FACULDADE DE CIÊNCIAS DA UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

LLM-Based Automated Taxi Request Handling in Multichannel Conversational Settings

Miguel Quatorze Matias Lopes





Masters in Artificial Intelligence

Supervisor: Prof. Michel Celestino Paiva Ferreira

July 31, 2026

© Miguel Quatorze Matias Lopes, 2026

LLM-Based Automated Taxi Request Handling in Multichannel Conversational Settings

Miguel Quatorze Matias Lopes

Masters in Artificial Intelligence

Approved in oral examination by the committee:

President: Prof. Carlos Soares
Referee: Prof. Ana Alves
Referee: Prof. Michel Ferreira

July 31, 2026

Abstract

This thesis investigates the operational viability of LLM-based agents for automated taxi request handling across voice and text channels in a real production environment. Four agents were implemented and deployed in live dispatch traffic at Geolink lda.: the WhatsApp Helper Agent (WAH), a structured multi-model pipeline using locally deployed Qwen3:30B and Gemma-4-31B-Heretic; the WhatsApp Conversational Agent (WAC), a fully conversational agent based on GPT 4.1-mini; the Napoleão voice expansion, a keyword-based voice detection layer built on a local Whisper model; and the Realtime Napoleão Agent (RTN), a low-latency voice agent built on OpenAI's Realtime GPT.

The results show that architectural scope, more than raw model capability, determined production viability. The WAH agent reached a service completion rate of 65% and an average user satisfaction score of 4.79 out of 5, with consistent growth over a three-month deployment. The Napoleão voice expansion accounted for 14% of voice prompt interactions and demonstrably served an older demographic at low operational cost. The RTN agent achieved a 53% service rate in live testing. The WAC agent, equipped with thirteen tools across five functional groups, could not sustain reliable operation due to hallucinated tool calls, incorrect tool parameterisation and message loops, and its test was ended early.

The central finding is that, with the models available at viable cost and latency, narrowing agent scope combined with tool-grounded commitments and explicit human-in-the-loop escalation outperforms full conversational autonomy in a production dispatch setting. The WAC and RTN architectures remain valid scaffolding for future models with stronger reasoning capability and lower cost.

Keywords: Large Language Models, Task-Oriented Dialogue, Conversational Agents, Taxi Dispatch Automation, Spoken Dialogue Systems, Automatic Speech Recognition, Human-in-the-Loop, Multi-channel Deployment

Acknowledgements

I'd like to thank every teacher, colleague, friend and family that helped me throughout my academic journey. I'd like to personality thank: My parents and sister.
The rest of my family.
My best friend João.
Diana and Hugo.
Geolink, especially Prof. Michel, Mr. Rui, Joaquim, André, Gonçalo, Gabrieli e Marcus.
My colleagues from M.IA, particularly Dani e Lucas.
My colleagues from B.IACD.
And everyone who shaped my journey.

Contents

Acknowledgements ii

List of Acronyms vii

1 Introduction 1
- 1.1 Context and motivation . . . 1
- 1.2 Problem definition . . . 1
- 1.3 Objectives . . . 1
- 1.4 Research questions . . . 2
- 1.5 Innovative aspects . . . 2

2 Background 4
- 2.1 Operational context . . . 4
- 2.2 Industry deployments of LLM-based voice agents . . . 4
  - 2.2.1 Drive-thru ordering as a high-noise, low-latency benchmark . . . 5
  - 2.2.2 Documented case studies and LLM-based conversational voice ordering . . . 5
  - 2.2.3 Implications for taxi request automation . . . 5

3 Literature Review 7
- 3.1 Task-oriented dialogue systems and LLMs . . . 7
  - 3.1.1 Classical modular architectures: NLU–DST–Policy–NLG . . . 7
  - 3.1.2 LLM-centric TOD: end-to-end prompting and tool-augmented agents . . . 8
  - 3.1.3 Controllability and safety: hallucinations, inconsistency, and prompt sensitivity . . . 8
  - 3.1.4 Evaluation: operational metrics for production deployment . . . 9
- 3.2 Spoken dialogue systems in telephony . . . 9
  - 3.2.1 Telephony as a constrained acoustic and interaction channel . . . 9
  - 3.2.2 Turn-taking, endpointing, and barge-in . . . 10
  - 3.2.3 Disfluencies, repairs, and grounding under uncertainty . . . 10
  - 3.2.4 Non-understanding detection and recovery policies . . . 11
  - 3.2.5 Evaluation for telephone-based spoken dialogue services . . . 11
- 3.3 Speech recognition for noisy environments . . . 11
  - 3.3.1 ASR in taxi call automation: challenge profile and model selection . . . 12
  - 3.3.2 Confidence estimation and risk-aware downstream decisions . . . 12
- 3.4 Information extraction and geographic grounding . . . 13
  - 3.4.1 Slot extraction as spoken language understanding . . . 13
  - 3.4.2 Structured extraction and schema validation in LLM-centric stacks . . . 13
  - 3.4.3 Geographic grounding . . . 14

3.4.4 Taxi-Link as a data and deployment case study . . . 14
3.5 Contextualization using historical user data . . . 15
3.5.1 Retrieval and ranking of top-k candidate locations . . . 15
3.6 Human-in-the-loop fallback and operational reliability . . . 15
3.6.1 Escalation triggers and transfer policies . . . 16
3.6.2 Context transfer and operator-facing summarization . . . 17
3.6.3 Measuring operational impact under quality and safety constraints . . . 18

4 Models and Model Deployment **19**
4.1 The balance problem . . . 19
4.2 Local deployment . . . 19
4.2.1 Hardware Setup . . . 20
4.3 OpenAI as a service . . . 20
4.4 Models . . . 20
4.4.1 Text Models . . . 21
4.4.2 Audio Models . . . 25

5 WhatsApp Helper Agent **29**
5.1 Auxiliary tools and structures . . . 29
5.2 Multi-model agent design . . . 30
5.2.1 Internal structure . . . 31
5.3 Integration with existing systems . . . 33

6 WhatsApp Conversational Agent **34**
6.1 LLM-based agent design . . . 34
6.1.1 Tools and functionalities . . . 34
6.1.2 Prompting . . . 38
6.2 Integration with existing systems . . . 39

7 Napoleão Voice Systems: IVR and RTN Agent **40**
7.1 Integration of Voice Detection . . . 40
7.1.1 Asterisk . . . 40
7.1.2 Internal Structure . . . 41
7.2 Realtime Napoleão Agent . . . 41
7.2.1 Low latency voice agent design . . . 41
7.2.2 Agent Structure . . . 42
7.2.3 Integration with existing systems . . . 45

8 Results **46**
8.1 Text Agents . . . 46
8.1.1 WhatsApp Helper Agent . . . 46
8.1.2 WhatsApp Conversational Agent . . . 48
8.2 Voice Agents . . . 48
8.2.1 Napoleão system . . . 48
8.2.2 Realtime Napoleão Agent . . . 49
8.3 Future Improvements . . . 50

9 Conclusions **52**
9.1 Architectural viability and the reliability-flexibility trade-off 52
9.2 Failure modes and the role of human-in-the-loop 53
9.3 Operational impact and deployment constraints 53
9.4 Limitations and outlook 54

References **55**

List of Figures

4.1 Qwen3:30B model statistics 22
4.2 Gemma-4-31B-Heretic model statistics 23
4.3 GPT 4.1-mini model statistics 25
4.4 Whisper model statistics 27
4.5 Realtime-GPT v1 model statistics 28

5.1 Web chat interface used by operators to monitor and intervene in WhatsApp conversations 30
5.2 WAH Agent Flow 31

6.1 WAC Agent Flow 35

7.1 Three Layer Conversation Agent 41
7.2 RTN Agent Flow 42

8.1 Daily evolution of conversations, unique clients, and finished services in the WhatsApp service 46
8.2 Napoleão system interaction outcomes during the voice prompt rollout 48

List of Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
ASR	Automatic Speech Recognition
BERT	Bidirectional Encoder Representations from Transformers
CPU	Central Processing Unit
DIET	Dual Intent and Entity Transformer
DST	Dialogue State Tracking
DTMF	Dual-Tone Multi-Frequency Signalling
ECML	European Conference on Machine Learning
ETA	Estimated Time of Arrival
FAQ	Frequently Asked Questions
FLAP	Faithful LLM Action Planning
GB	Gigabyte
GPS	Global Positioning System
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
HITL	Human-In-The-Loop
IVR	Interactive Voice Response
JGA	Joint Goal Accuracy
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LLM	Large Language Model
NLG	Natural Language Generation
NLU	Natural Language Understanding

PARADISE Paradigm for Dialogue System Evaluation

PBX Private Branch Exchange

PKDD Principles of Knowledge Discovery in Databases

POMDP Partially Observable Markov Decision Process

POS Point of Service

RAM Random Access Memory

RTN Realtime Napoleão Agent

SDS Spoken Dialogue System

SIP Session Initiation Protocol

SLU Spoken Language Understanding

TOD Task-Oriented Dialogue

UCI University of California, Irvine

VRAM Video Random Access Memory

WAC WhatsApp Conversational Agent

WAH WhatsApp Helper Agent

WER Word Error Rate

Chapter 1

Introduction

1.1 Context and motivation

Taxi dispatch services are operationally demanding: they require low-latency interaction, high reliability, and resilience to heterogeneous user behavior. In real deployments, taxi requests arrive through multiple channels, notably voice calls and messaging platforms. Voice calls are particularly challenging due to poor acoustic conditions, interruptions, and overlapping speech, while text requests can still be informal, incomplete, and ambiguous. Recent advances in LLMs open the possibility of improving task-oriented conversational automation by combining natural language understanding, controlled generation, and tool-assisted grounding.

1.2 Problem definition

The central problem is to determine which agent architectures and deployment strategies are operationally viable for LLM-based taxi request automation across voice and text channels. Rather than pursue isolated accuracy benchmarks, the thesis prioritizes real-world deployment: identifying which design patterns (structured intent-routing vs. full conversational agents) scale in production, which failure modes emerge in live traffic, and how effectively escalation mechanisms mitigate errors without overwhelming human operators. The evaluation focuses on two channels (Asterisk voice calls and WhatsApp text messages) operating in a live dispatch environment.

1.3 Objectives

The main objective is to implement and evaluate multiple agent architectures for LLM-based taxi request automation, deployed in live production traffic. Specifically, the thesis will:

- Implement four distinct agents spanning architectural approaches (structured multi-model routing vs. conversational end-to-end) and channels (voice via Asterisk and real-time SIP, text via WhatsApp).

- Deploy each agent incrementally in live traffic, measuring automation completion rates, operator handoff rates, and failure mode distributions in real operational conditions.
- Evaluate latency, cost, and capability trade-offs across deployment options: local model inference, cloud API services, and real-time streaming services.
- Document failure modes specific to telephony and messaging (ASR errors, ambiguous locations, user frustration) and assess how escalation policies and fallback mechanisms mitigate them.
- Quantify operational impact: reduction in routine operator workload, appropriate escalation coverage, and user acceptance across agent types.

1.4 Research questions

The work is guided by the following research questions (RQs):

1. RQ1 - Architectural viability: Which agent designs (structured vs. conversational, local vs. cloud) are deployable in production, and what are the failure rate and hallucination characteristics of each?
2. RQ2 - Failure modes and mitigation: What categories of errors emerge in live traffic (ASR failures, location ambiguity, model hallucinations), and how effectively do escalation and fallback mechanisms prevent dispatch errors?
3. RQ3 - Operational trade-offs: What automation rates can be sustained while maintaining acceptable escalation rates and user satisfaction, and how does this differ between voice and text channels?
4. RQ4 - Deployment constraints: How do latency budgets, operational costs, and reasoning capability constraints shape the choice between local models, cloud APIs, and real-time services?

1.5 Innovative aspects

- Comparative multi-agent deployment: simultaneous implementation of structured (WAH, Napoleão) and fully conversational (WAC, RTN) agents in the same operational environment, enabling direct assessment of architectural trade-offs (reliability vs. capability) under identical production conditions.
- Live-traffic evaluation: rather than offline benchmarks, agents are tested on actual customer interactions in live dispatch, capturing real-world failure modes, user behavior, and operational constraints (noise, ambiguity, user impatience) that lab evaluations typically miss.

- Staged deployment with early-risk milestone: keyword-based voice escalation (Napoleão expansion) deployed first to demonstrate proof-of-concept operational value, informing design of more complex agents (RTN) while minimizing production risk.
- Unified operator experience: web-based chat interface and context-transfer protocols ensure that escalations from any agent channel provide operators with structured state (inferred slots, confidence scores, reason for transfer), reducing re-elicitation and improving handoff efficiency.

Chapter 2

Background

2.1 Operational context

A taxi request arrives over one of two channels: voice, via a telephony stack built on Asterisk where a speech recognition module converts audio to text, or text, via the WhatsApp API. In both cases the conversational system must extract a pickup location, a destination, and any additional constraints (time, passenger count, accessibility needs) before a validated request can be handed to the dispatch backend. Telephony introduces particular challenges — band-limited speech, packet loss, and background noise — that directly constrain the quality of the speech-to-text step and must be accounted for throughout the pipeline.

Once a message is transcribed or received as text, LLMs handle the core conversational tasks: intent inference, structured slot extraction, response generation, and clarification question selection. Given the safety-critical nature of dispatch decisions, the system combines LLM output with deterministic validation rules and explicit escalation paths to human operators whenever uncertainty is high. This thesis focuses on these conversational intake stages; the dispatch backend is treated as a pre-existing operational component. The scientific techniques underpinning each stage — spoken dialogue, information extraction, grounding, and human-in-the-loop design — are surveyed in Chapter 3.

2.2 Industry deployments of LLM-based voice agents

While the scientific literature provides the methodological foundations for robust spoken dialogue, recent years have also seen an acceleration of industry deployments of AI-driven voice agents in high-volume transactional settings. These applied systems are relevant to taxi request automation because they operate under similar constraints: noisy audio, tight latency budgets, rapidly changing business rules (e.g., menus or policies), and low tolerance for semantic errors. Importantly, industry reports also emphasize the need for safe fallback and measurable operational impact, which complements academic evaluation protocols.

2.2.1 Drive-thru ordering as a high-noise, low-latency benchmark

Drive-thru order-taking has become a prominent application area for conversational voice AI. It combines strong acoustic mismatch (traffic noise, vehicle cabins, outdoor microphones), short but high-stakes dialogues, and a requirement to map free-form speech into structured actions over a constrained ontology (menu items, modifiers, quantities). These characteristics are closely analogous to taxi calls, where location entities and time constraints must be extracted and grounded reliably.

A salient example of the maturity challenges in real deployments is the reported termination of an automated drive-thru ordering pilot by McDonald's with IBM in 2024, illustrating that even large-scale trials can be constrained by error modes that are unacceptable in production (e.g., misunderstandings and recovery breakdowns) [1, 2]. In contrast, other chains have reported continued investment and scaling efforts. Wendy's has publicly described its "FreshAI" initiative in partnership with Google Cloud, explicitly positioning it as a generative-AI-enabled system designed to improve order handling in drive-thru settings [3–5].

More recently, Yum! Brands announced an industry-first collaboration with NVIDIA that includes "voice automated order-taking AI agents" spanning drive-thru and call-center operations, highlighting the strategic importance of voice automation at scale [6–8]. These industry-reported announcements, while not academic evaluations, are useful to motivate the feasibility of large-scale deployment and to identify recurring engineering constraints: integration with existing POS and operational systems, robust handling of local language and menu-specific terminology, and mechanisms for safe transfer to humans when uncertainty persists.

2.2.2 Documented case studies and LLM-based conversational voice ordering

Beyond press releases, some vendors publish case studies that document deployment design choices and measurable outcomes. SoundHound describes a drive-thru deployment at White Castle where a branded voice agent ("Julia") takes orders with human fallback, and the associated public case-study materials emphasize the importance of rapid response, order completion rates, and seamless staff handoff [9, 10]. Such case studies are valuable as "applied exemplars" of human-in-the-loop reliability: they foreground operational metrics (containment vs. transfer rates), and they illustrate how constraint satisfaction (menu ontology) is enforced in a noisy, real-time environment.

A complementary perspective appears in hardware/solution-provider case studies that explicitly mention large language models in drive-thru voice ordering. For example, an Intel case study describes a multilingual conversational voice AI order taker using an LLM to support context-aware responses and substitutions, underscoring the practical role of LLMs as orchestration and dialogue components around constrained business logic [11].

2.2.3 Implications for taxi request automation

These industry deployments suggest three practical lessons that transfer directly to taxi request automation. First, voice interaction quality is dominated by robustness under noise and by the sys-

tem's ability to recover from partial or incorrect hypotheses, rather than by average-case transcript accuracy alone. Second, high-stakes slots (for taxis: pickup/destination and time) require conservative confirmation and tool-grounded commitments, akin to the item/modifier confirmations in drive-thru ordering. Third, operational success depends on explicit human-in-the-loop designs and on monitoring metrics that balance efficiency (automation rate, time-to-completion) against safety and error severity.

Accordingly, including these applied systems in the dissertation background helps frame the research problem as part of a broader class of real-world voice automation challenges. The subsequent Literature Review then focuses on the scientific techniques needed to meet the reliability requirements observed in practice.

Chapter 3

Literature Review

The literature review covers the main techniques and failure modes relevant to the design decisions made in this work: task-oriented dialogue architectures, spoken dialogue in telephony, automatic speech recognition, and human-in-the-loop reliability strategies. Geographic grounding and historical-data contextualization are covered more briefly, as the agents developed here use location restriction and history-based suggestion strategies rather than full grounding or personalization pipelines.

3.1 Task-oriented dialogue systems and LLMs

Task-oriented dialogue (TOD) systems aim to help users accomplish well-scoped goals (e.g., booking, routing, customer support) by mapping natural language to a structured representation of intent and slots, maintaining dialogue state over multiple turns, and executing back-end actions (APIs, database queries) to complete the task [12]. In taxi request automation, TOD requirements are especially strict: the system must reliably identify pickup and destination locations, temporal constraints, party size, and special instructions, while managing uncertainty, user corrections, and operational constraints (e.g., service area, vehicle availability).

3.1.1 Classical modular architectures: NLU–DST–Policy–NLG

The traditional TOD architecture decomposes the problem into (i) natural language understanding (NLU), (ii) dialogue state tracking (DST), (iii) dialogue management / policy learning, and (iv) natural language generation (NLG) [12]. This modularity enables targeted supervision and debugging, but also introduces error propagation: ASR/NLU errors can corrupt the belief state and lead to incorrect actions (e.g., dispatching to the wrong pickup). A principled way to model uncertainty is to frame dialogue management as a partially observable Markov decision process (POMDP), where the system maintains a belief distribution over latent user goals and optimizes a policy under uncertainty [13]. This perspective is particularly relevant for taxi calls where noisy inputs and ambiguous locations are common.

3.1.2 LLM-centric TOD: end-to-end prompting and tool-augmented agents

LLMs have shifted TOD design toward natural-language prompting and end-to-end reasoning, often reducing hand-crafted rules and explicit intermediate representations. In an LLM-centric approach, the model may directly produce (a) a structured action (e.g., JSON function call) and (b) a user-facing utterance, sometimes in a single generation step. The benefit is rapid adaptation and broad linguistic coverage; the risk is reduced controllability and higher variance across prompts, temperature settings, or out-of-distribution user behavior.

A common mitigation is to combine LLMs with explicit tool use, where the model decides when to call external functions (e.g., geocoding, pricing, dispatch creation) and how to incorporate their outputs. Toolformer demonstrates a self-supervised approach in which language models learn to insert API calls during generation to improve factual and functional performance [14]. ReAct similarly interleaves reasoning traces and actions, showing that explicit action steps can reduce error accumulation in multi-step tasks [15]. For operational TOD, these ideas support an "agentic" architecture: the LLM acts as an orchestration layer, while correctness-critical operations are delegated to deterministic tools.

However, production TOD typically requires adherence to predefined flows (e.g., confirm pickup $\rightarrow$ confirm destination $\rightarrow$ confirm time $\rightarrow$ dispatch), especially in safety-critical domains. FLAP proposes constrained decoding to enforce workflow and API dependency constraints during generation, improving faithfulness to allowed action sequences without requiring extensive fine-tuning [16]. For taxi request automation, such constraints can encode domain policies (e.g., do not dispatch without an unambiguous pickup; require confirmation when geocoding confidence is low), making LLM behavior more predictable and auditable.

3.1.3 Controllability and safety: hallucinations, inconsistency, and prompt sensitivity

A central concern for deploying LLMs in TOD is hallucination: producing plausible but incorrect statements or parameters (e.g., fabricating a street name, asserting a booking succeeded without an API confirmation). Hallucination has been extensively documented and categorized in recent surveys, including taxonomies, detection strategies, and mitigation approaches [17]. In taxi automation, hallucinations can translate into operational failures with real-world consequences (wrong pickup, incorrect ETA, or unintended cancellations). Therefore, LLM outputs should be treated as *proposals* that must be validated against tools and constraints rather than as authoritative truth.

Practical controllability strategies include:

- **Schema-constrained outputs:** require the model to emit typed slots (pickup_lat, pickup_lon, pickup_confidence, etc.) and reject or repair invalid structures before execution.
- **Tool-grounded commitments:** only confirm actions (e.g., "Your taxi is booked") after receiving a successful response from the dispatch API.

- **Flow constraints and guarded transitions:** restrict allowable next actions (e.g., via constrained decoding [16]) and force explicit confirmation when ambiguity remains.
- **Uncertainty-aware dialogue:** use confidence thresholds to trigger clarification questions or escalation, echoing classical belief-based strategies [13].

3.1.4 Evaluation: operational metrics for production deployment

Formal benchmark metrics such as joint goal accuracy (JGA) or slot F1 on annotated corpora are difficult to apply to a deployment-first study. In practice, the operationally relevant measures are: automation completion rate, operator handoff rate and reason distribution, user satisfaction, and failure mode frequency. These are the metrics used throughout this thesis, motivated by surveys that emphasize corpus-based metrics can fail to predict real interactive performance in live settings [18, 19].

3.2 Spoken dialogue systems in telephony

Telephone calls remain a high-volume channel for taxi requests and impose constraints that are materially different from app- or chat-based interactions. Compared to text, telephony introduces acoustic degradation (band-limited speech, environmental noise), interactional phenomena (barge-in, disfluencies, overlap), and stringent latency expectations, all of which amplify error propagation from ASR to downstream dialogue decisions. This section reviews the main telephony-specific challenges and the stabilization techniques typically used to maintain robustness and safe task completion, with particular emphasis on turn-taking, confirmation, and non-understanding recovery policies. These considerations motivate the ASR robustness and confidence-estimation methods discussed in the following section.

3.2.1 Telephony as a constrained acoustic and interaction channel

Despite the rise of mobile applications, telephone calls remain a high-volume entry point for on-demand mobility in many markets, and they impose distinctive constraints on spoken dialogue systems (SDS). First, the acoustic channel is often *band-limited* (classically narrowband telephony), with additional distortions introduced by handset variability, background noise, packet loss (VoIP), and conversational overlap. These factors interact with ASR errors and latency, amplifying the risk of downstream dialogue failures (e.g., incorrect pickup grounding). Telephony therefore favours designs that explicitly manage uncertainty and minimize the cost of misunderstandings through rapid repair and confirmation.

Second, the interaction is *audio-only* and typically time-pressured: callers may be distracted, multi-tasking, or speaking from noisy environments, and they often expect fast progress rather than exploratory dialogue. From a grounding perspective, this raises the cost of extended clarification sub-dialogues and makes efficient confirmation crucial [20, 21].

3.2.2 Turn-taking, endpointing, and barge-in

Turn-taking is a core determinant of perceived naturalness and efficiency in telephony SDS. Unlike text chat, voice interaction is sensitive to *timing*: premature system responses can interrupt the caller, while delayed responses create awkward gaps that callers may interpret as failure. Endpointing decisions (when the system decides the user has finished speaking) are therefore not merely an ASR detail, but an interaction policy choice that affects dialogue flow and task success.

A large line of work treats endpointing and turn transitions as decision problems that trade off latency against the risk of cutting the user off. Classic approaches optimize endpointing thresholds using dialogue features [22] and model turn-taking with explicit state machines [23]. More recent research emphasizes incremental and predictive turn-taking: the system continuously estimates whether the user will continue, enabling earlier planning of responses and more human-like timing [24–27]. For taxi request automation, these models matter operationally because callers frequently produce incremental information ("é na Rua X... não, afinal na Avenida Y") and may interrupt prompts once they realize what the system is asking.

An emerging direction is *full-duplex* spoken interaction, where the agent can listen and speak simultaneously, enabling backchannels and mid-utterance repairs analogous to human conversation. Duplex modelling work highlights both the promise (lower interactional latency) and the engineering challenges (e.g., coherence of chunked generation and synthesis) [28, 29]. While many production IVR systems remain effectively half-duplex (prompt-then-listen), duplex capabilities are particularly attractive for short, transactional calls such as taxi bookings, where time-to-dispatch is a primary KPI.

3.2.3 Disfluencies, repairs, and grounding under uncertainty

Telephone speech is rich in disfluencies (filled pauses, repetitions, self-repairs) and often includes partial or evolving commitments about locations and times. Robust SDS must therefore distinguish between *content-bearing* information and repair phenomena, and they must support conversational mechanisms that restore shared understanding efficiently. Grounding theory motivates explicit acknowledgement and confirmation acts as a way to establish mutual belief about critical slots [20]. In practice, this often takes the form of mixed-initiative strategies: the system confirms uncertain fields while allowing the user to supply missing information in any order.

Recent reviews of conversational breakdowns emphasize that repair is not a single mechanism but a family of strategies deployed by both the system and the user (repetition, rephrasing, elaboration, specificity escalation), and that poorly chosen clarification requests can increase frustration and prolong calls [30]. For taxi automation, repairs frequently arise from ambiguous toponyms, near-homophones, and landmark-based referring expressions; effective strategies include (i) proposing short candidate lists, (ii) confirming at the granularity that matches uncertainty (street vs. number vs. landmark), and (iii) switching from open questions to constrained prompts when repeated failure signals a need for tighter control.

3.2.4 Non-understanding detection and recovery policies

A distinguishing feature of telephony SDS is the prevalence of *non-understandings* (the system cannot form an interpretation) and *misunderstandings* (the system forms an incorrect interpretation). Production systems typically incorporate confidence-based rejection and explicit recovery policies. Bohus and Rudnicky show that rejection thresholds induce a non-trivial trade-off between rejecting too often (causing unnecessary clarifications) and accepting low-confidence hypotheses (causing costly misunderstandings), and propose principled optimization methods [31]. They further analyze non-understanding recovery strategies empirically and demonstrate that different recovery moves have different success profiles [32]. Beyond hand-tuned heuristics (e.g., "repeat → help → transfer"), online learning can be used to adapt recovery policies over a set of strategies while respecting operational constraints [33].

For taxi request automation, recovery policies should be aligned with domain risk: a misunderstanding of pickup location is typically more severe than a misunderstanding of passenger count. This motivates asymmetric policies where low confidence in geocoding triggers stronger confirmation, while low confidence in secondary slots may trigger softer, implicit confirmation. Importantly, recovery should remain *task-progressive*: each repair step should either reduce uncertainty (e.g., by narrowing candidates) or trigger escalation to a human operator when uncertainty remains high after a bounded number of attempts.

3.2.5 Evaluation for telephone-based spoken dialogue services

Evaluating telephony SDS requires connecting component-level metrics (ASR accuracy, slot error rate, latency) with user-perceived service quality and operational outcomes. PARADISE provides an influential framework that relates task success and dialogue costs to an overall performance function, enabling comparisons of dialogue strategies beyond raw recognition metrics [34]. For telephone-based services specifically, quality-of-service taxonomies emphasize that perceived quality depends not only on correctness but also on interactional factors such as responsiveness, error recovery, and conversational efficiency [21].

In taxi dispatch settings, evaluation should therefore combine: (i) *end-to-end task success* under realistic telephony noise; (ii) *time-to-dispatch* and turns-to-completion; (iii) *repair burden* (frequency and depth of clarification sub-dialogues); and (iv) *handoff rates* with reason codes (low confidence, repeated failures, user request). This mixed evaluation view is essential because improvements in one dimension (e.g., more confirmations) may increase call duration while reducing costly errors, and the optimal point depends on operational constraints and safety requirements.

3.3 Speech recognition for noisy environments

Automatic speech recognition (ASR) is a critical enabling component for fully automated taxi request handling in telephony and voice channels. In practice, taxi calls are affected by strong acoustic and channel variability (e.g., background traffic, in-car noise, speakerphone, mobile codecs),

which makes robustness and uncertainty-aware downstream handling as important as raw word error rate (WER).

3.3.1 ASR in taxi call automation: challenge profile and model selection

Voice channels introduce strong acoustic variability: background traffic, in-car noise, speakerphone, and telephony compression artifacts. Robust ASR requires balancing transcription accuracy, latency, and deployment overhead. For the voice agents in this work, OpenAI's Whisper was selected: a large-scale weakly supervised model with strong cross-domain robustness that avoids the need for domain-specific fine-tuning, augmentation pipelines, or custom bias lists [35]. The accepted trade-off is that Whisper operates non-streaming (recording and then transcribing a fixed-length utterance), which introduces predictable latency but simplifies pipeline integration. The key limitation observed in practice was transcription accuracy under heavy noise and strong accents, which motivated the use of confidence-threshold fallback rather than attempting to correct ASR output downstream.

3.3.2 Confidence estimation and risk-aware downstream decisions

In taxi request automation, an ASR transcript is not an end in itself; it is an input to downstream dialogue management and dispatch actions. Consequently, the ASR should output not only text but also *calibrated confidence* at appropriate granularities (token, word, and utterance), enabling risk-aware policies: ask clarification questions, confirm critical entities, or escalate to a human operator.

Confidence estimation in end-to-end ASR has been studied both from an evaluation perspective and through model-based predictors trained to detect errors. Oneata et al. provide a systematic evaluation of word-level confidence methods for end-to-end ASR, highlighting practical baselines and calibration-oriented approaches [36]. Other approaches train dedicated confidence modules, including multi-task formulations that jointly predict word confidence and deletion events, which is important because end-to-end outputs may omit critical words that never appear as explicit hypotheses [37]. Complementary work proposes residual energy-based models that can act as utterance-level confidence estimators while also improving recognition quality [38].

Operationally, confidence should be used to enforce *tool-grounded commitments*: for example, a system should not confirm a pickup location solely based on a low-confidence named-entity hypothesis. Instead, it should trigger targeted confirmation (e.g., repeating the inferred street and municipality), or perform constrained re-asking focused on the uncertain span. This design links ASR uncertainty to dialogue policy in a principled way: uncertainty should drive clarification, disambiguation, or escalation rather than being treated as an internal diagnostic.

3.4 Information extraction and geographic grounding

Automating taxi requests requires converting conversational utterances into (i) a structured request object (pickup, destination, time, passengers, constraints) and (ii) grounded geographic entities that can be mapped to coordinates and operational regions. In real deployments, the difficulty is not only extracting the correct slots, but ensuring that each extracted location is *actionable*: it must be resolvable to a precise point or polygon with adequate confidence, under noisy ASR transcripts, partial addresses, colloquial references, and local naming conventions.

3.4.1 Slot extraction as spoken language understanding

Slot extraction in task-oriented systems is commonly framed as sequence labeling (token-level tagging) paired with utterance-level intent detection, often trained jointly to exploit dependencies between global intent and local slot spans [39]. Transformer-based encoders have become standard for this setting: fine-tuning BERT-style representations for joint intent classification and slot filling yields strong results on canonical SLU benchmarks and improves robustness to rare or domain-specific lexical items [40]. In practical taxi automation, this matters because pickup/destination mentions are frequently sparse, abbreviated, or include uncommon named entities (e.g., minor landmarks).

In addition to fully neural encoders, production SLU stacks often benefit from hybridization with sparse features and domain dictionaries. DIET (Dual Intent and Entity Transformer) exemplifies this pragmatic design: it pairs transformer representations with additional sparse signals in a multi-task architecture for intent and entity prediction, a combination that can be advantageous when labeled in-domain data is limited [41]. For taxi requests, domain lexicons (street names, POIs, hospitals, transit hubs) can be injected at multiple points: as gazetteer features for candidate generation, as contextual bias in decoding, or as post-hoc validators of extracted spans.

A recurring challenge is that the "correct" slot value is not merely a span of text but an *operational variable*. For example, "Aeroporto" may be a valid destination mention but must be resolved to a specific airport terminal or entrance consistent with the service area and dispatch conventions. Consequently, the extraction module should output not only slot values but also uncertainty measures (e.g., confidence, alternative hypotheses), which can trigger clarification turns or tool-based verification.

3.4.2 Structured extraction and schema validation in LLM-centric stacks

Recent deployments increasingly use LLMs to produce structured outputs directly (e.g., typed JSON objects) and to orchestrate tool calls (geocoding, routing, pricing, dispatch APIs). While this can reduce feature engineering, it introduces new failure modes: schema violations (missing fields, wrong types) and confident but incorrect slot values. A practical response is to treat the model output as a *proposal* and enforce strict post-validation. For instance, function-calling fine-tuning and multi-task supervision have been studied explicitly to improve tool-oriented structured

generation [42]. Complementarily, post-processing layers that transform unconstrained model text into schema-compliant records (while preserving content) have been proposed to raise reliability in structured generation settings [43]. In taxi automation, this "extract–validate–ground" pattern supports auditable pipelines where downstream actions are executed only after schema checks and tool-grounded confirmation.

3.4.3 Geographic grounding

Full geocoding pipelines (toponym resolution, candidate disambiguation, fuzzy matching against gazetteers) are beyond the scope of the agents developed in this work. Location ambiguity is instead addressed at the agent level: the RTN agent restricts pickup locations to a pre-defined set of the caller's most frequent addresses, while the WAC agent uses service-area coverage checks, delegating unresolvable cases to human operators rather than attempting autonomous address disambiguation.

3.4.4 Taxi-Link as a data and deployment case study

In addition to general-purpose geoparsing and geocoding benchmarks, the taxi domain has benefited from real operational datasets and deployment-oriented studies that expose geographically grounded failure modes under realistic constraints. A notable example is the Porto taxi trajectory dataset released in the context of the ECML/PKDD 2015 Discovery Challenge and later mirrored by the UCI Machine Learning Repository [44]. The dataset comprises a full year of trajectories for hundreds of taxis in Porto and includes variables that reflect dispatch context (e.g., whether a trip was dispatched by a central, anonymized caller identifiers for centrally dispatched trips, taxi-stand identifiers, and a polyline with GPS samples), enabling research on origin/destination inference and operationally constrained prediction.

This dataset has been repeatedly used by the research community as a reference for mobility prediction tasks. For example, Lan et al. leverage it as one of their real-world evaluation datasets for travel time estimation, explicitly citing the Porto taxi trajectory dataset and its public availability [45]. While trajectory prediction is not itself a geocoding problem, it is tightly coupled to geographic grounding in taxi automation: origin and destination hypotheses must be mapped to actionable coordinates and validated against service constraints (coverage areas, stand conventions, plausible routes and travel times).

Complementarily, deployment-oriented work in the Taxi-Link ecosystem shows how geographic grounding interacts with conversational automation in telephony. Ribeiro's master's thesis describes a real call-center IVR setting where historical caller data is used to predict and propose pickup locations, which must then be confirmed and grounded to operational addresses before dispatch [46]. Together, these sources motivate grounding pipelines that (i) separate candidate generation from disambiguation, (ii) enforce explicit confirmation under uncertainty, and (iii) instrument error severity (e.g., wrong street vs. wrong municipality) to reflect dispatch risk.

3.5 Contextualization using historical user data

Contextualization leverages past interactions to reduce ambiguity and shorten the path to task completion. In taxi request automation, historical signals are particularly valuable because many callers exhibit strong regularities: recurring pickup points (e.g., home, workplace), habitual time windows, and stable service preferences. When the system can condition its hypotheses on caller identity and context, it can move from "extract everything from scratch" to a *predict–confirm* interaction style, where the main role of dialogue is to validate high-probability candidates rather than to elicit all slots from first principles.

3.5.1 Retrieval and ranking of top-k candidate locations

A practical framing of contextualization is to treat the next pickup (and, when applicable, destination) as a ranking problem: given a caller identifier and the current context, retrieve the most plausible candidates from that caller's historical set and/or from a population-level candidate pool. The output is a top-k list that can be presented for confirmation (e.g., "Is the pickup at *Rua X*? Press 1 for yes, 2 for no...") or used to bias downstream modules (ASR, NLU, and geocoding). This approach is attractive in telephony because it enables low-latency interactions with minimal cognitive load and can be aligned with deterministic validation logic.

In a real taxi call-center setting, Ribeiro's dissertation demonstrates this principle through pickup-address forecasting conditioned on each customer's call history [46]. The proposed pipeline combines (i) an unsupervised clustering stage that groups geographically proximate pickup points and (ii) a supervised classification stage that predicts the next pickup based on historical patterns and temporal features [46]. Importantly, the system operationalizes prediction as *top-k suggestion*: after predicting a cluster or class, a readable address suggestion is produced (e.g., by selecting the most frequent address within the predicted cluster), enabling an IVR to confirm the hypothesis with the caller [46]. The same work reports integration into an operational IVR ("Napoleão") where callers validate the suggested pickup using DTMF input, providing an end-to-end template for "history-conditioned suggestion + explicit confirmation" in telephony [46].

This approach directly informs the RTN agent design in this work, where each caller's five most common pickup locations are used as the candidate set, operationalising the predict-then-confirm pattern from [46] within a voice interaction.

3.6 Human-in-the-loop fallback and operational reliability

In real-world taxi request automation, *human-in-the-loop* (HITL) mechanisms are not merely a contingency plan but a core reliability layer: they provide a safety net against recognition errors, ambiguous grounding, and unanticipated conversational behavior, while preserving user trust and service continuity. In telephony settings, the cost of unresolved uncertainty can be operationally high (e.g., dispatching to an incorrect pickup) and socially salient (e.g., customer frustration after repeated clarifications). Accordingly, robust systems operationalize HITL as an explicit part of the

dialogue policy, defining when to escalate, what information to transfer, and how to quantify the trade-offs between automation rate and service quality.

3.6.1 Escalation triggers and transfer policies

A classical and still highly relevant perspective is to treat *non-understanding* and *misunderstanding* as distinct failure modes that demand different recovery strategies. Empirical analyses of spoken dialogue breakdowns show that non-understanding events (e.g., ASR failure, out-of-grammar input) often require targeted repair prompts or explicit confirmation, whereas misunderstandings can propagate silently and may only surface later as operational errors [47]. In telephony-based taxi ordering, the latter is especially problematic because the dialogue can remain locally fluent while gradually drifting toward an incorrect pickup, destination, or time constraint.

From a policy standpoint, escalation triggers can be grouped into three complementary families:

(i) Confidence- and uncertainty-driven triggers. Confidence signals can originate at multiple layers: ASR (word/utterance confidence), NLU (intent/slot confidence), and downstream grounding (e.g., multiple plausible pickups after geocoding). When the system cannot reduce uncertainty below a domain-specific threshold through clarification (e.g., "Which of these two streets do you mean?"), escalation is preferred over continued probing. A key insight from prior work is that recovery should be *adaptive*: instead of fixed scripts, systems can learn when to ask a clarification, when to confirm, and when to transfer [33]. This is particularly relevant for taxi calls, where time pressure and caller impatience constrain the number of repair turns that remain acceptable.

(ii) Interaction-pattern triggers. Repeated repair loops (e.g., multiple failed confirmations, repeated "I didn't catch that" events) are a strong operational signal that the dialogue is unlikely to converge within an acceptable number of turns. In practice, escalation policies often use bounded retry budgets (e.g., after k repair attempts) combined with severity-aware rules (e.g., a single unresolved ambiguity in the pickup may justify escalation earlier than an ambiguity in the number of passengers). This aligns with telephony realities: a short and successful human takeover is frequently preferable to a prolonged automated interaction with a high risk of error.

(iii) User-driven and affective triggers. Even when the system remains technically capable of continuing, explicit requests such as "I want to speak to a person" should be treated as authoritative escalation triggers. Additionally, pragmatic cues of frustration (barge-ins, repeated negations, sighs, prosodic changes) are well-known predictors of deteriorating user experience in spoken systems. While robust affect modelling is outside the scope of many deployments, *lightweight* proxies—such as rapid turn-taking, repeated corrections, or negative lexical markers—can be integrated as conservative escalation signals.

Taxi-call automation in operational settings provides a concrete illustration of these principles. In the Taxi-Link context, Ribeiro's thesis describes an IVR workflow in which an automatic agent

suggests a predicted pickup address and the caller can either accept via DTMF or transfer to a human operator, making escalation an explicit and user-controlled branch of the flow [46]. This design embodies a pragmatic stance: automation is pursued where confidence is high, but human support remains immediately available to preserve service quality.

3.6.2 Context transfer and operator-facing summarization

Escalation quality depends not only on *when* the system transfers but also on *what* it transfers. Without structured context transfer, handoff can paradoxically increase total workload (operator time plus caller repetition), undermining the purpose of HITL. Consequently, robust architectures treat the operator as a downstream "consumer" of the dialogue state, requiring a concise, faithful, and action-oriented summary of what the system inferred and why it escalated.

A practical operator handoff package typically includes:

- **Current hypothesis state:** normalized pickup/destination candidates (top-k), time constraints, passenger count, and special instructions.
- **Confidence and evidence:** confidence scores, grounding evidence (e.g., matched landmark), and ambiguity sources (e.g., two streets with similar names).
- **Dialogue trace (compressed):** the minimal subset of turns explaining the state and user corrections.
- **Reason for escalation:** which trigger fired (low confidence, repeated repairs, explicit request), enabling operators to calibrate trust in the inferred state.

Recent work in industry-track dialogue summarization supports the feasibility of *incremental* note-taking that reduces agent context-switching costs while retaining human oversight. Wu et al. describe a production-deployed incremental note generation workflow with quality control and an "agent-edits" feedback loop, reporting measurable reductions in handling time while maintaining agent satisfaction [48]. While their setting is text-based customer support, the architectural implication generalizes to telephony: the automation system can continuously maintain an operator-readable state summary, updated after each high-information turn (e.g., location mentions, corrections), so that if escalation happens the operator receives an immediately actionable snapshot rather than a raw transcript.

More broadly, hybrid human–AI support paradigms emphasize that the goal is not necessarily full replacement but *collaboration*: the AI can act as a "silent assistant" that monitors the interaction and prepares structured help for the human agent [49]. In taxi call centers, this framing is particularly natural because operators are already embedded in workflows where speed and correctness dominate, and where partial automation (e.g., pre-filling pickup suggestions) can produce large efficiency gains even when full autonomy is unattainable.

3.6.3 Measuring operational impact under quality and safety constraints

Finally, HITL strategies must be evaluated as part of an end-to-end operational system rather than as isolated dialogue behaviors. A useful measurement approach distinguishes between *efficiency metrics* and *quality/safety metrics*, acknowledging that improvements in one can degrade the other.

Efficiency and workload metrics. Common indicators include automation containment rate (fraction of calls completed without operator involvement), mean time-to-dispatch, number of turns per successful dispatch, and operator workload reduction. However, these metrics are only meaningful when controlled for call mix and difficulty (e.g., ambiguous pickups vs routine callers).

Quality and safety metrics. Quality must account for error *severity*, not just frequency. For taxi automation, a single wrong pickup can dominate the cost profile relative to many benign clarifications. Therefore, systems should track: (i) wrong-dispatch rate, (ii) cancellation or re-contact rates attributable to automation errors, and (iii) escalation appropriateness (how often escalation prevented an error versus how often it was unnecessary). Recent work on actionable conversational quality indicators (ACQIs) provides a complementary lens: instead of scoring conversations only globally, it proposes turn- and conversation-level indicators that can be tied to concrete improvement actions, helping practitioners identify *which* failure types most harm user experience and how to address them [50].

Closing the loop via operator feedback. An important advantage of HITL architectures is that operator interventions can become training signals: operator-corrected slots, final dispatch outcomes, and escalation reasons form a high-value stream of supervision for improving both recognition components and the escalation policy itself. Systems that explicitly incorporate human edits—either online (immediate correction) or offline (periodic retraining)—can therefore improve over time without sacrificing safety guarantees [48].

Overall, human-in-the-loop fallback should be viewed as a first-class design objective for telephony-based taxi automation: it enables conservative error handling under uncertainty, preserves customer trust, and provides a practical mechanism for continuous system improvement grounded in real operational feedback [33, 46, 47].

Chapter 4

Models and Model Deployment

One of the most pressing issues with the implementation of a user facing agent is the need to keep the response time low (latency). As one of the biggest concerns in this project it was imperative that the model and agentic solutions used keep this time low while not compromising other aspects of the project.

4.1 The balance problem

This was one of the first problems to solve, there exists a need to balance between three different aspects when selecting for LLM model to use as a base to the agents developed:

Latency The response time of the model, this needs to be kept low in order to keep a proper conversational flow between agent and user.

Usability/Intelligence The ability of the model to handle the instructions, tools and flexibility given to it as an agent.

Cost The cost of running the models, this being energy, money or resources.

There is a need to balance between these three, but this balance will look different depending on the objective at hand. To solve these problems there are only two actionable ways of implementation, local deployment or use of a service, this is then compounded by the choice of model that will serve as base to the agents developed.

4.2 Local deployment

Local deployment is defined by an upfront investment cost followed by low cost of operation and usually limited computing power which limit both the latency and usability. This makes it difficult

to use this setup for the more complex and competent agents. However, that still makes it useful to run simpler models for less computational expensive tasks.

4.2.1 Hardware Setup

In almost all implementations of a local model, for the purpose of agent building or otherwise, there is a need for fairly strong hardware. Two computers were used for this project a custom-built pc equipped with an AMD Ryzen 9, an Nvidia GeForce RTX 5090 and 64 GB of RAM, and a Dell pro max gb10.

Two different machines were used due to the different strengths that they processed. The custom-built pc presented a strong deployment environment with its GPU, but remained limited in the size of model that it could handle due to the present VRAM limit. A GPU's integrated RAM memory enables it to fasten the inference speeds of a model, this memory serves as a size limit to the models that can be deployed and a sharp drop in performance occurs when it is passed and there is a need to use the non GPU memory. This made this pc the best option for smaller models that must have low latency.

The Dell pro max gb10, while also possessing a strong GPU, has a 128 GB unified ram pool. This coherent pool is shared between the GPU and the CPU which allows for the full use of the machine's RAM without any sharp drops in performance. This serves as a trade-off to the fact that this type of architecture makes the GPU memory access inherently slower as it is less integrated with the GPU. This machine is purpose built for the deployment of larger scale local models and was the best option of the two for stronger models at the cost of higher latency.

4.3 OpenAI as a service

Local deployment isn't suitable for all applications, for the bigger and more complex models needed for more capable agents the only feasible option is the use of remote model services. These services are provided by many companies with each having pros and cons depending on what is needed. Of all the biggest and more reliable providers, at the time of development, OpenAI was the only service that could provide a model needed to make a real time voice agent.

4.4 Models

As stated previously, the balance of the three main attributes is an essential part of development and served as a guide for the model choice. Each different facet of the project called for different priorities depending on function and intent.

In a text based agent the need to respond quickly, while important, isn't the biggest concern as the nature of text messaging leaves a generous margin to respond, this lets the model selected be cheaper to use and more intelligent.

The Voice model, on the other hand, has to have a low latency to be useful for a user facing agent, the agent needs to be able to speak in quick succession and in a way that mimics natural

language as to support a conversation. This emphasis on low latency forces the cost to be higher than preferable and the Intelligence of the model to be lower than ideal.

4.4.1 Text Models

For the text agents two models were selected as a base, these were selected in order to fulfill a specific composition of needs depending on its function.

4.4.1.1 Qwen3:30B

For the first text agent there was a need for a model capable of handling simpler and faster tasks while maintaining low operational cost. For this purpose the Qwen3:30B model was selected and deployed locally through Ollama.

The model was hosted through an Ollama instance, allowing the agent to access the model through local API calls instead of relying on external services. This reduced operational costs significantly and removed dependency on external providers for simpler interactions.

Qwen3:30B is a general purpose language model designed for conversational interaction, instruction following and text generation tasks. Within the agent the model was mainly responsible for simpler requests, shorter conversations and tasks that required fast response times.

One of the main advantages of this model was inference speed when compared to larger and more capable models. Since many user interactions did not require advanced reasoning, using a lighter local model allowed the system to respond more quickly while reserving larger models for more demanding tasks.

In order to make local deployment possible both Qwen3:30B and Gemma-4-31B-Heretic had to be quantized. Quantization reduces the numerical precision of the model weights, lowering memory usage and computational requirements while allowing the models to run on more limited hardware. This made it possible to deploy both models locally without the need for enterprise level GPU infrastructure.

The local deployment also allowed greater control over the system configuration and reduced token based operational costs associated with remote APIs. However, this approach introduced hardware limitations as running a 30B parameter model locally requires considerable computational resources, particularly GPU memory.

Compared to larger language models, Qwen3:30B also demonstrated weaker reasoning capabilities in longer or more complex interactions. This made it less suitable for tasks requiring deeper contextual understanding or more advanced conversational behavior.

Despite these limitations, the model provided an effective balance between speed, cost and capability for lightweight interactions and simpler processing tasks within the agent.

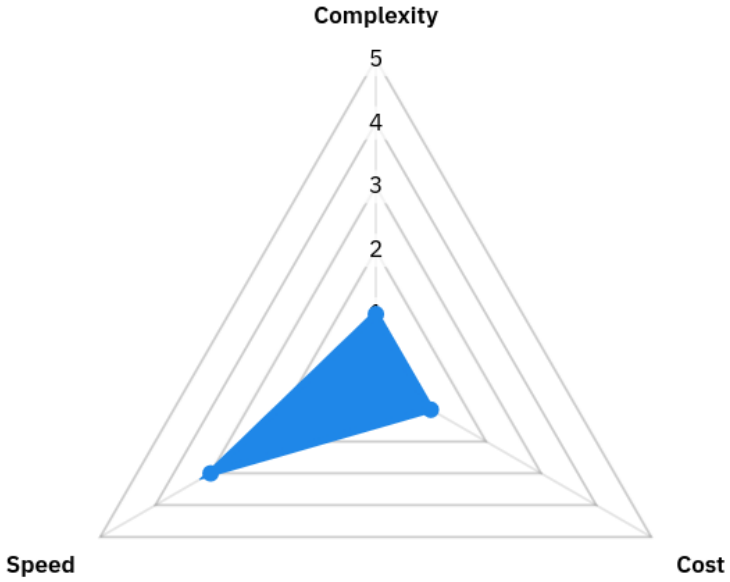


Figure 4.1: Qwen3:30B model statistics

4.4.1.2 Gemma-4-31B-Heretic

For more complex interactions a second model was integrated into the same text agent, the Gemma-4-31B-Heretic model. While Qwen3:30B was responsible for faster and simpler tasks, Gemma-4-31B-Heretic was used in situations that required stronger reasoning capabilities and more detailed responses.

The use of two different models inside the same agent was intended to balance latency and intelligence more effectively. Simpler requests could be processed quickly using the lighter model, while more demanding tasks could be routed to the larger model even if this resulted in increased response time.

Gemma-4-31B-Heretic demonstrated stronger contextual understanding and better performance in longer conversations and multi step reasoning tasks. This made it more suitable for handling complex prompts, detailed explanations and interactions requiring more consistent conversation continuity.

Like Qwen3:30B, the Gemma-4-31B-Heretic model also had to be quantized in order to be used locally. Without quantization the memory and computational requirements of the model would exceed the capabilities of the available hardware. Although quantization can slightly reduce model accuracy, the reduction in resource usage made local deployment practical while still

maintaining acceptable performance.

The larger size of the model however introduced higher computational requirements and slower inference speeds. Compared to Qwen3:30B, response generation was noticeably slower, making it less appropriate for lightweight interactions where quick responses were more important than advanced reasoning.

Like the Qwen model, Gemma-4-31B-Heretic was deployed locally. This avoided recurring API costs and provided greater control over deployment and configuration, but also increased hardware requirements considerably due to the size of the model.

Using both models together allowed the agent to dynamically balance performance and capability depending on the complexity of the request. Faster interactions could be handled efficiently by Qwen3:30B while more demanding tasks were delegated to Gemma-4-31B-Heretic. This architecture improved the overall responsiveness of the system without sacrificing the ability to handle more advanced interactions when necessary.

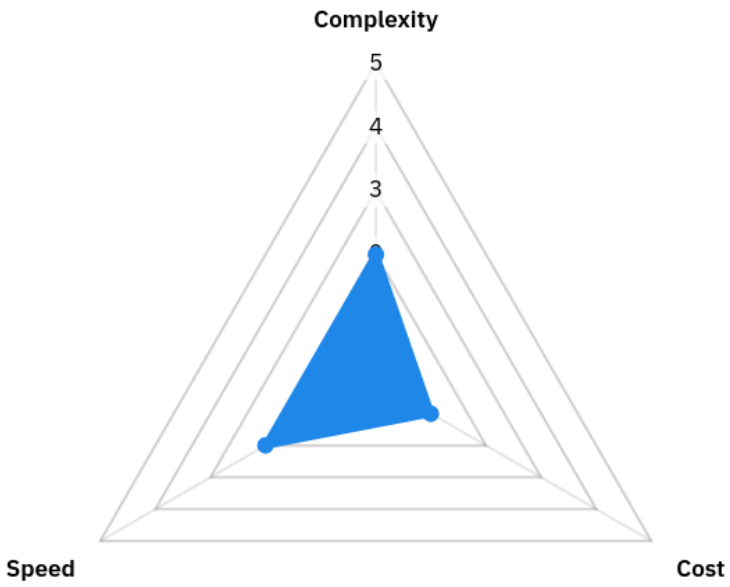


Figure 4.2: Gemma-4-31B-Heretic model statistics

4.4.1.3 GPT 4.1-mini

For the second agent there was a need for a model capable of natural language conversation, this required a more capable LLM model. For this selection the GPT 4.1-mini model was chosen. This

model provided a balance between response time, cost and reasoning capability, making it suitable for a conversational text agent while still remaining viable for continuous use.

GPT 4.1-mini is part of OpenAI's GPT 4.1 family of models and is designed for general purpose conversation and instruction following. Compared to smaller and faster models it is capable of handling more complex prompts and maintaining more coherent conversations over longer interactions. This was important for the text agent as it needed to maintain context while also following system instructions and using external tools.

One of the more important features for this project was the support for large context windows. This allows the model to process longer conversations and larger prompts without immediately losing previous information. In practice this made it possible to maintain conversation history while also using detailed system prompts to guide the behavior of the agent.

Another important feature was support for structured outputs and tool calling. Through the OpenAI API the model can generate outputs that follow predefined formats such as JSON structures and tool calling. This made communication between the model and backend systems more reliable and allowed the agent to interact with APIs and memory systems with fewer formatting errors.

The model also supports multimodal inputs through the OpenAI platform, allowing both text and image information to be processed in the same request. While these features were not used directly in the implementation of the project, they provide useful possibilities for future development, especially for WhatsApp based applications involving image analysis or document processing.

The model is accessed remotely through OpenAI API services instead of being deployed locally. This removes most of the computational load from the local machine and avoids the need for dedicated GPU hardware. Requests are processed on OpenAI infrastructure and returned through HTTPS API calls. This simplified deployment considerably and made it easier to support multiple simultaneous users.

One of the main disadvantages of this approach is operational cost. The API pricing model is based on token usage, meaning that larger prompts and longer conversations increase the cost of operation. Because of this there was a need to carefully manage prompts and conversation history in order to reduce unnecessary token usage while still maintaining conversation quality.

Latency was also an important factor in the model selection. Although GPT 4.1-mini is optimized for faster inference than larger models, it is still slower than lightweight models specifically designed for very low latency tasks. For a text based application this tradeoff was acceptable as the improved conversation quality and reasoning capability justified the increase in response time.

Prompt engineering also played an important role in the implementation of the agent. System prompts were used to define the behavior, restrictions and personality of the model. Careful prompt structure helped improve consistency and reduced undesirable responses such as formatting issues or hallucinated information.

The choice of GPT 4.1-mini represented a balance between capability and deployment limitations. Larger models provided better reasoning capabilities but increased both cost and latency,

while smaller models were faster and cheaper but less reliable in longer interactions. GPT 4.1-mini provided the most balanced solution for the requirements of the project and allowed the development of a responsive and reliable conversational agent.

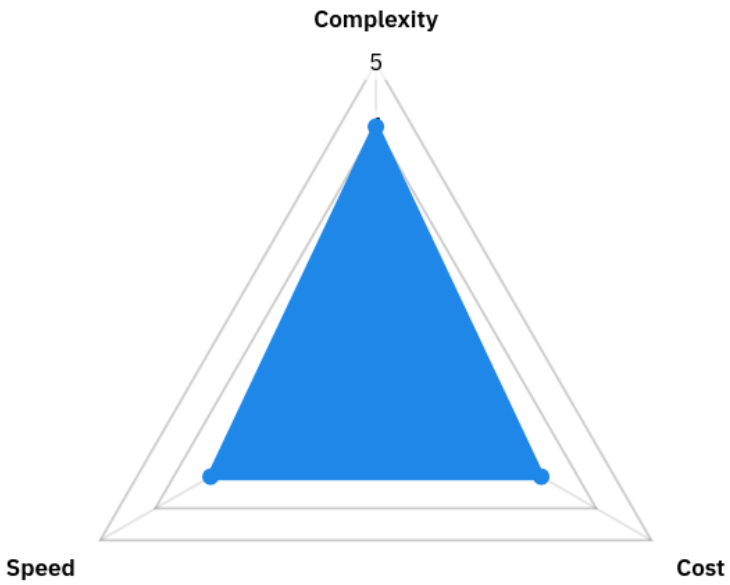


Figure 4.3: GPT 4.1-mini model statistics

4.4.2 Audio Models

For the voice agents there was a need for a different set of models than the ones used for the text agents. Unlike text based interactions, voice conversation requires extremely low latency in order to maintain a natural conversational flow. The processing pipeline also becomes more complex as the system needs to process audio input, convert speech into text, generate a response and converting the response back into audio.

4.4.2.1 Whisper

For speech recognition the Whisper model was selected. This model is an Automatic Speech Recognition (ASR) model designed to convert spoken language into text. In the context of the voice agent this was one of the most latency sensitive parts of the system, as delays in speech transcription directly affect the responsiveness of the conversation.

The main reason for selecting Whisper was its balance between transcription speed and recognition accuracy. While larger speech recognition models may provide better accuracy, they also introduce additional latency which negatively impacts real time interaction. Whisper was capable of maintaining sufficiently accurate transcription while still operating fast enough for conversational usage.

Another important factor was the ability of the model to handle natural speech patterns. In voice conversations users frequently interrupt themselves, change speaking speed or pause unexpectedly. The speech recognition model therefore needed to remain robust to incomplete sentences and inconsistent pacing while still producing usable transcriptions.

The model was deployed locally which reduced the dependency on external services and avoided additional network latency during transcription. Since speech recognition is performed continuously during voice interaction, local deployment also helped reduce operational costs compared to constantly sending audio streams to external APIs.

Despite these advantages, local deployment introduced hardware requirements. Speech recognition models require continuous processing of audio streams and therefore benefit significantly from GPU acceleration. This created additional computational overhead on the host machine and limited scalability compared to fully remote solutions.

Another limitation was transcription accuracy in noisy environments. Background noise, microphone quality and unstable speech patterns could reduce transcription quality and occasionally introduce incorrect outputs. Because the language model depends directly on these transcriptions, recognition errors could propagate into the conversational response generated by the agent.

Even with these limitations, Whisper provided an effective balance between speed, resource usage and transcription quality, making it suitable for the requirements of the real time voice agent.

4.4.2.2 Realtime-GPT v1

For the second voice agent the Realtime-GPT v1 model was selected. Unlike the text based agents, this system required direct real time voice interaction with the user, making low latency the highest priority in the model selection process. In voice conversation even small delays become noticeable and negatively affect the natural flow of interaction.

Realtime-GPT v1 is optimized for real time conversational applications and continuous audio interaction. Compared to larger language models it prioritizes fast response generation and conversational fluidity over advanced reasoning capability. This made it suitable for a voice based assistant where maintaining a natural speaking rhythm was more important than handling highly complex reasoning tasks.

The model was operated through OpenAI realtime API services using a SIP connection. The Session Initiation Protocol (SIP) was used to establish and manage the audio communication session between the voice agent and the external telephony system. This allowed the agent to participate in real time audio calls while maintaining a continuous bidirectional audio stream with the OpenAI realtime infrastructure.

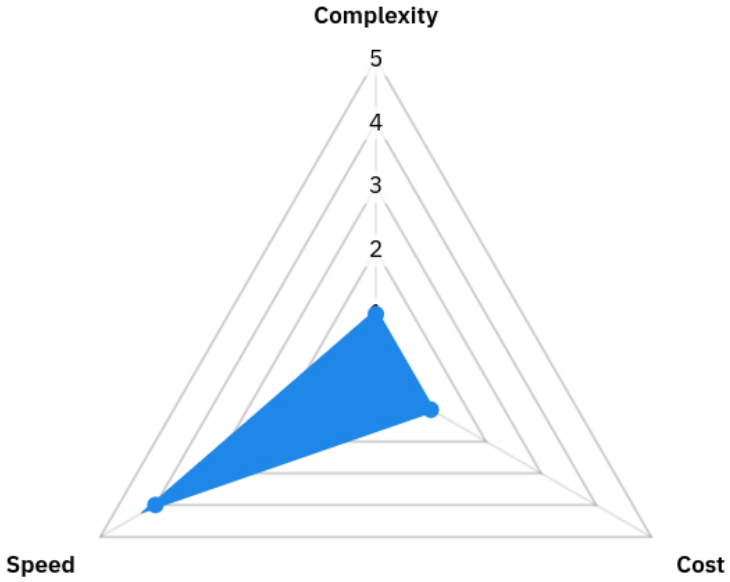


Figure 4.4: Whisper model statistics

One of the main advantages of this architecture was the ability to process speech and generate responses incrementally. Instead of waiting for complete message generation, the system was capable of streaming responses continuously, allowing audio playback to begin while the remainder of the response was still being generated. This reduced the perceived latency considerably and helped maintain a more natural conversational flow.

Another important characteristic was the model's ability to maintain short term conversational context during continuous interaction. This allowed the voice agent to preserve conversational continuity across multiple exchanges while still responding quickly enough for real time communication.

Using the realtime API through SIP also simplified the deployment of the voice agent. Instead of hosting a large conversational model locally, most of the conversational processing was handled remotely through OpenAI infrastructure while the local system focused on orchestration and call handling.

One of the disadvantages of this approach was dependency on network quality and API availability. Since both audio streaming and response generation depended on remote services, unstable network conditions could directly affect call quality and response latency.

Another limitation was operational cost. Continuous real time audio streaming and response generation through API services creates a higher operational cost than text only interactions. This

made efficient session management and prompt control important parts of the implementation.

Compared to larger text focused models, Realtime-GPT v1 also has lower reasoning capability in more complex tasks. This was however considered an acceptable tradeoff since the primary objective of the system was maintaining responsive and natural voice conversation rather than advanced reasoning performance.

The selection of Realtime-GPT v1 represented a compromise between conversational quality, latency and operational cost. Larger and more capable models could provide better reasoning performance but introduced response times that were unsuitable for natural voice interaction. Realtime-GPT v1 provided the best balance for the needs of the project and allowed the implementation of a responsive real time voice agent.

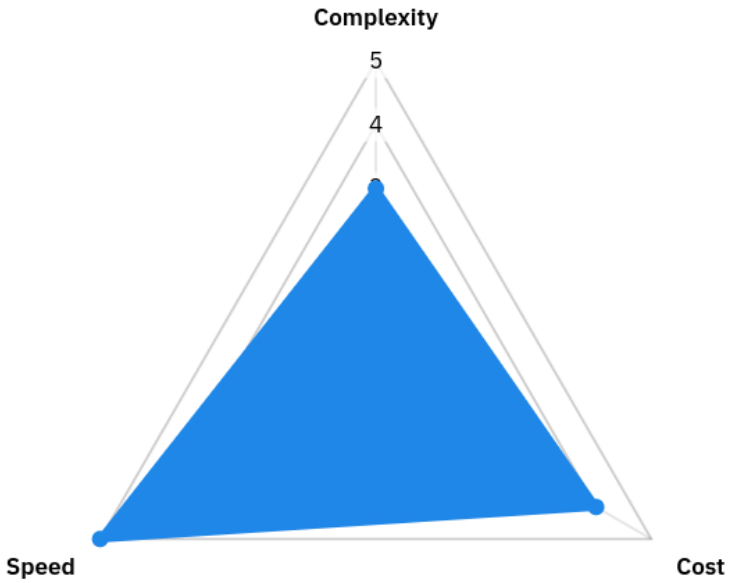


Figure 4.5: Realtime-GPT v1 model statistics

Chapter 5

WhatsApp Helper Agent

While the first things that comes to mind when talking about AI agents are the full conversational agents (Agents that use natural language and speak directly with the user) these aren't the only agentic approach, and in some cases aren't even the correct approach. Depending on the problem at hand there is space for highly structured agents that use the natural language interpretation capabilities of LLMs to coordinate data flows and scripts in a reliable and consistent way, while maintaining the intelligence of the agent.

5.1 Auxiliary tools and structures

This agent's architecture was devised in a way to make the change from the previous system to the agentic approach as seamless as possible for the user, this lead the agent to be developed over the existing architecture in a way that preserves the functionalities and tools already implemented.

The WAH Agent retains the tools of the old system:

1. Taxi request handling
2. Live service notification
3. Cancellation handling

While adding new functionalities:

1. Natural language handling for both taxi and cancellation requests
2. User language detection
3. Auto translation to PT-PT for the operator interface
4. Gives the ability for the operator to translate messages to the client's language

In regard to the operator interaction, the previous system was overly obtuse. Operators had to reach out using an individual WhatsApp number to each and every client that needed human interaction this made the process of receiving and sending operator messages tedious and introduced a need to handle multiple message avenues and numbers.

To solve this problem and to centralize all the message avenues a chat web interface was created. This centralized dashboard displays incoming messages and conversations in a unified interface, enabling operators to monitor multiple clients simultaneously and intervene in escalations without context switching across individual WhatsApp numbers.

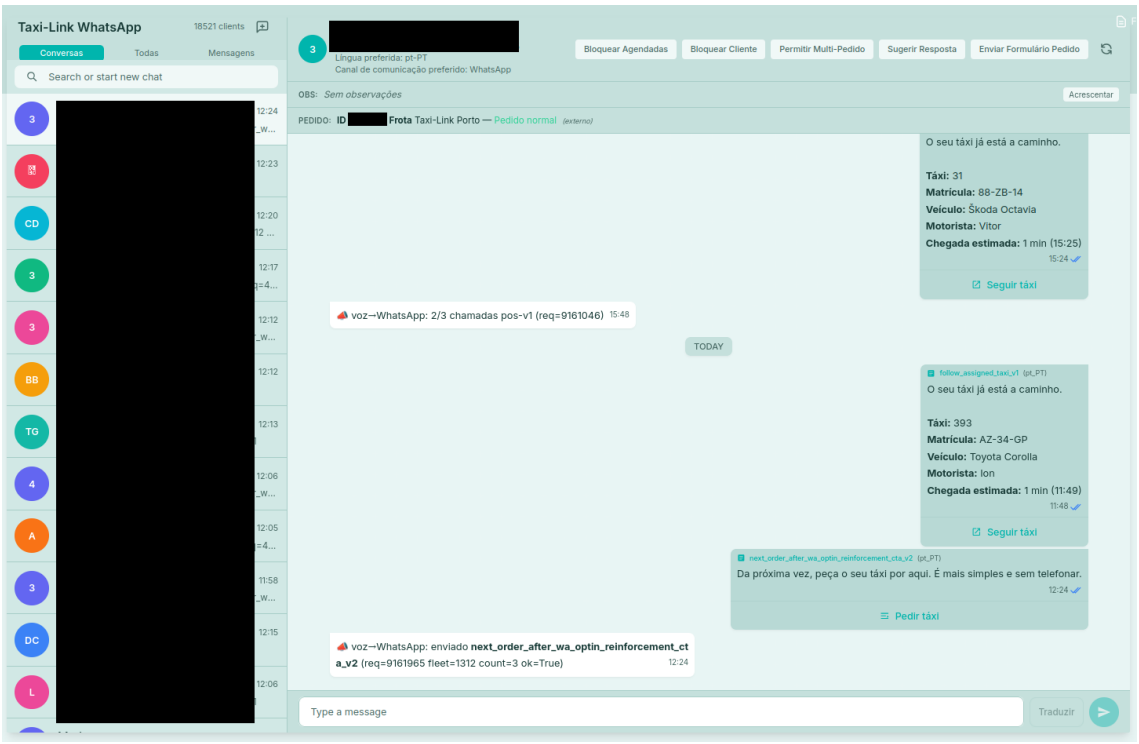


Figure 5.1: Web chat interface used by operators to monitor and intervene in WhatsApp conversations

5.2 Multi-model agent design

The WhatsApp Helper Agent (WAH Agent) was developed to bring more flexibility to the previously implemented flow, this was achieved by implementing a two-step intent classification that leverages the language understanding of LLMs to steer the flow of the WhatsApp interaction. The Agent also works in the background in a way that helps both the user and the possible operator by using tools mid-conversation to bridge language barriers. This serves as an expansion of the previous system instead of a total overhaul making it more flexible and responsive while not giving the agent a high level of independence.

5.2.1 Internal structure

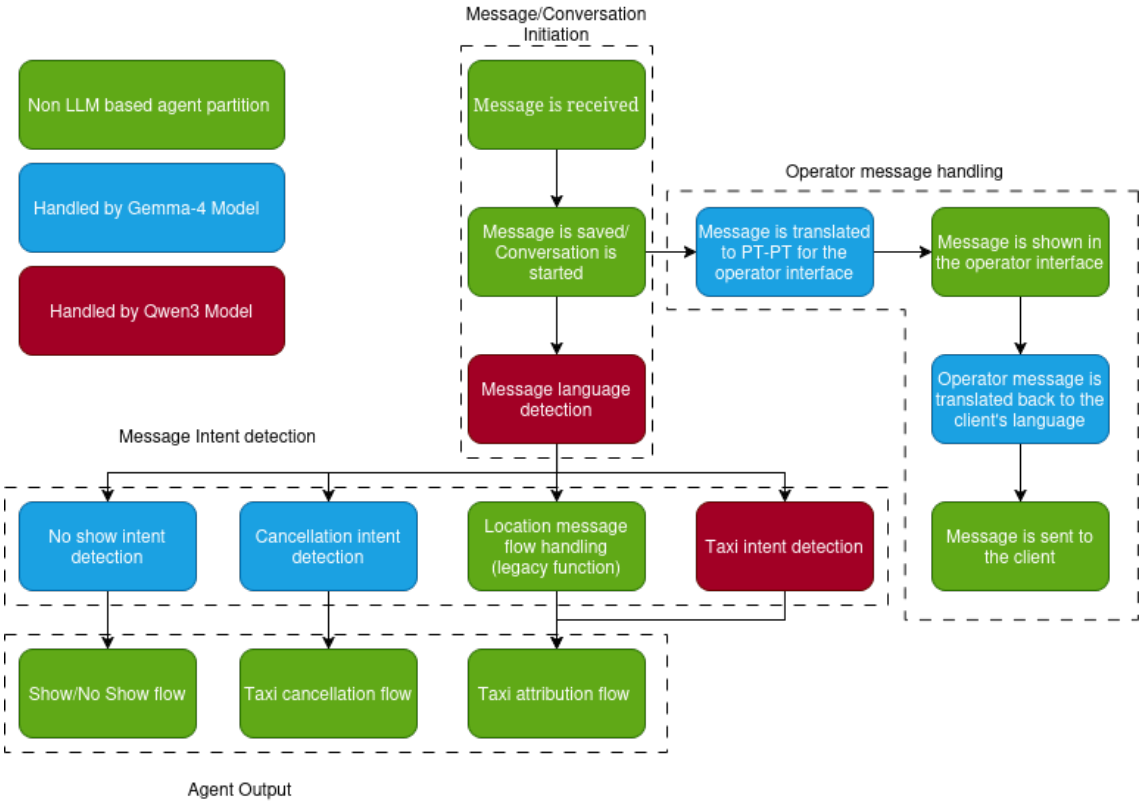


Figure 5.2: WAH Agent Flow

The WAH Agent is structured by three interconnected parts, two models that work in parallel, Qwen3:30B and Gemma-4-31B-Heretic, and the pre-existing processes of the service. The WAH agent works by deciding which part of the pre-existing code should be executed depending on the interpretation of the intent of the message received made by one or both of the internal models.

The Qwen3 model is an extremely low latency model that is deployed locally and is able to do low complexity classification and decisions while the Gemma-4 model is used for more complex tasks and more ambiguous or uncertain message intentions.

The agent flow can be divided into 4 distinct sections.

5.2.1.1 Message/Conversation Initiation

This section contains all the logic that is used to receive WhatsApp messages from the Meta API, the message is received in a webhook, if that client doesn't have any active trips, and it was the first message in some time, a new Conversation is created this is an entry in a database that is used to track groups of messages and interactions in a way that makes it easier to catalog, analyze and utilize for all the separate functionalities of the services.

Each message received is saved, attributed to a conversation, and analyzed as to quickly infer the language used by the client, this is saved and used by the rest of the agent to configure the

correct forms to be used. After saving and categorizing the message received the model sends it to intent detection. Parallel to this process the message is sent to operator message handling.

5.2.1.2 Message Intent Detection

The main function of the agent is the detection of the intent of the received message, that is done in this section. The message is classified into one of four different categories, Show/No-Show, Cancellation Intent, Taxi Request and Location Message.

Show/No-Show corresponds to the logic of when a taxi request was made and the taxi already arrived at the location, but the client wasn't at the location, this could be by any reason, was waiting in the wrong place, is inside the building, already found an alternative to the taxi, doesn't want to take the taxi anymore, etc. and the taxi driver cancels the trip. When this happens there is a time window when the client can justify the reason of the No-Show and state that they still want the trip, when that happens the agent makes a new request with the same taxi that was assigned to the cancelled trip.

Cancellation Intent corresponds to the type of messages that show that the client wants to cancel the requested trip, this can only be done to clients that have an active trip in progress that, was scheduled to a later date or a trip that doesn't have an assigned taxi yet.

Taxi Request corresponds to when the message is seen as a greeting or an explicit taxi request, as this classification is simpler than the two before and doesn't present any prerequisites it is done with the faster model as to be as responsive as possible.

Location Message is a legacy system that was preserved as to not interfere with old users, this system uses WhatsApp's location message as a way to initialize the taxi request flow with an already defined location.

5.2.1.3 Agent Output

The Agent is unable to make custom responses or responding in natural language, this means that it has to rely on hand made WhatsApp flows that collect the data needed for all the intermediate steps and communicates information to the client.

WhatsApp flows consist in templated messages with defined structures, these can be forms that receive data in text boxes, pre-made cancellation messages, requests for location, etc. These are the main ways conventional automated systems communicate using the WhatsApp API. Many of these flows were ported from the previous implemented system and serve as a base where agentic decision-making can be introduced.

5.2.1.4 Operator Message Handling

The call center operators consist in the extremely important human fall back element of any automated system, they consist in the best way to resolve problems or give information that the system is unable to resolve. The WAH agent in conjunction with the chat interface also function to make it easier for the operator to jump into unexpected problems and communicate with clients that

previously they wouldn't be able to respond to. The agent serves as an intermediate step between the client and the operator that translates the interactions as to promote a better system.

At any point in any interaction between a client and the agent if the operator so chooses they are able to override the agent and communicate directly with the client at which point the agent stops functioning as a decision maker and starts working as an intermediate between both parties.

5.3 Integration with existing systems

While a detailed explanation of the internal systems of Geolink lda. is outside the scope of this dissertation, the environment in which the agent is integrated must be briefly described. Geolink lda. operates a large-scale taxi dispatch service comprising roughly 4000 taxis organised into 30 distinct fleets, covering the main cities of mainland Portugal as well as the Autonomous Regions of the Azores and Madeira. Each fleet corresponds to an operationally independent unit with its own coverage area and dispatch logic, while sharing the common backend infrastructure that handles request intake, taxi attribution and service tracking. The WAH agent was integrated on top of this existing infrastructure rather than replacing it. Incoming WhatsApp messages are routed through the agent, which then interacts with the same dispatch backend, booking and cancellation mechanisms, and client records already in use across the service. This fleet-aware integration allows the agent to operate transparently over a heterogeneous, multi-fleet environment while preserving full compatibility with the operational systems, and it is what enables the agentic decision layer described in the previous sections to be deployed without disrupting the established dispatch workflow.

Chapter 6

WhatsApp Conversational Agent

The main alternative to the highly structured architecture of the WAH agent is a fully conversational agent, where all the decisions and output are done directly by LLM. The WhatsApp Conversational Agent (WAC agent) was the agent developed with this approach. By utilizing commercially available services, custom-made tools and structures it was possible to create an agent that is flexible, intelligent and able to speak in natural language.

6.1 LLM-based agent design

As a conversational agent, the WAC agent uses a LLM model as the core orchestration element. The model chosen to serve as the core of the agent was OpenAI's GPT4.1-mini for its high speed, high reasoning, and low cost, this was necessary for the agent's ability to communicate, use the necessary tools, retain pertinent knowledge and maintain a low response time in the user/agent interactions.

6.1.1 Tools and functionalities

For the WAC agent to be able to interact with all the functionalities of the Geolink lda. internal systems, get pertinent information and handle the user requests it had to be equipped with a multitude of tools and functionalities. These were divided into five groups, Booking Tools, Coverage and FAQ, Communication/Handoff, Conversation Control and Status and History.

6.1.1.1 Booking Tools

These are the tools that are used for the main function of the WAC agent, they bridge the agent and the internal booking mechanisms of the Geolink system, and they consist in:

- Agent Dispatcher: The Agent Dispatcher tool is the main flow tool, it is called after all the trip information is collected by the agent. It receives, the dispatch date, a sting containing the time in ISO 8601 when the trip is scheduled for a later date or a None for immediate dispatch, the pickup address, a JSON string with all the information needed to set the pickup



Figure 6.1: WAC Agent Flow

location, it includes latitude and longitude, address name, door number and some other internal information needed to set the location, destination address, same JSON structure as the pickup location with it being optional and taxi/service preferences that are sent to the service server as tags to filter the assigned taxis. It returns a string to inform the agent of the success status of the dispatch.

- Cancel Trip: The agent tool responsible for the cancellation of the latest trip of the user, not every trip can be cancelled, as there are restriction in place to stop users from cancelling when the service already has an assigned taxi. This tool differentiates from the other cancellation tool in the aspect that it is faster to prevent erroneous trips from being finalized. It takes a cancellation reason as input and returns a string to inform the agent of the success status of the cancellation.

- Cancel Booking: Tool used to cancel bookings, trips planned for a later date, as this is a less time sensitive function, and it needs a higher granularity to cancel between more than one booking, this agent function needed a separate tool to handle. It receives as input a booking ID and fleet ID, information acquired by another tool call and returns a string to inform the agent of the success status of the cancellation.

- Update Service: Tool used to update trip requests that haven't yet been finalized, without an attributed taxi, these can be both normal trips and bookings. As input the tool takes the

request number, the fleet and the information to be updated. Returns a string to inform the agent of the success status of the update.

6.1.1.2 Coverage and FAQ

When dealing with other services or flow both the user and the agent can have questions that need answering, many of these can be kept in the initial prompt for quick access, but in questions that need dynamic answers tools need to be used:

- Check Covered Area: This tool checks if a given latitude and longitude is covered by the Geolink services, for that as input it receives the latitude and longitude of a point to check and returns a JSON string with covered boolean value and if true with fleet id and region.
- Check Covered by Name: Does the same as Check Covered Area with the difference of accepting a location by name instead of coordinates, and with that tries to check if the location is covered by the service, it isn't always possible to be one hundred percent certain of what the location is as the name can indicate multiple locations, might be unclear or unrecognizable by the service, to mitigate this the tool tends to lean on the side of caution and unless certain it won't return true.
- Get Current Time: A model that doesn't have any tools is unable to know the time and date, this information is useful for the purposes of setting reservations that use time relative language, this function allows the model to know the current date and time. It returns the current date and time.
- Get Price Info: Consist of a placeholder tool that would call an estimation service that the Geolink services are planning to introduce. At the moment returns an explanation string that says the agent still can't make estimations on the price

6.1.1.3 Communication/Handoff

The WAC agent isn't always able to resolve all the possible problems that might occur, and it also isn't able to control the behavior of the taxi drivers and services, to mitigate that these tools are used as an intermediate between the user and the pertinent parties:

- Pass Message to Taxi: This tool lets the agent function as a message way between the user and the taxi driver using the Geolink services to display a message to the motorist app. Receives as input the message and returns a string with the status of the process.
- Pass to Human Agent: Stops the internal flow of the agent, stopping the agent from responding and alerts an operator of the problem at hand. Receives a reason for the alert and returns a string with the status of the process.

6.1.1.4 Conversation Control

The WAC agent has to be able to manage the end of the conversation flow, to do that it has to be able to end a conversation and understand when it's not speaking with a human. To do that the following tools were implemented into the agent:

- Mark Auto Reply: If the agent thinks it isn't speaking with a human it will use this tool to end the conversation. When called the tool will close the conversation and stop future conversation with the user.
- Acknowledge Thanks: It is the way the model is able to finish the conversation with a user. When called ends the conversation and sends a goodbye message.

6.1.1.5 Status and History

The agent needs to be able to inform itself and the user of previous trips as well as keep track of statuses of active requests:

- Get Active Request Status: This tool is used to get the status and information of an active request of the user. The agent needs to input a request id and a specific fleet as to stop any possible ambiguity problems. This tool then returns to the agent a JSON string containing internal information about the specific trip.
- Get Last Service: This tool returns the request id and the fleet of the last service that is associated with the user, as seen in previous tools this information is needed for many other agent functionalities.
- List Client Bookings: Same as the Last service tools with the caveat that it returns all the bookings of the user as a list.

6.1.1.6 Text and Location Messages

The agent also needs to be able to communicate with the user, this functionality is so intrinsic to the agent that it isn't incorporated as an agent tool. All the agent's model output is considered and expected to be a message that has to be delivered to the user whilst being able to make use of the agent tools in parallel, to be able to achieve this the model output is structured to be a JSON string with two fields, message type and message body, the message type can be "text" or "location" depending on the WhatsApp message type the model intends to send, "text" if it is a simple message that only contains text and "location" if the message should include a WhatsApp location request and the message body contains the actual message text that the agent intends to send to the user.

This structure helps the agent with the problem of over-reliance on agent tools, as the presence of a tool interaction in every turn of conversation, if sending the messages was a tool, lead to higher incorrect tool use chances.

6.1.2 Prompting

Other than the tools an agent has access to and the model it is based on the third main aspect that contributes to the agent's behavior is its initial Prompting. Initial Prompting consists of a comprehensive set of rules and guidelines that are used in the initialization of an agent to control its behavior, output, tool usage and interactions. Its structure is essential to the good functioning of the agent and its integrity is one of the biggest contributors of behavior/flow problems in any agent.

While the exact composition of the prompt used for the WAC agent is outside the scope of this paper the overall structure of the prompt is an important element to explain and explore. The prompt can be divided into seven groups of instructions, Identity, Guardrails, Booking Workflow, Tools and Escalation, Communication, Ground Facts and Examples.

- Identity: The Identity portion of the prompt controls how the agent sees itself, it's a single paragraph of mission critical information that shapes all the agent's interaction, it contains what the model is, WhatsApp Conversation Agent working for Geolink lda. , what its functions are, book taxi trips, answer questions, handle trip statuses, etc. and set the scope of the agents function.

- Guardrails: The Guardrails are an extremely important portion of the prompt as it introduces the behavior/procedure the agent must take and maintain in order to protect itself from intentional attacks that target is logic and how it should behave with the data it has access to in order to prevent data breach issues.

- Booking Workflow: As the main function of the agent the booking flow is the most elaborate and comprehensive containing step-by-step instructions that dictate the workflow the agent should take. The agents start by greeting the user, then it determines the intent of the user, if the user intends to book a trip the agent should start the data collection flow by asking the time and date when the user needs the taxi, then the agent is supposed to get the pickup location using a "location" message followed by asking if the user wants a destination and if it is needed asking for it, finally it asks if the user wants has any preferences on the taxi and makes the booking with the data it collected. This portion also contains notes that clarify behavior in edge cases and in special circumstances.

- Tools and Escalation: This portion explains to the agent all the tools it has access to and their functionalities as well as listing all the different scenarios that would require escalation to a human operator.

- Communication: Communication governs how the agent should communicate with the user, it governs the messages tone and length as well as introducing the language logic the agent should take, the agent should use the user's language. This portion also contains some warning to the agent about internal language problem that should be cautioned against when

dealing with possible data that is received in different languages than the one the user is using.

- Ground Facts: These consist in a fixed list of important information that is given to the agent, so that it is able to answer common questions that the user might have or that it can use for its functionalities.
- Examples: Small list of interaction examples that increase the understanding of the model in how it should behave.

6.2 Integration with existing systems

In contrast with the WAH agent the WAC agent isn't made to overlay the existing system, it instead replaces it entirely. The flow is redirected to the WAC agent flow whenever a message from a client assigned to this agent is received letting it handle all the internal steps. The agent is still interconnected with the internal infrastructure of the Geolink lda. services to be able to make, check and overall interact with taxi trips as well as check internal client information that is stored in the system. It also is integrated with the WhatsApp and web chat interfaces used by the WAH agent in order to be able to interact with users and to be able to be monitored by human agents while in use. The WAC agent then represents a parallel flow to the WAH agent that is used in a user by user basis as an alternative to the simpler agent.

Chapter 7

Napoleão Voice Systems: IVR and RTN Agent

The Napoleão call manager system consists in a call auto-answering system that is activated for users that have a predictable pickup location, be that because they are extremely consistent or because they represent a specific place such as when a hotel calls a taxi for a guest. This service makes it so that a taxi can be called to a specific place with a press of a button on the phone and without any human interaction needed.

The auto-answering system uses a dynamic voice synthesization program to answer calls and give the option of fast-tracking the process of calling a taxi or calling an operator by choosing to press one of two buttons.

7.1 Integration of Voice Detection

One of the most intuitive ways to improve this kind of service is the ability of a user to use their voice to select the options instead of using buttons, this feature would be helpful for many types of situations where a person is unable to currently click a number on the phone.

This was implemented using OpenAI's local model Whisper and integrated into the existing Asterisk system.

7.1.1 Asterisk

Asterisk is an open-source telephony platform used to manage and automate telephone communications. It functions as a Private Branch Exchange (PBX), allowing the creation of call flows, Interactive Voice Response (IVR) menus, call routing logic and integrations with external services.

Within the Geolink infrastructure, Asterisk is responsible for handling incoming calls and controlling the different automated call processes. Its flexibility and support for external integrations make it suitable for the implementation of services such as the Napoleão call manager, where call handling logic can be customized to meet specific operational requirements.

7.1.2 Internal Structure

The expansion of the Napoleão system consisted in the introduction of a new voice detection system. To accommodate this change the dynamic voice synthesization text was modified to reflect the new voice detection possibilities, to call a taxi say "taxi" or say "operador" to call an operator after a sound is played. The response is then recorded for a fixed time and this recording is then processed. This specific process was used instead of a direct streaming solution because of the inability of the model to transcribe in real-time interactions.

After receiving the recording the audio is inputted into the model with the specific language setting of Portuguese and the output is post processed and a confidence value of the transcription is made for the two different options. If the confidence value surpasses a specific threshold is returned to the Asterisk system to be used in the rest of the flow, in the case of a transcription not surpassing any of the thresholds a fallback option is reached, and the user is redirected to an operator.

Due to the local nature of the model and its small size the transcription process is extremely fast, averaging processing times of less than half a second.

7.2 Realtime Napoleão Agent

As an alternative for the Napoleão system of auto answering the Realtime Napoleão Agent (RTN Agent) was developed. This agent represents a more powerful and flexible approach than the Napoleão system as well as constituting a call based alternative to the WAC agent. The RTN agent functions as a low latency and fully conversational agent capable of booking taxi trips in a voice to voice environment.

7.2.1 Low latency voice agent design

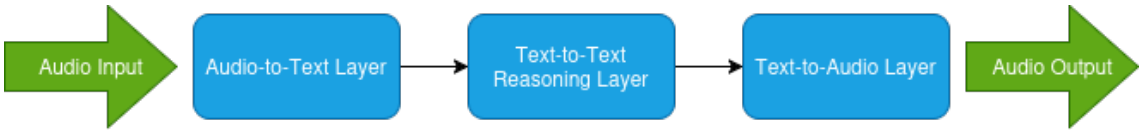


Figure 7.1: Three Layer Conversation Agent

For most of the existence of conversational agents the only option was to chain together three different systems, an audio transcription model, a text-to-text reasoning model and an audio synthesization layer that would translate the text into an audio output. This solution while practical were slow and lacked the back and forth quality that real conversations have. Realtime GPT is one of the first commercially available audio-to-audio models, that being a model that can directly process an audio input, reason and present an audio output. This approach has the advantages of keeping the latency extremely low and reducing the information lost that normally would occur in both the audio-to-text translation process and the text-to-audio process with the main disadvantage of having lower reasoning capabilities for a model of similar size.

Another characteristic of the model is its ability to process the audio input in a continuous way instead of chunks, this paired with the low latency made it the ideal choice to use as the base of the RTN agent, as the objective of the agent is its capability to have a realistic conversation with the user where there is a need for fluidity and fast responses and the lack of reasoning power of the model doesn't affect the performance of the agent.

7.2.2 Agent Structure

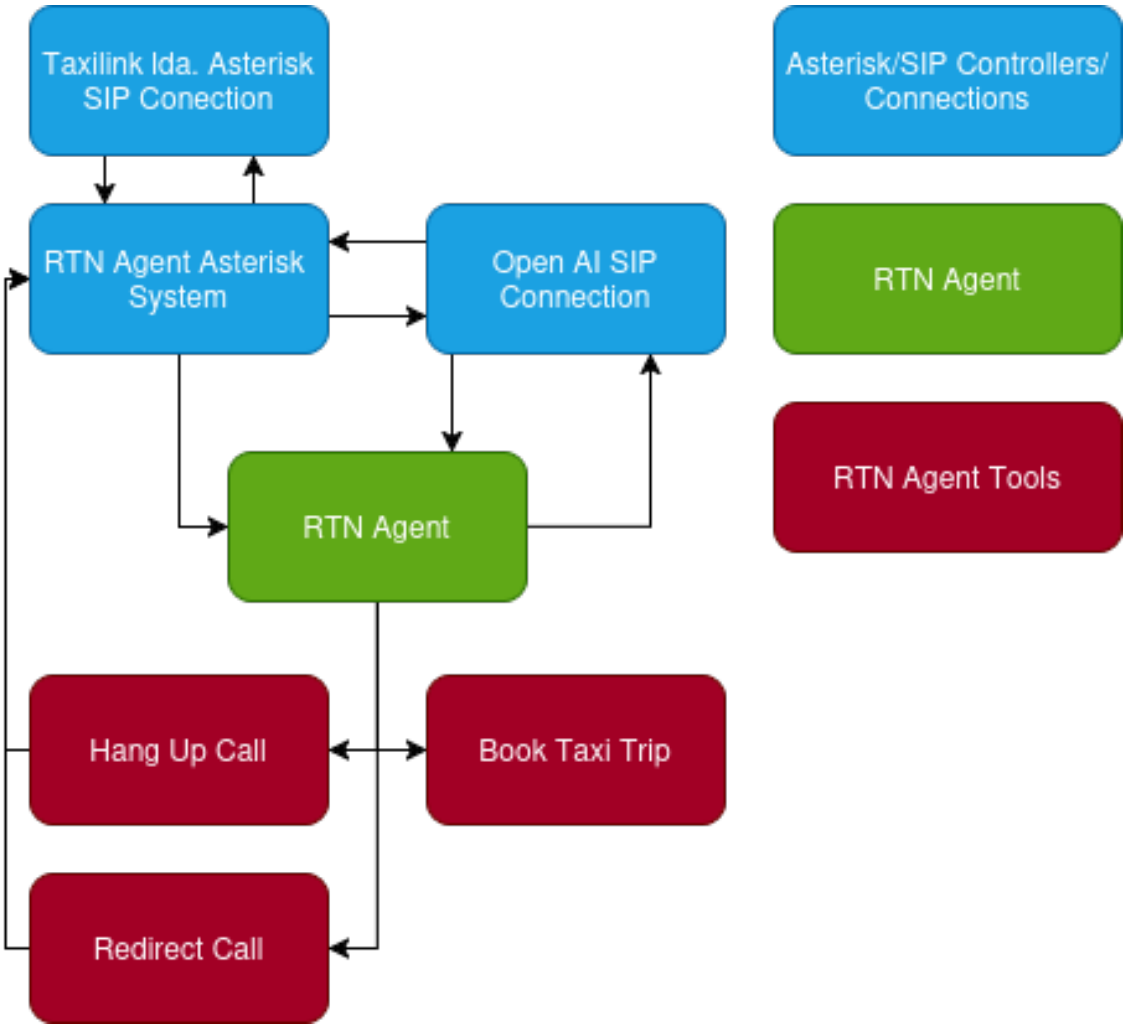


Figure 7.2: RTN Agent Flow

The RTN agent was developed over the Asterisk SIP architecture and used it as a bridge between the internal Geolink services, the RTN agent Asterisk system and OpenAI's Realtime GPT SIP connection. The Agent is composed of two different components that work with each other, the Asterisk system that controls the SIP connections and the call and the server where the model is housed.

7.2.2.1 Asterisk System

The agent Asterisk system works by receiving an open SIP connection from the Geolink lda. services that forward the user's call into the agent. When the Asterisk system receives the SIP connection it then sends an API request to the model server, that request initiates the agent call instance and saves the important information for the call. If the initialization process is successful and no connection error occurs the model server responds with a "go ahead" signal that gives the green light to the agent Asterisk system to pass the SIP connection to the OpenAI SIP services and send a request to the server with the connection information.

The model server after the connection to the OpenAI SIP services is made, makes API calls to the OpenAI services and receives information in specific webhooks to functions as a control center of the call, initializing the model instance, answering the call that is received through the SIP connection and executing the tool calls that come from the model.

After the model portion of the agent ends its interactions with the user the SIP connection with the OpenAI services is closed, and the endpoint is returned to the Asterisk controller where, depending on the outcome of the interaction with the user, the Asterisk system performs post call cleanups and ends the call or sends the SIP connection back to Geolink lda. internal Asterisk system to be redirected to a human operator.

7.2.2.2 Tools and functionalities

As an agent based around a less powerful model, built with continuous conversation and ultra low latency in mind, this agent had to be streamlined into a single main function, that being the booking of taxi trips. The agent can use three tools with one being its main function of booking a taxi trip and the other two being call controlling tools.

- Book Taxi Trip: This is the main tool that enables the agent's functionality, this tool is called when the agent has collected the pickup location, the time and date of the trip and preferences for the taxi. To overcome the ambiguity problem when a location is specified the agent had to be restricted in its scope, it will be made aware of the five most common pickup locations of the user, the possible pickup locations will be limited to only these locations and the in the case of the user requiring a different location than the ones in that list the agent would then redirect the call to a human operator that would sort out that specific trip instance. The tool receives the pickup location information as a pointer to a location saved in the Geolink lda. servers that is then used to book the trip, the time and date of pickup is received in ISO 8601 format with the possibility of being a null value when the trip is to be immediate and the preferences are received as boolean value that are then translated to flags for the service. The tool then returns a confirmation message to the agent containing the status of the tool execution.

- Hang Up Call: This is the one of the call control tools that the agent has access to, it gives the agent the ability to end its own call at the end of an interaction. It functions by making a

hang-up call to the OpenAI's Realtime API services containing the call id and identification tokens that tell the service to close its SIP connection, at the same time the status of the registry of the call is changed to reflect the reason the call was ended, this same status is then used in the agents Asterisk system to perform the specific routing to end the call. This tool is called by the agent when there hasn't been any errors or problems in the call and when is main purpose, booking a taxi trip, has been fulfilled to end the call.

- Redirect Call: As the second call control tool and much like the Hang Up Call tool, it also functions by sending an API request to OpenAI's Realtime API services to end the SIP connection, but it differs in the aspect that it will change the status of the call registry to specify that the call need to be redirected, this makes it so that when the SIP connection endpoint returns to the agent's Asterisk controller the call isn't ended but redirect back to the Geolink lda. internal Asterisk system to be redirected to a human operator. This tool is called whenever there is a problem that has to be resolved by a human agent or when specifically requested by the user.

7.2.2.3 Prompting

As with the WAC agent, the tools and model are only part of what defines the final behavior of the RTN agent. In a voice based environment the initial prompt becomes even more important, as it has to control not only the logical flow of the interaction but also the rhythm, brevity and safety of the spoken responses. Because the agent operates in real time, the prompt was designed to minimize hesitation, reduce unnecessary turns and enforce an operational scope that fits taxi booking.

The RTN agent prompt was written as a procedural instruction set that constrains the model to a narrow and highly predictable behavior. Instead of encouraging open ended conversation, the prompt focuses on short spoken turns, explicit confirmation steps and clear transfer conditions. This structure was necessary due to the lower reasoning reliability of realtime audio models when compared with text based models, and also due to the higher operational risk of errors in a live phone call. In practice the prompt can be divided into seven major groups of instructions:

- Identity and Interaction Rules: section controls how the model sees itself, its behavior and the language instructions it should follow. The model is a Geolink lda. agent with the purpose of booking trips for the user. It prompts the model to act with a friendly and professional tone, using short phrases, natural pauses and making it not interrupt the user. As for language it should speak in European Portuguese in all interactions and always remain in that language. This portion of the initial prompt also governs some minor interaction rules the agent must follow, like limiting the agent to one idea per phrase as to keep the interactions easier to understand and follow, only responding to the main user, remain silent when waiting for tool result and time out behavior.

- Address, Time and Date Handling: This section governs how the model handles and verifies both the address and the time provided. The model can only accept as a pickup location one of the five locations it was given and must be sure that the pickup location provided by the user corresponds to one of them or is phonetically close to it, if unsure or if the user asks for a place that isn't in the list it should inform the user and then redirect the call to a human operator. As for the time it should recognize turns of phrase specific to the Portuguese language and be able to interpret them correctly, this section also provides some examples and meanings of such turns of phrase.

- Conversation Flow: This give the model a flow script that it should follow with a specific question order to collect the taxi trip booking. It should start by greeting the user then ask, for the time and day of the trip, the pickup location, taxi preferences and then a confirmation of the information. After collecting and confirming the information it then should call the Book Taxi Trip tool to make the booking and depending on the response of the tool it then should end the call.

- Tool Usage Rules: This contains an explanation on how the model should use the Tools it has access to, the names of the tools, what information to send as input and their functionalities.

- Error and Transfer Logic: This portion Contains a list of rules for error handling mid-flow, as a way to re-collect information as well as possible logic and tool errors that would require the model to redirect a call to a human operator.

- Success and Closing Responses: This section prompts the model in how it should handle end of call interactions such as goodbyes and end of flow interactions and the Hang Up Call usage cases.

- Reference Table and Examples: Finally this last section contains some ground truth references and some examples of possible interactions that might occur.

7.2.3 Integration with existing systems

The RTN agent interacts with the existing Geolink lda. systems in two different ways. In the first the agent's Asterisk controller is able to receive and redirect SIP connections with the overall Asterisk system, this permits calls to be directly directed to the RTN agent's flow with a simple conditional clause for the test case users and also permits call to be redirected back to the overall system for different handling. The second way is through Geolink lda. internal service API that permits the tools allocated to the agent to be used to book taxi trips.

Chapter 8

Results

In the same way that each of the four developed agents have different approaches, specifications and objectives the testing of each of the agents was different as to best detect its real world value. These Tests were performed in accordance with the availability of Geolink lda's services and internal decision, this shaped the test format of all the agents and was done in a way to minimize the impact on normal services of the company. The real world impact of these tests also shaped their duration and format.

8.1 Text Agents

Both of the text agents were tested in a WhatsApp environment interacting with a subgroup of the total WhatsApp traffic with the objective of analyzing results, sentiment and adoption.

8.1.1 WhatsApp Helper Agent

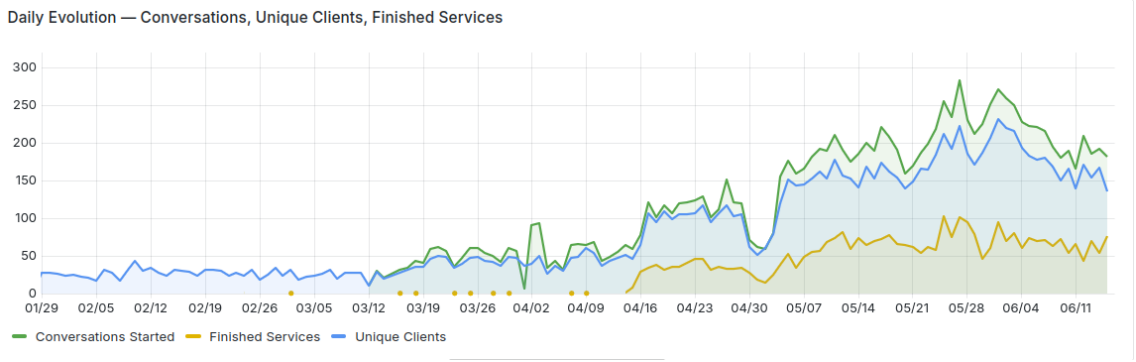


Figure 8.1: Daily evolution of conversations, unique clients, and finished services in the WhatsApp service

The WAH Agent was designed with the intention of replacing the previous WhatsApp service, this design decision shaped the test of the agent to a long term analysis of the agent's performance.

The structured nature of the WAH agent also left open more direct user sentiment gathering systems instead of manual review.

The recorded test took place in a timeline of three months with two months of comparable data of the previous system this test was rolled out to all users and continues to be the default WhatsApp system in use.

In this evaluation of the system the main metrics that most reflected the real world impact of the WAH agent in the service were, the service growth trend, the concluded service number when compared to the total number of conversations, service requests and service attributions, conversion rate and average sentiment rating.

The best indication for the efficacy of the WAH agent since the beginning of the test is the steady growth that occurred in the WhatsApp service. As seen in Figure 8.1, which plots the evolution of the conversation count each day, the new users that have used the system that day and the number of services done by the agent. The graph shows a consistent income of new messages and users before agent was enabled, with a slow rise of new users and sporadic services.

At this point it's important to point out that the discrepancy between the number of new daily messages and the number of services come from the fact that many of the messages received in the WhatsApp service don't reach a booked service process, this has many reasons, stemming from no longer needing the service or finding alternatives to being a message that doesn't have intention of making a taxi booking, such as when asking questions or spam messages, when that is the case there is always a human intervention step.

The test started with a limited testing run mid-March where the agent was tested in a small scale. In this initial release the number of services made by the WAH agent at this time is very small and was used as a consistency test for the future release. By mid-April in conjunction with a publicity drive the WAH agent was distributed to the total of the WhatsApp services, this is characterized by a sharp increase in all three metrics, and it is possible to see that accelerated growth in the use of the WAH agent roughly reflects the growth in new users.

From this its also possible to see that as time in the test has passed the percentage of returning users, or the user conversion of the agent also increases, this is reflected in the gap in new users and new conversations widening.

While the initial growth can be largely derived from the publicity drive the subsequent acceleration in the growth of the service can be attributed to the efficacy of the WAH agent. After each service the user is prompted to answer a simple five-star satisfaction survey to garner their opinion on the service, with an average of 4.79 score out of five it is also possible to see that the service reception is high.

The service also was able to reach an effective service completion rate, that being the rate of finished services of all initiated services, of 65% which is comparable to the human operator rate in the pre deployment service.

Overall the WAH agent use reflects a lower workload to the human operators that constitute Geolink lda's WhatsApp services as the most menial and occurring portion of the service is able to be automated, leaving operators available to handle non booking related services and edge cases.

8.1.2 WhatsApp Conversational Agent

The test for the WAC agent was planned to take place throughout a week's time being used for a subgroup corresponding to a fleet, this test was supposed to garner the performance of the agent, how it behaved in real life cases and user satisfaction of the agent. This test had to be ended early because of extreme errors with hallucinations and tool use. Even after extensive iteration in prompting and tool clarification the model consistently hallucinated tool use, misinputted tool data and entered in message loops with the user. Even when instructed to ask for operator assistance when having problems it wouldn't ask for assistance and continue in loops.

After the test was stopped and analysis of the agent was made it was clear that the model that was used was unable to handle the scope of the agent and tools. Only two solutions to this problem are possible, the reduction of the scope of the agent, which would make it as inflexible as the WAH agent with a higher cost, or using a more powerful model which would increase the cost to unacceptable levels and increase the latency making the agent slow and unwieldy. After some consideration it was decided that the gains of the agent, with the models currently available at the current price ranges, it wasn't feasible to employ an agent with these functionalities and more focus was made on improving the WAH agent. The architecture and surrounding structures remain valid and usable in the case of a future model becomes available to make the base of the agent.

8.2 Voice Agents

Both of the voice agents were tested in an Asterisk environment interacting with a subgroup of the total call traffic with the objective of analyzing results, sentiment and adoption.

8.2.1 Napoleão system

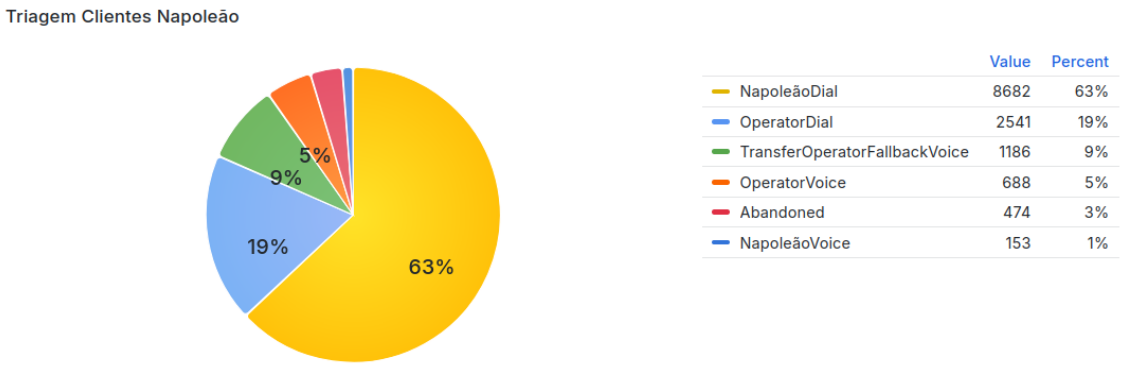


Figure 8.2: Napoleão system interaction outcomes during the voice prompt rollout

As an expansion of the existing system the rollout of the voice detection was implemented in a supporting way to the existing system which allowed it to be used by all the current Napoleão

system users, as stated previously in the Napoleão Call Manager section, this system was designed for a specific subgroup of users that have a very consistent location as the pickup location as a way to fast track the process by giving the option to ask for a taxi to a specific location at the press of a button, with the expansion allowing the use of voice prompts to make this selection.

As an audio based system data gathering in the intracall process is difficult, making it necessary for the review process to be done by hand. The test consisted in a month-long test that was applied to all the Napoleão system user calls. After the test period due to the small nature of the service change it was kept in service.

In this test we can see in Figure 8.2 that most of the users still use the dial system, this is expected as most of the users are previous users of the service and are pre disposed to use the dial buttons, the structure of the voice prompt also leads people who want a faster process to press the dial buttons. When considering all the possible interactions, the voice process accounts for 14% of the call traffic.

These 14% are distributed between 1% voice activated taxi 5% voice activated operator and 9% operator fallback. The low percentage of voice activated taxi trips is attributed to the previous causes where most of the call traffic are familiar with the system and don't reach the end of the voice prompt, same with the voice activated operator calls. In both cases manual review shows that of the voice activation users are older people, this was one of the target demographics with this change to help with usage of the system. The 9% of fallback consist of 6% user error/ purposeful silence, where users fail to say the correct words or say them before the signaling sound or users that leave the call on without any input. The 3% left consist on interpretation errors where the model is unable to attribute between the two options even when one of them is said, this can be exacerbated by accents, noisy environments or background voices.

Overall, while not having a massive impact in the totality of the service this test shows that there is a specific demographic that is being helped by this expanded service, and combined with its low cost of operation and its nature as an expansion of the service instead of a replacement made the decision to continue the use of the service after the test an easy one.

8.2.2 Realtime Napoleão Agent

The RTN agent was tested as an alternative to the normal call system, with the objective of offloading the less intensive interactions of booking and asking for taxis, the test recurred throughout a week's time in a single queue corresponding to a specific fleet, the nature of the full audio data made it difficult to analyze the intracall data and made necessary the manual review of the audio files of the calls. Throughout the week that the test took place the RTN agent received and managed over one hundred calls with a service rate of 53%, a redirect call rate of 21% and a user hangup rate of 26%.

When analyzing these three different metrics more closely it can be noted that most of the successful calls follow the prompted script, with few corrections being needed by the user to reach the final pickup location in the remaining cases. This can be attributed to the agents very reduced function set, limited possible pickup locations and low tool number which when compared to the

WAC agent shows an extremely low percentage of hallucination. While low the hallucination problems weren't non-existent as in one extremely peculiar case the model hallucinated all tool calls that it could make without making them and making up the tool call results to the user, saying that the taxi trip was booked and trying to end the call, at which point it remained silent until the user hung up themselves.

Most of the hallucinations that occurred consisted in the agent imagining places that weren't in the allowed list of pickup locations or the call of tools using nonsensical values, these cases provoked errors that were catchable by the fall back system and were redirected to a human operator, with, as stated previously, a minimal number of cases where the agent hallucinated tool calls.

Other than the hallucinations one of the biggest setback of the agent comes from the model itself, while it is able to speak in European Portuguese it has the tendency to randomly start speaking in Brazilian Portuguese mid-conversation, this bug persisted even after many prompting iterations and seems to stem from the abundance of Brazilian Portuguese training data compared to European Portuguese used in the model's training.

Of the cases where the call was redirected to a human operator the majority consisted in users requesting places outside the list given to the agent following the expected error flow and fallback. In the user hangup cases most occurred immediately after the first interaction with the agent. The agent while conversationally fluent is still recognizable as an automated AI process, this still turns away a non-insignificant number of users who aren't willing or comfortable in using this type of services.

While no direct user rating system could be accurately implemented due to hallucination problems, in manual analysis the sentiment seemed to be mostly neutral, with some overly positive reactions, with some users even experiencing excitement over the agent's capability to carry out a conversation, and some negative with users becoming aggravated when the agent didn't fully understand what was being said and having to redirect the call to a human operator. When accounting for all the cases the sentiment in this test seemed to be slightly positive.

8.3 Future Improvements

For most of the agents their improvement can be boiled down to improvement in their base models. In the case of the voice agents, The Napoleão system can be improved when new, more powerful, efficient and compact models become available to run locally as well as employing an audio streaming approach with a compatible local model to make it possible for the user to use the voice recognition at any point of the call. As for the RTN agent, the new audio-to-audio realtime models have been rapidly improving, since the test was performed two new realtime models were released by OpenAI, realtime GPT 1.5 and 2.0, these could allow new future versions of the agent to expand its functionalities and perform with fewer errors.

In the text based agents, the WAH agent is planned to increase its functionalities to include direct calls between the user and the taxi drivers, increase the available hardware to use better local models already in circulation as well as increasing the scope of inputs linking it to more

WhatsApp numbers to increase the reach of the tool. Finally, the WAC agent requires new more powerful agents that keep the cost and latency low enough to handle the total of its functionalities without such an error rate.

Chapter 9

Conclusions

This thesis set out to determine which agent architectures and deployment strategies are operationally viable for LLM-based taxi request automation across voice and text channels. Four agents were implemented and tested in live dispatch traffic at Geolink lda.: the WhatsApp Helper Agent (WAH), the WhatsApp Conversational Agent (WAC), the Napoleão voice expansion, and the Real-time Napoleão Agent (RTN). The agents spanned two fundamental design poles: structured multi-model pipelines that use LLM reasoning only for classification and routing (WAH, Napoleão), and fully conversational end-to-end agents where a single model orchestrates all interaction and tool use (WAC, RTN). The central finding is that architectural scope, more than raw model capability, determined production viability. The three agents built around a constrained scope, tool-grounded commitments and explicit escalation paths reached sustainable operation, while the broader-scope conversational agent could not.

9.1 Architectural viability and the reliability-flexibility trade-off

Addressing RQ1 directly, the results show that structured and scope-limited designs are deployable in production with acceptable failure rates, while fully conversational designs are not, given the models available at viable cost and latency at the time of deployment. The WAH agent, the Napoleão voice expansion and the RTN agent all reached a state where they could be left running with human operators only handling edge cases and escalations. The WAC agent did not reach this state.

The WAC case makes the trade-off concrete. The agent was equipped with thirteen tools spanning five functional groups (booking, coverage queries, communication and handoff, conversation control, and status tracking), alongside a detailed multi-section prompt. This breadth exceeded what the underlying GPT 4.1-mini model could reliably orchestrate in live traffic: the result was hallucinated and misused tool calls, message loops and an inability to self-escalate even when explicitly instructed to do so. The only viable fixes identified were reducing the scope of the agent, which collapses it back into a design comparable to the WAH agent, or using a more powerful model, which raises cost and latency to unacceptable levels for a real-time dispatch service. This

is not a failure of the architecture itself but a reflection of what current models can reliably sustain under a broad instruction set in live traffic. By contrast, the RTN agent, limited to three tools and a tightly scoped prompt, showed a markedly lower hallucination rate under the same class of reasoning model.

The lesson from this is that, with the models currently available at acceptable cost and latency, narrowing scope combined with tool-grounded commitments and explicit escalation outperforms full conversational autonomy in a production setting.

9.2 Failure modes and the role of human-in-the-loop

Across all agents the recurring failure categories that emerged in live traffic were hallucinated or incorrectly parameterised tool calls, out-of-list location requests, ASR and interpretation errors in noisy telephony conditions and, specific to the RTN agent, a persistent tendency for the underlying model to drift from European Portuguese into Brazilian Portuguese mid-conversation despite extensive prompting iterations.

The agents that remained in service handled these failures through escalation and fallback. The Napoleão voice expansion redirected to a human operator when confidence in the transcription fell below threshold. The RTN agent redirected when a location could not be resolved to the allowed set or when tool execution failed. The WAH agent escalated through its structured flow whenever a request fell outside the defined scope. In all three cases the human operator was the recovery mechanism that absorbed what the agent could not handle. Addressing RQ2, the conclusion is that escalation and fallback were not contingency measures but a first-class design layer without which none of the agents would have been safe to deploy. The defining failure of the WAC agent was precisely that this layer did not function reliably.

9.3 Operational impact and deployment constraints

Addressing RQ3 and RQ4 together, the quantified outcomes across agents reflect how the latency, cost and capability triangle shaped each deployment decision. The WAH agent reached a service completion rate of 65%, comparable to the human operator rate prior to deployment, and an average user satisfaction score of 4.79 out of 5, with consistent growth in use following the full rollout. The Napoleão voice expansion accounted for 14% of voice prompt interactions, with manual review confirming that the users engaging with the voice option were predominantly from an older demographic that the service had previously been unable to adequately serve, all at a low operational cost. The RTN agent reached a 53% service rate in its test week, with a 21% redirect rate and a 26% user hangup rate, the latter largely attributable to users unwilling to interact with an automated voice system.

These outcomes reflect the different constraints of the two channels. Voice requires low latency and leaves very little room for reasoning errors in real time, which made a locally deployed Whisper model the right choice for the Napoleão expansion, being fast, offline, and sufficient for

binary intent classification, and OpenAI's Realtime GPT the right choice for the RTN agent, accepting its lower reasoning ceiling in exchange for the audio-to-audio latency needed for natural conversation. Text, by contrast, tolerates slightly more latency and allows the richer dual-model architecture of the WAH agent, where a locally deployed Qwen3:30B handles fast intent classification and a locally deployed Gemma-4-31B-Heretic handles the more complex reasoning tasks, keeping the entire pipeline off cloud infrastructure and reducing operational cost. The WAC agent, requiring broader reasoning capability, relied on a cloud-hosted GPT 4.1-mini, which provided the necessary context window and tool-calling support but introduced ongoing token-based cost and a dependency on external infrastructure.

9.4 Limitations and outlook

Several constraints limit the generalisability of these results. All deployments took place within a single operator environment, Geolink lda., with a specific user base and a specific fleet management system. The audio-based nature of the voice agents made automated intracall data collection impractical, requiring manual review, which constrained the scale of analysis. Test windows for the RTN agent and WAC agent were short, in part because the WAC test was ended early due to the severity of its failure modes.

The model landscape has also continued to evolve rapidly since the tests were conducted. Two new versions of the OpenAI Realtime model, GPT 1.5 and 2.0, were released after the RTN agent was tested. This reinforces rather than undermines the conclusions: the RTN and WAC agent architectures remain valid scaffolding for stronger and cheaper future models. As reasoning capability increases and cost decreases, the scope of what constrained agents can safely handle will expand, and the WAC design in particular represents a ready framework for that transition, as noted in the Future Improvements section.

Overall, the work demonstrates that meaningful automation of taxi request handling is achievable today with structured, scope-limited agents and explicit human-in-the-loop fallback, and that the path to broader conversational automation runs through better models rather than through relaxing the architectural constraints that made the viable agents safe.

References

[1] Associated Press. *McDonald's ends test run of AI-powered drive-thrus with IBM*. Accessed 2026-02-02. June 2024. URL: https://apnews.com/article/mcdonalds-ai-drive-thru-ibm-bebc898363f2d550e1a0cd3c682fa234.

[2] Restaurant Business Online. *McDonald's is ending its drive-thru AI test*. Accessed 2026-02-02. June 2024. URL: https://restaurantbusinessonline.com/technology/mcdonalds-ending-its-drive-thru-ai-test.

[3] Wendy's. *Leading Drive-Thru Innovation with Wendy's FreshAI*. Accessed 2026-02-02. Dec. 2023. URL: https://www.wendys.com/blog/drive-thru-innovation-wendys-freshai.

[4] Google Cloud. *How Wendy's new AI-powered drive-thru is speeding service*. Accessed 2026-02-02. Sept. 2023. URL: https://cloud.google.com/transform/wendys-generative-ai-drive-thru-reinvention-worker-freedom.

[5] Fortune. *Inside Wendy's drive-thru AI that makes ordering fast food faster*. Accessed 2026-02-02. Oct. 2024. URL: https://fortune.com/2024/10/15/wendy-google-ai-drive-thru-expansion/.

[6] Yum! Brands. *Yum! Brands to accelerate AI innovation in an industry-first collaboration with NVIDIA*. Accessed 2026-02-02. Mar. 2025. URL: https://www.yum.com/wps/portal/yumbrands/Yumbrands/news/press-releases/yum+brands+to+accelerate+ai+innovation+in+an+industry-first+collaboration+with+nvidia.

[7] NVIDIA Blog. *AI on the Menu: Yum! Brands and NVIDIA Partner to Advance AI at Restaurants*. Accessed 2026-02-02. Mar. 2025. URL: https://blogs.nvidia.com/blog/yum-brands-ai/.

[8] Restaurant Dive. *Yum Brands, Nvidia will deploy new AI at 500 restaurants*. Accessed 2026-02-02. Mar. 2025. URL: https://www.restaurantdive.com/news/yum-brands-nvidia-ai-taco-bell-pizza-hut-kfc-deal/742926/.

[9] SoundHound AI. *White Castle Case Study (PDF)*. Accessed 2026-02-02. Feb. 2024. URL: https://www.soundhound.com/wp-content/uploads/2024/02/White_Castle_Case_Study.pdf.

[10] Business Insider. *White Castle is taking drive-thru orders using an AI voice system*. Accessed 2026-02-02. June 2024. URL: https://www.businessinsider.com/white-castle-drive-thru-ai-says-more-accurate-than-humans-2024-6.

[11] Intel. *Americana Restaurants/KFC & Sodaclick Case Study (PDF)*. Accessed 2026-02-02. July 2025. URL: https://www.intel.com/content/dam/www/central-libraries/us/en/documents/2025-07/kfc-americana-case-study.pdf.

[12] Hongshen Chen et al. "A Survey on Dialogue Systems: Recent Advances and New Frontiers". In: *ACM SIGKDD Explorations Newsletter* 19.2 (2017), pp. 25–35. DOI: 10.1145/3166054.3166058.

[13] Steve Young et al. "POMDP-Based Statistical Spoken Dialogue Systems: A Review". In: *Proceedings of the IEEE* 101.5 (2013), pp. 1160–1179. DOI: 10.1109/JPROC.2012.2225812.

[14] Timo Schick et al. "Toolformer: Language Models Can Teach Themselves to Use Tools". In: *Advances in Neural Information Processing Systems*. Vol. 36. Curran Associates, Inc., 2023. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

[15] Shunyu Yao et al. "ReAct: Synergizing Reasoning and Acting in Language Models". In: *International Conference on Learning Representations (ICLR)*. arXiv:2210.03629. 2023. URL: https://arxiv.org/abs/2210.03629.

[16] Shamik Roy et al. "FLAP: Flow-Adhering Planning with Constrained Decoding in LLMs". In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Association for Computational Linguistics, 2024, pp. 517–539. DOI: 10.18653/v1/2024.naacl-long.29. URL: https://aclanthology.org/2024.naacl-long.29/.

[17] Lei Huang et al. "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions". In: *ACM Transactions on Information Systems* (2024). DOI: 10.1145/3703155. URL: https://arxiv.org/abs/2311.05232.

[18] Jan Deriu et al. "Survey on evaluation methods for dialogue systems". In: *Artificial Intelligence Review* 54.1 (2021), pp. 755–810. DOI: 10.1007/s10462-020-09866-x.

[19] Sarah E. Finch and Jinho D. Choi. "Towards Unified Dialogue System Evaluation: A Comprehensive Analysis of Current Evaluation Protocols". In: *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, 2020, pp. 236–245. DOI: 10.18653/v1/2020.sigdial-1.29. URL: https://aclanthology.org/2020.sigdial-1.29/.

[20] Herbert H. Clark and Susan E. Brennan. "Grounding in Communication". In: *Perspectives on Socially Shared Cognition*. Ed. by Lauren B. Resnick, John M. Levine, and Stephanie D. Teasley. Washington, DC, USA: American Psychological Association, 1991, pp. 127–149. DOI: 10.1037/10096-006.

[21] Sebastian Möller. *Quality of Telephone-Based Spoken Dialogue Systems*. New York, NY, USA: Springer, 2005. ISBN: 978-0-387-23190-7. DOI: 10.1007/b100796.

[22] Antoine Raux and Maxine Eskenazi. "Optimizing Endpointing Thresholds using Dialogue Features in a Spoken Dialogue System". In: *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*. Ed. by David Schlangen and Beth Ann Hockey. Columbus, Ohio: Association for Computational Linguistics, June 2008, pp. 1–10. URL: https://aclanthology.org/W08-0101/.

[23] Antoine Raux and Maxine Eskenazi. "A Finite-State Turn-Taking Model for Spoken Dialog Systems". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Ed. by Mari Ostendorf et al. Boulder, Colorado: Association for Computational Linguistics, June 2009, pp. 629–637. URL: https://aclanthology.org/N09-1071/.

[24] Tiancheng Zhao, Alan W Black, and Maxine Eskenazi. "An Incremental Turn-Taking Model with Active System Barge-in for Spoken Dialog Systems". In: *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Ed. by Alexander Koller et al. Prague, Czech Republic: Association for Computational Linguistics, Sept. 2015, pp. 42–50. DOI: 10.18653/v1/W15-4606. URL: https://aclanthology.org/W15-4606/.

[25] Erik Ekstedt and Gabriel Skantze. "TurnGPT: a Transformer-based Language Model for Predicting Turn-taking in Spoken Dialog". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Ed. by Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 2981–2990. DOI: 10.18653/v1/2020.findings-emnlp.268. URL: https://aclanthology.org/2020.findings-emnlp.268/.

[26] Erik Ekstedt and Gabriel Skantze. "Voice Activity Projection: Self-supervised Learning of Turn-taking Events". In: *Interspeech 2022*. 2022, pp. 5190–5194. DOI: 10.21437/Interspeech.2022-10955.

[27] Galo Castillo-López, Gael de Chalendar, and Nasredine Semmar. "A Survey of Recent Advances on Turn-taking Modeling in Spoken Dialogue Systems". In: *Proceedings of the 15th International Workshop on Spoken Dialogue Systems Technology*. Ed. by Maria Ines Torres et al. Bilbao, Spain: Association for Computational Linguistics, May 2025, pp. 254–271. ISBN: 979-8-89176-248-0. URL: https://aclanthology.org/2025.iwsds-1.27/.

[28] Xinrong Zhang et al. "Beyond the Turn-Based Game: Enabling Real-Time Conversations with Duplex Models". In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 11543–11557. DOI: 10.18653/v1/2024.emnlp-main.644. URL: https://aclanthology.org/2024.emnlp-main.644/.

[29] Shengpeng Ji et al. "WavChat: A Survey of Spoken Dialogue Models". In: (2024). DOI: 10.48550/arXiv.2411.13577. arXiv: 2411.13577 [eess.AS]. URL: https://arxiv.org/abs/2411.13577.

[30] Essam Alghamdi, Martin Halvey, and Emma Nicol. "System and User Strategies to Repair Conversational Breakdowns of Spoken Dialogue Systems: A Scoping Review". In: *ACM Conversational User Interfaces 2024 (CUI '24)*. Association for Computing Machinery, July 2024, pp. 1–13. DOI: 10.1145/3640794.3665558. URL: https://doi.org/10.1145/3640794.3665558.

[31] Dan Bohus and Alexander I. Rudnicky. "A principled approach for rejection threshold optimization in spoken dialog systems". In: *Interspeech 2005*. 2005, pp. 2781–2784. DOI: 10.21437/Interspeech.2005-820.

[32] Dan Bohus and Alexander I. Rudnicky. "Sorry and I Didn't Catch That! - An Investigation of Non-understanding Errors and Recovery Strategies". In: *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*. Ed. by Laila Dybkjær and Wolfgang Minker. Lisbon, Portugal: Special Interest Group on Discourse and Dialogue (SIGdial), Sept. 2005, pp. 128–143. DOI: 10.18653/v1/2005.sigdial-1.14. URL: https://aclanthology.org/2005.sigdial-1.14/.

[33] Dan Bohus et al. "Online Supervised Learning of Non-Understanding Recovery Policies". In: *2006 IEEE Workshop on Spoken Language Technology (SLT 2006)*. IEEE, 2006, pp. 75–80. DOI: 10.1109/SLT.2006.326844. URL: https://doi.org/10.1109/SLT.2006.326844.

[34] Marilyn A. Walker et al. "PARADISE: A Framework for Evaluating Spoken Dialogue Agents". In: *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*. Madrid, Spain: Association for Computational Linguistics, July 1997, pp. 271–280. DOI: 10.3115/976909.979652. URL: https://aclanthology.org/P97-1035/.

[35] Alec Radford et al. "Robust Speech Recognition via Large-Scale Weak Supervision". In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 28492–28518. URL: https://proceedings.mlr.press/v202/radford23a.html.

[36] Dan Oneaţă et al. "An Evaluation of Word-Level Confidence Estimation for End-to-End Automatic Speech Recognition". In: *2021 IEEE Spoken Language Technology Workshop (SLT)*. 2021, pp. 258–265. DOI: 10.1109/SLT48900.2021.9383570.

[37] David Qiu et al. "Multi-Task Learning for End-to-End ASR Word and Utterance Confidence with Deletion Prediction". In: *Interspeech 2021*. 2021, pp. 4074–4078. DOI: 10.21437/Interspeech.2021-1207.

[38] Qiujia Li et al. "Residual Energy-Based Models for End-to-End Speech Recognition". In: *Interspeech 2021*. 2021, pp. 4069–4073. DOI: 10.21437/Interspeech.2021-690.

[39] Henry Weld et al. "A Survey of Joint Intent Detection and Slot Filling Models in Natural Language Understanding". In: *arXiv preprint arXiv:2101.08091* (2021). URL: https://arxiv.org/abs/2101.08091.

[40] Qian Chen, Zhu Zhuo, and Wen Wang. "BERT for Joint Intent Classification and Slot Filling". In: *arXiv preprint arXiv:1902.10909* (2019). URL: https://arxiv.org/abs/1902.10909.

[41] Tobias Bunk et al. "DIET: Lightweight Language Understanding for Dialogue". In: *arXiv preprint arXiv:2004.09936* (2020). URL: https://arxiv.org/abs/2004.09936.

[42] Ibrahim Abdelaziz et al. "Granite-Function Calling Model: Introducing Function Calling Abilities via Multi-task Learning of Granular Tasks". In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*. arXiv:2407.00121. 2024. URL: https://aclanthology.org/2024.emnlp-industry.85/.

[43] Darren Yow-Bang Wang et al. "SLOT: Structuring the Output of Large Language Models". In: *arXiv preprint arXiv:2505.04016* (2025). URL: https://arxiv.org/abs/2505.04016.

[44] Luis Moreira-Matias, Michel Ferreira, João Mendes-Moreira, et al. *Taxi Service Trajectory - Prediction Challenge, ECML PKDD 2015*. UCI Machine Learning Repository (Dataset). Public dataset mirrored by UCI; original challenge page referenced therein. 2015. DOI: 10.24432/C55W25. URL: https://archive.ics.uci.edu/dataset/339/taxi+service+trajectory+prediction+challenge+ecml+pkdd+2015.

[45] Wuwei Lan, Yanyan Xu, and Bin Zhao. "Travel Time Estimation without Road Networks: An Urban Morphological Layout Representation Approach". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*. 2019. URL: https://www.ijcai.org/proceedings/2019/0245.pdf.

[46] Rafael Marques Ribeiro. "Using Machine Learning for Robotic Answering of Taxi Request Phone Calls". Master's Thesis. Faculty of Sciences of the University of Porto (FCUP), Department of Computer Science, 2024. URL: https://sigarra.up.pt/fcup/en/pub_geral.show_file?pi_doc_id=449203.

[47] Dan Bohus and Alexander I. Rudnicky. "Sorry, I Didn't Catch That! An Investigation of Non-understanding Errors and Recovery Strategies". In: *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*. 2005. DOI: 10.18653/v1/2005.sigdial-1.14. URL: https://aclanthology.org/2005.sigdial-1.14/.

[48] Yisha Wu et al. "Incremental Summarization for Customer Support via Progressive Note-Taking and Agent Feedback". In: *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*. Suzhou (China): Association for Computational Linguistics, Nov. 2025, pp. 2000–2015. DOI: 10.18653/v1/2025.emnlp-industry.140. URL: https://aclanthology.org/2025.emnlp-industry.140/.

[49] Debayan Banerjee et al. "A System for Human-AI collaboration for Online Customer Support". In: *AAAI 2023 Workshop on Representation Learning for Responsible Human-Centric AI (R2HCAI)*. 2023. URL: https://www.inf.uni-hamburg.de/en/inst/ab/lt/publications/2023-banerjee-r2hcai.pdf.

[50] Michael Higgins et al. "Actionable conversational quality indicators for improving task-oriented dialog systems". In: *Natural Language Engineering* 30.6 (2024), pp. 1229–1254. DOI: 10.1017/S1351324923000372. URL: https://www.cambridge.org/core/journals/natural-language-engineering/article/actionable-conversational-quality-indicators-for-improving-taskoriented-dialog-systems/97B0827A8F75093990BCF487469225BC.