impact. Holding Areas (Stacks) are designated zones where aircraft may be instructed to hold in a circular or racetrack pattern before receiving clearance for approach. These are used near busy airports to sequence arrivals efficiently. Aircraft descend in a controlled manner, following altitude-based separation before being handed over to Approach Control.

Standard Instrument Departures (SIDs) transition aircraft from the Control Zone (CTR) through the Terminal Maneuvering Area (TMA) into en-route airspace. Managed by Approach Control (APP), SIDs organize departures to maintain separation from arriving traffic, reduce communication load, and ensure obstacle clearance. The Aerodrome Control Tower (TWR) oversees operations within the Control Zone (CTR), managing VFR traffic, taxiing, takeoffs, and final approach. In low visibility, Airport Surface Detection Equipment (ASDE) provides ground surveillance, supplementing visual observations.

ATC ensures safe separation using lateral (1–6 NM), longitudinal (2–12 NM), and vertical (1,000 feet) criteria. Above FL290, standard vertical separation increases to 2,000 feet, except in Reduced Vertical Separation Minimums (RVSM) airspace, where it remains 1,000 feet (FL290–FL410) in designated regions like Europe, the North Atlantic, and the Middle East.

As a last-resort safety system, the Airborne Collision Avoidance System (ACAS) autonomously detects midair conflict risks. Many aircraft are also equipped with the Traffic Alert and Collision Avoidance System (TCAS), which actively interrogates nearby transponders to issue collision advisories. Aircraft separation minimums also depend on wake turbulence categories, with larger separation required behind heavier aircraft.

E.4. Air Traffic Control Communications

ATC communication involves the exchange of critical information between air traffic controllers and pilots to ensure the safe and efficient movement of aircraft. The content of these communications typically includes instructions, clearances, weather reports, updates on flight paths, altitude assignments, and responses to queries. Controllers provide pilots with specific information on headings, altitude changes, and directions to avoid air traffic congestion or navigate airspace. Pilots, in turn, report their position, altitude, and intentions, as well as any anomalies, such as equipment malfunctions or unexpected weather conditions. These communications ensure that aircraft follow safe and precise routes, avoiding conflicts and enabling smooth coordination between various air traffic sectors. In Table E.6 some examples of ICAO standard phraseology can be found.

To minimize the risk of miscommunication and ensure clarity in a high-pressure environment, ATC uses a standardized set of phrases and terminology, known as ATC phraseology. This standardized language ensures that instructions are short, unambiguous, and universally understood by controllers and pilots worldwide, regardless of their native language. For example, the use of phonetic alphabets (e.g., “Alpha” for ”A” and “Bravo” for ”B”) eliminates confusion when communicating letters over radio. Numbers are read out in specific ways (e.g., ”one five” for 15 and ”two five zero” for 250) to prevent misinterpretation. Moreover, certain phrases like ”roger” (meaning ”received and understood”) and ”wilco” (short for ”will comply”) are used to streamline communication and avoid unnecessary repetition.

The structure and content of ATC phraseology are dictated by international standards set by the International Civil Aviation Organization (ICAO) [58], which ensures consistency in communication across different regions and languages. ICAO’s guidelines, as laid out in documents like Annex 10, define the rules for how messages should be conveyed, the sequence of exchanges, and the expected responses. These guidelines also ensure that communication is efficient, particularly in emergencies, where clear and concise instructions are crucial for safety. In addition to basic phraseology, ICAO has developed specialized terminology for specific situations, such as distress calls, weather updates, and equipment malfunctions, all designed to facilitate quick and accurate decision-making in dynamic, high-stress environments.

Table E.6: Standard ICAO ATC Phraseology Examples

Category	**Description and Example Phraseology**
Callsigns	Aircraft are identified by: - **Airline + flight number**: e.g., ”KLM123, climb FL350.” - **Aircraft registration**: e.g., ”November 123 Alpha Bravo, taxi to holding point Runway 18 via Alpha.”
Runway Designation	Named based on **magnetic heading** (e.g., Runway 09L = 090° Left). Example: ”Line up and wait Runway 09R.”
Taxi Instructions	ATC guides aircraft on the ground using taxiways and holding points. Example: ”Taxi to holding point Runway 27L via taxiways Alpha and Bravo.”
Takeoff Clearance	ATC provides takeoff authorization based on traffic and weather. Example: ”Lufthansa 789, wind 280 degrees, 5 knots, Runway 27R, cleared for takeoff.”
En-Route Navigation	ATC provides altitude, heading, and speed instructions. Example: ”KLM123, climb FL350.” ”Turn left heading 270, descend 5000 feet, QNH 1013.”
Approach Clearance	ATC directs aircraft toward the airport via STAR procedures. Example: ”Lufthansa 456, descend FL070, expect ILS approach Runway 25R.”
Landing Clearance	ATC issues final clearance for landing. Example: ”Lufthansa 456, wind 240 degrees, 8 knots, Runway 25R, cleared to land.”
Go-Around Instruction	If the runway is not clear, ATC instructs a missed approach. Example: ”Lufthansa 456, go around, climb 3000 feet, turn right heading 360.”
Emergency Phraseology	Standard ICAO emergency calls: - Mayday (Distress, e.g., engine failure, fire) - Pan-pan (Urgency, e.g., medical emergency) Example: ”Mayday, Mayday, Mayday, Lufthansa 789, engine failure, request immediate return to Frankfurt.”

Available Datasets

Finally, this section outlines freely available datasets for transcribed speech. These datasets are valuable as they provide ground truth transcriptions along with corresponding audio files, which can be processed using various ASR models to generate realistic erroneous transcriptions for analysis.

ATCOsim

The ATCOsim Air Traffic Control Simulation Speech corpus is a collection of air traffic control (ATC) operator speech, developed by Graz University of Technology (TUG) and the Eurocontrol Experimental Centre (EEC) [59]. It contains ten hours of recorded speech data captured during real-time ATC simulations, with the recordings made using a close-talk headset microphone. The speech is in English and is spoken by ten non-native speakers. The corpus includes orthographic transcriptions as well as additional metadata about the speakers and the recording sessions [59].

ATCO2

The ATCO2 test dataset, developed by the Idiap Research Institute [60], contains audio recordings along with transcriptions and various metadata. ATCO2-Test subset inclydes includes four hours of audio-transcription pairs, with 1.1 hours publicly available for free. These also come with metadata as well as lists closeby callsigns and waypoints. The dataset was collected from LKTB, LKPR, LZIB, LSGS, LSZH, LSZB, and YSSY airports, and the transcriptions were created by a combination of volunteers and paid annotators [1], [61].

LINDAT/CLARIAH-CZ

LINDAT/CLARIAH-CZ is a Czech center that provides certified data storage and natural language processing services. Their Air Traffic Control Communication corpus consists of recorded exchanges between air traffic controllers and pilots. The speech has been manually transcribed and annotated with speaker roles (pilot or controller, without revealing personal identities). With 20 hours of data, this corpus serves as a valuable resource for studying accented speech, as it originates from a non-English native country [62].

ANSP Dataset (LVNL)

Audio files have been provided to NLR from LVNL [63]. A subset of which was transcribed manually by a previous student at TU Delft, Jan van Doorn [9]. The data and transcriptions are not available publicly, but can be accessed by NLR employees. This dataset is quite useful for training models which are meant to be deployed by LVNL or, more generally, in the context of Dutch airspace. The length of the transcribed audio is 3 hours.

Research Definition

This section outlines the main research objective and corresponding research questions of the thesis.

Research Objective

The objective of this research is to develop and evaluate post-processing techniques using large language models (LLMs) to enhance the accuracy of automatic speech recognition (ASR) transcriptions in air traffic control (ATC) communications.

E.5. Primary Research Question

The main research question guiding this study, derived straight from the research objective, is:

Of which the primary research question can be formulated as follows:

Research Question

How can large language models improve the accuracy of ASR transcriptions in ATC communications?

The expectation is that LLMs will be able to enhance ASR transcriptions in ATC communications by leveraging contextual awareness, enforcing standardized phraseology, and correcting common ASR errors. By incorporating historical exchanges, radar data, and domain-specific patterns, LLMs can refine transcriptions, reducing ambiguities and improving accuracy in ATC transcriptions. They also have potential to infer missing words, and enforce ICAO-standard structures. However, integrating LLMs introduces challenges, such as the risk of hallucinations, where the model generates plausible but incorrect outputs, as well as potential overcorrection of ASR transcriptions, which may alter actual transmissions (although this should not be a big concern for most use cases, as long as the message is conveyed correctly). Additionally, latency concerns in real-time ATC operations and dependency on high-quality training data pose limitations. Finally, prompting and normalization on ASR models technically also target a lot of these strategies. Due to these factors, it is difficult to say at this point what the improvement rate will be of the post-processing step.

E.6. Secondary Research Questions

Furthermore, the thesis will aim to answer the following secondary research questions

- **SRQ1.1:** What types of common ASR errors can LLM-based post-processing techniques effectively correct and what types of errors are they less effective for?
- **SRQ1.2:** What is the impact of LLM size and complexity on ASR transcription accuracy in ATC environments?
- **SRQ1.3:** How does fine-tuning LLMs on ATC-specific data impact post-processing performance?
- **SRQ1.4:** Can LLM-generated synthetic data enhance the performance of fine-tuned LLMs on transcription correction tasks?
- **SRQ1.5:** (Optional) Can post-processing framework be adapted for real-time ASR transcription in ATC environments?

The expectation is that fine-tuning will enhance the performance of LLMs in post-processing and, at the very least, require less (elaborate) prompts to get desired outcomes. It is highly plausible that finetuning will decline the performance of the model on other, more general tasks. However, this is an acceptable trade-off, as the LLM is specialized for improving ATC transcriptions.

One of the main limitations of this research is hardware constraints. The available Tesla V100 GPU can only support training smaller LLMs (e.g., 3B or 7B parameters at most), restricting the exploration of larger-scale fine-tuning. Additionally, publicly available transcribed ATC data is scarce, necessitating the generation of synthetic training data. While artificial error injection in ATC transcripts seems straightforward, it may fail to accurately replicate real ASR errors. Using actual ASR transcriptions as a reference when prompting LLMs to synthesize training data could help mitigate this issue, but there is no guarantee that synthetic data will be fully representative of real-world ASR errors.

This research will explore these challenges and trade-offs to assess the viability of LLM-based post-processing for ASR in ATC communications.

References

[1] *Data Available from project*. URL: https://www.atco2.org/data.

[2] Hartmut Helmke et al. *Readback Error Detection by Automatic Speech Recognition and Understanding - Results of HAAWAII project for Isavia's Enroute Airspace*. Dec. 2022.

[3] Kevin L. Fox et al. "Leverage Large Language Models For Enhanced Aviation Safety". In: *2024 Integrated Communications, Navigation and Surveillance Conference (ICNS)*. ISSN: 2155-4951. Apr. 2024, pp. 1–11. DOI: 10.1109/ICNS60906.2024.10550651. URL: https://ieeexplore.ieee.org/abstract/document/10550651 (visited on 01/21/2025).

[4] Sandeep Badrinath et al. "Automatic Speech Recognition for Air Traffic Control Communications". In: *Transportation Research Record: Journal of the Transportation Research Board* 2676 (Aug. 2021), p. 036119812110363. DOI: 10.1177/03611981211036359.

[5] Juan Zuluaga-Gomez et al. *A Virtual Simulation-Pilot Agent for Training of Air Traffic Controllers*. arXiv:2304.07842 [eess]. Apr. 2023. DOI: 10.48550/arXiv.2304.07842. URL: http://arxiv.org/abs/2304.07842 (visited on 02/28/2025).

[6] Amrutha Prasad et al. *Speech and Natural Language Processing Technologies for Pseudo-Pilot Simulator*. Dec. 2022.

[7] Yi Ren et al. *FastSpeech 2: Fast and High-Quality End-to-End Text to Speech*. arXiv:2006.04558 [eess]. Aug. 2022. DOI: 10.48550/arXiv.2006.04558. URL: http://arxiv.org/abs/2006.04558 (visited on 01/30/2025).

[8] Hartmut Helmke et al. "Supporting Simulation Pilots by Automatic Speech Recognition and Understanding". en. In: *14th SESAR Innovation Days 2024, SIDS 2024*. Rom, Italien, Nov. 2024, pp. 1–9. DOI: 10.61009/SID.2024.1.03. URL: https://www.sesarju.eu/sites/default/files/documents/sid/2024/papers/SIDs_2024_paper_004%20final.pdf (visited on 01/20/2025).

[9] Jan-Willem van Doorn et al. "Whisper-ATC". en. In: (2024). URL: https://repository.tudelft.nl/record/uuid:8e02d222-5775-441d-94d2-96c26156cf43 (visited on 01/27/2025).

[10] Ashish Vaswani et al. *Attention Is All You Need*. arXiv:1706.03762 [cs]. Aug. 2023. DOI: 10.48550/arXiv.1706.03762. URL: http://arxiv.org/abs/1706.03762 (visited on 02/26/2025).

[11] Anton Chernyavskiy et al. *Transformers: "The End of History" for NLP?* arXiv:2105.00813 [cs]. Sept. 2021. DOI: 10.48550/arXiv.2105.00813. URL: http://arxiv.org/abs/2105.00813 (visited on 02/26/2025).

[12] Shuo Chen et al. "Read Back Error Detection using Automatic Speech Recognition". In: 2017. URL: https://www.semanticscholar.org/paper/Read-Back-Error-Detection-using-Automatic-Speech-Chen-Kopald/64018f82ee972144b46deb3073295f58010bfcff (visited on 01/31/2025).

[13] Karpagavalli S et al. "A Review on Automatic Speech Recognition Architecture and Approaches". In: *International Journal of Signal Processing, Image Processing and Pattern Recognition* 9 (Apr. 2016), pp. 393–404. DOI: 10.14257/ijsip.2016.9.4.34.

[14] *Introducing Whisper*. en-US. Apr. 2022. URL: https://openai.com/index/whisper/ (visited on 02/15/2025).

[15] Steffen Schneider et al. *wav2vec: Unsupervised Pre-training for Speech Recognition*. arXiv:1904.05862 [cs]. Sept. 2019. DOI: 10.48550/arXiv.1904.05862. URL: http://arxiv.org/abs/1904.05862 (visited on 02/15/2025).

[16] Juan Zuluaga-Gomez et al. *How Does Pre-trained Wav2Vec 2.0 Perform on Domain Shifted ASR? An Extensive Benchmark on Air Traffic Control Communications*. arXiv:2203.16822 [eess]. Oct.

2022. DOI: 10.48550/arXiv.2203.16822. URL: http://arxiv.org/abs/2203.16822 (visited on 02/11/2025).

[17] Juan Zuluaga-Gomez et al. “Lessons Learned in Transcribing 5000 h of Air Traffic Control Communications for Robust Automatic Speech Understanding”. en. In: *Aerospace* 10.10 (Oct. 2023). Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, p. 898. DOI: 10.3390/aerospace10100898. URL: https://www.mdpi.com/2226-4310/10/10/898 (visited on 02/15/2025).

[18] Shruthi Shetty et al. “Early Callsign Highlighting using Automatic Speech Recognition to Reduce Air Traffic Controller Workload”. en. In: 2022. DOI: 10.54941/ahfe1002493. URL: https://openaccess.cms-conferences.org/publications/book/978-1-958651-36-0/article/978-1-958651-36-0_66 (visited on 03/05/2025).

[19] Humza Naveed et al. *A Comprehensive Overview of Large Language Models*. arXiv:2307.06435 [cs]. Oct. 2024. DOI: 10.48550/arXiv.2307.06435. URL: http://arxiv.org/abs/2307.06435 (visited on 02/26/2025).

[20] Linting Xue et al. *mT5: A massively multilingual pre-trained text-to-text transformer*. arXiv:2010.11934 [cs]. Mar. 2021. DOI: 10.48550/arXiv.2010.11934. URL: http://arxiv.org/abs/2010.11934 (visited on 02/26/2025).

[21] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. arXiv:1910.10683 [cs]. Sept. 2023. DOI: 10.48550/arXiv.1910.10683. URL: http://arxiv.org/abs/1910.10683 (visited on 02/26/2025).

[22] Hyung Won Chung et al. *Scaling Instruction-Finetuned Language Models*. arXiv:2210.11416 [cs]. Dec. 2022. DOI: 10.48550/arXiv.2210.11416. URL: http://arxiv.org/abs/2210.11416 (visited on 02/26/2025).

[23] Daniel M. Ziegler et al. *Fine-Tuning Language Models from Human Preferences*. arXiv:1909.08593 [cs]. Jan. 2020. DOI: 10.48550/arXiv.1909.08593. URL: http://arxiv.org/abs/1909.08593 (visited on 02/26/2025).

[24] Hugo Touvron et al. “LLaMA: Open and Efficient Foundation Language Models”. In: *arXiv preprint arXiv:2302.13971* (2023).

[25] Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: https://jmlr.org/papers/v21/20-074.html.

[26] Mike Lewis et al. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: *arXiv preprint arXiv:1910.13461* (2019).

[27] OpenAI. “GPT-4 Technical Report”. In: *arXiv preprint arXiv:2303.08774* (2023).

[28] Yuliia Kniazieva. *Types of LLMs: Classification Guide*. 2024. URL: https://labelyourdata.com/articles/types-of-llms.

[29] AI-Pro. *A Comprehensive Comparison of All LLMs*. 2024. URL: https://ai-pro.org/learn-ai/articles/a-comprehensive-comparison-of-all-llms/.

[30] Mingyu Zong et al. *a survey on GPT-3*. arXiv:2212.00857 [cs]. Dec. 2022. DOI: 10.48550/arXiv.2212.00857. URL: http://arxiv.org/abs/2212.00857 (visited on 02/27/2025).

[31] OpenAI. *GPT-4 | OpenAI*. URL: https://openai.com/index/gpt-4-research/.

[32] *Chat-GPT*. URL: https://chatgpt.com/.

[33] *Guide to local LLMs*. URL: https://scrapfly.io/blog/guide-to-local-llm/?utm_source=chatgpt.com.

[34] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. arXiv:2302.13971 [cs]. Feb. 2023. DOI: 10.48550/arXiv.2302.13971. URL: http://arxiv.org/abs/2302.13971 (visited on 02/26/2025).

[35] *Introducing Meta Llama 3: The most capable openly available LLM to date*. URL: https://ai.meta.com/blog/meta-llama-3/.

[36] Dan Hendrycks et al. *Measuring Massive Multitask Language Understanding*. arXiv:2009.03300 [cs] version: 3. Jan. 2021. DOI: 10.48550/arXiv.2009.03300. URL: http://arxiv.org/abs/2009.03300 (visited on 02/26/2025).

[37] Mark Chen et al. *Evaluating Large Language Models Trained on Code*. arXiv:2107.03374 [cs] version: 2. July 2021. DOI: 10.48550/arXiv.2107.03374. URL: http://arxiv.org/abs/2107.03374 (visited on 02/26/2025).

[38] Karl Cobbe et al. *Training Verifiers to Solve Math Word Problems*. arXiv:2110.14168 [cs] version: 1. Oct. 2021. DOI: 10.48550/arXiv.2110.14168. URL: http://arxiv.org/abs/2110.14168 (visited on 02/26/2025).

[39] *Get up and running with large language models.* URL: https://ollama.com.

[40] *Deepseek R1 Release*. URL: https://api-docs.deepseek.com/news/news250120.

[41] DeepSeek-AI et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. arXiv:2501.12948 [cs]. Jan. 2025. DOI: 10.48550/arXiv.2501.12948. URL: http://arxiv.org/abs/2501.12948 (visited on 02/26/2025).

[42] *mistral 7b*. URL: https://mistral.ai/news/announcing-mistral-7b.

[43] Google DeepMind. *Gemini 2.0*. URL: https://deepmind.google/technologies/gemini/.

[44] Nitin Rane et al. "Gemini versus ChatGPT: applications, performance, architecture, capabilities, and implementation". en. In: *Journal of Applied Artificial Intelligence* 5.1 (Mar. 2024). Number: 1, pp. 69–93. DOI: 10.48185/jaai.v5i1.1052. URL: https://sabapub.com/index.php/jaai/article/view/1052 (visited on 02/27/2025).

[45] Google. *Gemma: Introducing new state-of-the-art open models*. 2024. URL: https://blog.google/technology/developers/gemma-open-models/.

[46] Gemma Team et al. *Gemma 2: Improving Open Language Models at a Practical Size*. arXiv:2408.00118 [cs]. Oct. 2024. DOI: 10.48550/arXiv.2408.00118. URL: http://arxiv.org/abs/2408.00118 (visited on 02/27/2025).

[47] Argonne National Laboratory. *Argonne Deploys New Groq System at ALCF AI Testbed*. 2023. URL: https://www.alcf.anl.gov/news/argonne-deploys-new-groq-system-alcf-ai-testbed-providing-ai-accelerator-access-researchers.

[48] Groq Inc. *Groq AI Technology White Paper*. 2022. URL: https://groq.com/technology.

[49] Pengfei Liu et al. *Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing*. arXiv:2107.13586 [cs]. July 2021. DOI: 10.48550/arXiv.2107.13586. URL: http://arxiv.org/abs/2107.13586 (visited on 03/03/2025).

[50] Banghao Chen et al. *Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review*. arXiv:2310.14735 [cs]. Sept. 2024. DOI: 10.48550/arXiv.2310.14735. URL: http://arxiv.org/abs/2310.14735 (visited on 03/03/2025).

[51] Louie Giray. "Prompt Engineering with ChatGPT: A Guide for Academic Writers". en. In: *Annals of Biomedical Engineering* 51.12 (Dec. 2023), pp. 2629–2633. DOI: 10.1007/s10439-023-03272-4. URL: https://doi.org/10.1007/s10439-023-03272-4 (visited on 03/03/2025).

[52] Jules White et al. *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. en. arXiv:2302.11382 [cs]. Feb. 2023. DOI: 10.48550/arXiv.2302.11382. URL: http://arxiv.org/abs/2302.11382 (visited on 03/03/2025).

[53] Jeremy Howard et al. *Universal Language Model Fine-tuning for Text Classification*. 2018.

[54] Suchin Gururangan et al. *Don't Stop Pretraining: Adapt Language Models to Domains and Tasks*. 2020.

[55] Long Ouyang et al. *Training language models to follow instructions with human feedback*. arXiv:2203.02155 [cs]. Mar. 2022. DOI: 10.48550/arXiv.2203.02155. URL: http://arxiv.org/abs/2203.02155 (visited on 03/03/2025).

[56] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. arXiv:2106.09685 [cs]. Oct. 2021. DOI: 10.48550/arXiv.2106.09685. URL: http://arxiv.org/abs/2106.09685 (visited on 03/03/2025).

[57] Skybrary. *Classification of Airspace*. URL: https://skybrary.aero/articles/classification-airspace.

[58] *ICAO Standard Phraseology. A Quick Reference Guide for Commercial Air Transport Pilots*. URL: https://skybrary.aero/sites/default/files/bookshelf/115.pdf.

[59] *ATCOSIM: Air Traffic Control Simulation Speech Corpus*. URL: https://www.spsc.tugraz.at/databases-and-tools/atcosim-air-traffic-control-simulation-speech-corpus.html.

[60] Juan Zuluaga-Gomez et al. *ATCO2 corpus: A Large-Scale Dataset for Research on Automatic Speech Recognition and Natural Language Understanding of Air Traffic Control Communications*. arXiv:2211.04054 [cs]. June 2023. DOI: 10.48550/arXiv.2211.04054. URL: http://arxiv.org/abs/2211.04054 (visited on 02/15/2025).

[61] Stephen S B Clarke et al. “Towards Understanding Data Requirements for Developing Automatic Speech Recognition Systems for Air Traffic Control”. en. In: ().

[62] *Air Traffic Control Communication Corpus*. URL: https://lindat.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0001-CCA1-0.

[63] *Luchtverkeersleiding Nederland*. URL: https://www.lvnl.nl.