Master Computer Science

LLMs at your Service: Intent-Guided Fine-Tuning of Large Language Models for Customer Support Applications

Name: Ramiro X. Kroonenburg

Student ID: s3738256

Date: 01-09-2025

Specialisation: Data Science

Daily supervisor: Andreas Paraskeva
1st supervisor: Jan N. van Rijn
2nd supervisor: Panagiotis Eustratiadis (UvA)
Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Leading large language models (LLMs) for customer service are often closed-weight and accessible only through APIs, leading to high inference costs, limited controllability, and dependence on external providers. Open-source LLMs hosted locally offer lower costs and greater control. In this work, we investigate whether an open-source LLM can be fine-tuned to generate reply suggestions for customer service employees, and whether its performance can approach a closed-weight model, using real-world customer service conversations. A key challenge is that conversations are noisy and unstructured, requiring normalization before fine-tuning. We address this with a two-stage pipeline. In the first stage, a closed-weight LLM anonymizes and cleans conversations, and an iterative LLM-based labeling algorithm assigns message intents. Using LLM-guided bandit-style prompt optimization, we improve clustering quality by 19% compared to a non-bandit variant, yielding thousands of intents with a long-tail distribution similar to our human-labeled set. In the second stage, we fine-tune an open-weight model on these pre-processed, intent-annotated conversations, comparing several training strategies: response-only fine-tuning, chain-of-thought training (predicting intents before responses), a staged approach (learning intents then responses), and a curriculum mixing these regimes. Across lexical and semantic evaluation metrics, chain-of-thought performed best among the intent-guided variants but did not surpass response-only fine-tuning. Retrieval-augmented fine-tuning outperformed retrieval only at inference, and scaling from 1.7B to 4B parameters further increased semantic similarity and lexical overlap. Using an LLM-as-a-judge for evaluation, our best open-weight models matched or exceeded a closed-weight GPT-4o baseline in semantic similarity and professionalism, though GPT-4o remained stronger in helpfulness. Intent-level analysis revealed complementary strengths: the open-weight model excelled on frequent intents, while GPT-4o was better on rare, knowledge-intensive cases.

Contents

List of Figures

List of Tables

1 Introduction

The introduction of Large Language Models (LLMs) based on the Transformer architecture [1] has revolutionized the natural language processing (NLP) domain. LLMs such as GPT-3 [2] and Llama 3 [3] have shown strong generalization across many tasks without requiring task-specific architectures. Their success in conversational AI has driven a shift from rule-based chatbots [4][5] to assistants capable of complex, multi-turn conversations [6] [7].

In customer service, LLMs promise more efficient support. Before LLMs, pipelines typically combined intent classifiers with pre-defined response templates [8]. With LLMs, intent classification and response generation can be handled within a single system, allowing for more natural, non-predefined replies and better reasoning.

However, most leading LLMs used in production today are closed-weight models. These models are only accessible through APIs and offer no transparency. This leads to practical constraints, including high inference costs from long prompts, limited controllability, and dependence on external providers. Open-weight models such as Llama-3 and Qwen3 [9] avoid these issues by providing access to model weights. Open-weight models can be deployed locally, are more cost-efficient, and give complete control over model weights, allowing domain-specific fine-tuning. Recent results demonstrate that open-weight models can match closed-weight models in performance, while offering additional advantages [10].

Our final goal is to fine-tune an open-weight model on historical, anonymized customer-service conversations so that it approaches the quality of a closed-weight system while reducing cost and regaining control over model weights. However, customer service conversations are unstructured and noisy, which could hinder the fine-tuning process. Messages within conversations can have multiple purposes or be ambiguous, making it difficult to extract the true intent behind a message.

To set up a structured and cleaned training dataset for fine-tuning an open-weight LLM, we propose to use message-level intents to normalize the conversations: each message receives a short, reusable intent label, grouping messages with the same semantic meaning, but different wording under the same intent. These intent-labeled messages will guide an open-weight LLM during the fine-tuning process. Existing generative approaches create such intent labels by calling an LLM with a prompt and letting it generate intent names [11, 12]. Building on this idea, we introduce a generative intent finding algorithm (GIFA)

tailored for high granularity (possibly thousands of intents). GIFA operates at the message level with an evolving known intent database. Given a whole conversation and a list of relevant known intents, a closed-weight LLM is prompted with rules and examples that specify when to reuse an existing intent or create a new one. For each message, the LLM returns either a known intent or a new intent, which is added to the known intent database and becomes available for subsequent conversations. Over time, the intent database grows as conversational purposes change, without manually re-labeling historical data.

The GIFA prompt largely determines the granularity of generated intents. Overly broad prompts merge distinct messages, while overly strict prompts fragment the reuse of known intents. We pair GIFA with a generative prompt finding algorithm (GPFA) that aims to optimize the prompt. GPFA runs a black-box prompt search over candidate prompts: it proposes a prompt, runs GIFA on a small subset of conversations with human-labeled intents, and scores the results with clustering-based metrics. Like other recent black-box prompt optimization frameworks such as PromptAgent [13] and strategic exploration and exploitation (SEE) [14], GPFA iteratively refines the prompt using LLM-generated feedback; to control cost, we adapt SEE with Hyperband-SEE to allocate evaluation budget early to promising candidates. The best-performing prompt found is then applied at scale to label all available conversations, yielding intents that both normalize the messages within the conversations and provide compact guidance signals during fine-tuning.

In this work, we use intents as auxiliary guidance signals during fine-tuning. Concretely, we evaluate five variants: a response-only baseline that ignores intents and learns to produce the next assistant reply based on the conversation history; a chain-of-thought (CoT) variant in which the model is trained first to predict a list of response intents (a plan) and then generate the replies; two-step variants where the model (via separate adapters) first learns to predict intents and then learns either to generate the replies directly or to produce both the intent chain and the replies in CoT format; and a mixed multi-task variant that interleaves intent prediction with CoT training in a single curriculum. During fine-tuning, we include intents as supervised targets in intent-guided variants. During evaluation, however, we assess only the quality of generated responses, not the intent chain. Thus, intents function as auxiliary supervision rather than as end goals. We also compare retrieval setups, retrieval-augmented generation (RAG) at inference versus retrieval-augmented fine-tuning (RAFT) during training, and the effects of model scaling (1.7B versus 4B).

These challenges and methods are broadly relevant across fields but become clearer when tested in the real world. We focus on a case study at Albert Heijn, the largest supermarket chain in the Netherlands. Albert Heijn handles many customer problems each week. An LLM-based chatbot handles a substantial share of these customer problems, while human employees handle the other half. The human employees are provided with re-

ply suggestions from a so-called ”co-assistant”, which uses a GPT-4o, RAG-based setup. In this setup, given the current conversation, the system retrieves relevant FAQ documents, appends them to the conversation history, and sends the combined input to a private hosted GPT-4o via Microsoft Azure's API, which forwards the request to the closed-weight model. In contrast, we call the model directly with open-weight models and retain complete control over parameters and fine-tuning. Our GPT-4o RAG baseline mirrors Albert Heijn's production design in broad strokes but is not identical; we omit proprietary components and implementation details to protect confidentiality.

While effective, the GPT-4o RAG setup introduces challenges. Each request to GPT-4o includes the conversation history and several retrieved FAQ documents, leading to high token usage and thus high API costs. Furthermore, their dependence on a closed-weight model limits their ability to improve the model through fine-tuning and makes the company dependent on other providers. Given these problems, Albert Heijn wants to explore whether past customer service conversations can be used to fine-tune an open-weight LLM, creating a domain-specific, self-hosted LLM.

We organize the work as a two-stage pipeline with five sub-stages (A-E). Stage 1 (labeling, sub-stages A-C) comprises LLM-based pre-processing of the raw conversations in sub-stage A (anonymization of personal information, cleaning through message merging and redundancy removal, and completion classification to flag whether the customer's issue appears resolved), followed by intent generation with GIFA in sub-stage B and prompt search with GPFA and Hyperband-SEE in sub-stage C. Stage 2 (modeling, sub-stages D-E) focuses on modeling: sub-stage D fine-tunes open-weight models on next-reply generation with optional intent guidance, and sub-stage E integrates retrieval (RAG/RAFT).

Our main objective is to determine whether historical customer-service conversations labeled with LLM-generated intents can be leveraged to fine-tune open-weight LLMs so that they perform competitively with closed-weight systems. If successful, this would allow Albert Heijn to replace its current LLM with an open-weight alternative while keeping the rest of its RAG system unchanged.

We study two main questions for our two-stage pipeline:

RQ-A (Stage 1: Labeling). Can we automatically generate reusable, appropriately granular message-level intents from noisy, multi-turn customer-service conversations?

Consisting of the following sub-questions:

A1 How do pre-processing choices: anonymization, cleaning, and the treatment of incomplete conversations, affect downstream label quality and fine-tuning performance? (Sub-stage A)

A2 How effectively can intent labels be discovered using the proposed GIFA/GPFA framework? (Sub-stage B)

A3 Does Hyperband-SEE improve prompt-search efficiency and final labeling quality over SEE and simple baselines? (Sub-stage C)

RQ-B (Stage 2: Modeling). Given conversations labeled with intents, how do intent signals and retrieval strategies affect the quality of an open-weight model trained to generate the next assistant reply, and how does it compare to the GPT-4o RAG baseline?

Consisting of the following sub-questions:

B1 Do intent-guided variants (CoT, two-step, mixed) outperform response-only fine-tuning? (Sub-stage D)

B2 Does RAFT outperform RAG? (Sub-stages D-E)

B3 How does model size impact the performance of these intent-guided and retrieval-augmented training strategies? (Sub-stages D-E)

B4 How do the best open-weight configurations compare with the closed-weight GPT-4o RAG co-assistant across evaluation dimensions (semantic alignment, professionalism, helpfulness)? (Sub-stages D-E)

In our work, we make the following contributions:

1. **Addresses RQ-A1:** We design an LLM-based anonymization, cleaning, and completion-classification pipeline for customer-service conversations and show empirically that filtering out incomplete conversations lowers downstream fine-tuning performance, whereas cleaning improves it.
2. **Addresses RQ-A2:** We introduce GIFA to automatically generate reusable, appropriately granular message-level intents using an evolving intent database; we analyze granularity and reuse under different retrieval/sampling settings. We couple GIFA with a GPFA that uses a small human-labeled subset and clustering-based metrics to optimize prompts for the desired granularity and reuse.
3. **Addresses RQ-A3:** We adapt SEE with Hyperband-SEE for early stopping, prioritizing promising prompts, and improving labeling quality per budget.

4. **Addresses RQ-B1:** We fine-tune open-weight models on the next-reply generation task and compare response-only against intent-guided variants (CoT, two-step, mixed). CoT is the strongest intent-guided variant but does not surpass the response-only baseline in our setup.

5. **Addresses RQ-B2, RQ-B3:** We compare RAG with RAFT and find that RAFT outperforms RAG; scaling from 1.7B to 4B further improves scores across settings.

6. **Addresses RQ-B4:** We conduct a case study at Albert Heijn and show that our best RAFT-trained open-weight models match or exceed the GPT-4o RAG co-assistant on semantic alignment and professionalism for common intents, while GPT-4o remains stronger on helpfulness and rare intents.

The remainder of this thesis is organized as follows. Chapter 2 reviews the literature on chatbots in customer service, intent discovery, prompt optimization, and fine-tuning strategies for conversational AI. Chapter 3 presents the proposed two-stage pipeline in detail, describing the pre-processing steps, GIFA, GPFA, the Hyperband-SEE extension, and the fine-tuning configurations evaluated in this work. Chapter 4 outlines the experimental setup and results for the GIFA and GPFA combination. Chapter 5 then introduces the experimental setup, results for the modeling stage, and the effect of pre-processing choices. Chapter 6 discusses the limitations of our approach and explores possible future research. Finally, Chapter 7 summarizes the main conclusions and contributions of this thesis.

2 Related Work

This section reviews prior work that informs our approach. We begin with the progression from rule-based chatbots to transformer-based LLMs and contrast open-weight and closed-weight deployments in customer service (subsection 2.1). We then define intent discovery (versus intent detection) and survey clustering-based and generative methods for producing message-level labels (subsection 2.2). Next, we examine techniques for shaping model behavior through prompt design, emphasizing black-box prompt search and resource-aware tuning of complete prompts (subsection 2.3). Finally, we summarize strategies for specializing LLMs to a domain, including supervised fine-tuning with lightweight adapters, in-context learning and retrieval of external documents at inference time, and during fine-tuning (subsection 2.4).

2.1 Evolution of Conversational AI: From Rule-Based to LLM

Early customer service chatbots were mainly built with rule-based logic and pattern matching. To the best of our knowledge, the first used chatbot was ELIZA [4], designed to imitate a psychotherapist through rules and keyword matching. Using these rules, the system could generate a fixed response or transform the user's input into a reply. After ELIZA came PARRY [5], which simulated the behavior of a paranoid schizophrenic patient, also through a set of rules and pattern matching. The performance of these systems depended on the size and quality of their rule bases. Nevertheless, such systems had limitations: the responses felt uncreative and repetitive, and the systems had problems with open-ended and ambiguous questions. [15] [16].

Over time, a distinction formed between task-oriented and chat-oriented chatbots. Task-oriented chatbots focused on solving direct problems like booking tickets, account issues, or other customer support issues [17]. On the other hand, chat-oriented chatbots focused on more general open conversations [18]. Also, the evaluation protocols differed for both. Task-oriented chatbots were judged on their task completion rate, while chat-oriented chatbots were mainly assessed based on their ability to engage users during conversations.

After the era of rule-based systems, machine learning (ML) and neural networks made a big step forward within the field. These models could learn from data instead of relying on a fixed rule base. In 2010, IBM Watson [19] exemplified this by combining natural language processing with ML-based information retrieval and showing the possibilities

of using AI in natural language processing. The system could search across knowledge bases and predict probabilities for different possible answers given a question.

The introduction of recurrent neural networks (RNNs) [20] , followed by more advanced forms like long short-term memory (LSTM) [21] and gated recurrent unit (GRU) [22], improved the modeling of sequential user queries. However, these models still processed inputs step by step, limiting their contextual range. The transformer architecture solved these constraints [1], which introduced self-attention to process entire text sequences in parallel. The self-attention mechanism enabled the model to weigh the importance of words within a sentence or document for a given task. Today's LLMs are all built on this transformer architecture. LLMs such as GPT-3 [2] and Llama 3 [3] are trained on massive amounts of text and can generalize across various tasks, such as question-answering and intent recognition [2]. These LLMs are widely used in industries such as customer service [7] and e-commerce [6], where they raise overall worker productivity and particularly help less experienced employees improve their speed and quality of work.

LLMs often fall into two broad categories: open-weight and closed-weight. Open-weight models, such as Llama-3 and Qwen3 [9], make their weights publicly available. That openness means they can be deployed locally, adapted to domain-specific needs, and give organizations more control over their data. One of the stronger arguments for using open-weight models is that they can run entirely within an organization's infrastructure, without sending data outside [10]. On the other hand, closed-weight models such as GPT-4o from OpenAI [23] do not release their weights, and are only available through external APIs. Closed-weight models offer relatively high performance across tasks, but offer no transparency and keep the user dependent on the provider's infrastructure.

Of the recent open-weight releases, Qwen3 is particularly interesting. It performs well on multilingual benchmarks and includes a thinking mode, which supports improved reasoning and multi-step problem solving [9]. Benchmarks place Qwen3 alongside models such as Llama-3, and in several cases, it is on par with the performance of closed-weight systems [24]. Furthermore, Qwen3 stands out because it can be tuned locally without having the costs or restrictions of closed-weight alternatives.

This work focuses on fine-tuning a smaller open-weight LLM, with particular attention to Qwen3. However, fine-tuning on raw customer service conversations is not straightforward: the data is noisy and unstructured. Conversations must first be cleaned and enriched with structure to make them suitable for training. We achieve this by labeling individual messages with intents and creating reusable signals to guide the model during fine-tuning. The following subsection introduces the task of intent discovery and outlines existing approaches to identify and generate such labels automatically.

2.2 Intent Discovery

Intent discovery is the task of automatically uncovering and naming previously unknown user intents from unlabeled (or partially labeled) conversational data. In contrast, intent detection assigns utterances to a fixed, predefined label set [8] [25]. This setup works well within a static domain but restricts the system to a fixed intent set. Any utterance outside the list is treated as an unknown intent. Open-world intent detection addresses this by identifying utterances outside predefined labels, particularly important in dynamic domains such as customer service, where new intents are frequently introduced. Examples of open-world intent detection methods, such as softmax thresholding [26] and one-vs-rest classifiers [27], can recognize a novel intent, but not describe it.

Building on this, intent discovery methods aim to uncover novel intents directly from unlabeled data. Unsupervised methods such as Auto-dialabel [28] try to cluster sentence embeddings from encoder models. Later, DeepAligned [29] aligned embeddings from labeled and unlabeled utterances using a BERT encoder. Deep semi-supervised contrastive clustering (DSSCC) [30] pre-trained a SentenceBERT model and applied contrastive learning to improve clustering for labeled and unlabeled data. While these approaches improve cluster quality, a limitation is that they generally work on individual utterances, ignoring the larger conversation around them, which often describes the meaning of an utterance.

Generative models have opened different routes. Intent discovery with abstractive summarization (IDAS) [12] treats intent discovery as a summarization task: first, the utterances are clustered, then an LLM produces a short label for each group. IntentGPT [11] uses few-shot prompting to label utterances and then iteratively expands its prompt with new intents discovered along the way, relying on models like GPT-4o. These generative methods skip supervised training entirely and draw on the knowledge built into the LLMs. However, they usually work at a coarse level of granularity and are not designed to work with a large, diverse set of intents.

We introduce GIFA, a semi-supervised algorithm, which generates intent labels for every message in a conversation in a single LLM pass. The key is to leverage the full conversational context and retrieve relevant prior intents for prompting.

LLMs can help group and label novel utterances, but the granularity of the intent detail depends on how the prompt is designed. The following subsection looks at frameworks for automatic prompt optimization to set up these prompts without relying on human tuning.

2.3 Prompt Optimization and Search Strategies

Prompt engineering has become essential in controlling LLMs, especially with closed-weight black-box models, where model weights are unavailable. Early work focused on training soft prompts through gradient-based tuning [31] [32]. However, these methods could not be applied to closed-source LLMs like GPT-4o, where only the inputs and outputs are available. In such cases, the optimization process is treated as a black-box optimization using discrete prompt search strategies guided by evaluation metrics based on the LLM outputs.

Gradient-free prompt optimization methods usually follow an iterative search process: a prompt is sampled, evaluated, and refined for multiple iterations. Early methods generated prompt variants via simple text edits such as deletion, substitution, or paraphrasing [33]. Recently, approaches have incorporated LLM-generated natural language feedback to guide the prompt optimization process [34]. These strategies help improve prompts, but cannot effectively explore the prompt space and combine the strengths from several already found prompts. PromptAgent solves these issues by seeing the prompt optimization process as a planning problem and uses Monte Carlo Tree Search (MCTS) to guide the exploration [35]. A search tree is grown by treating prompts as nodes and LLM-generated feedback as edges, which is then traversed.

Another direction to traversing the prompt space effectively was introduced by methods such as EvoPrompt [36] and PromptBreeder [37], which use evolutionary strategies to traverse the prompt space. These methods evolve a population of prompts via mutation and selection, but are often limited to optimizing only short and straightforward instructions rather than full prompts. Most prompt evolution strategies focus on modifying solely the instruction component, instead of jointly optimizing the instruction and the in-context examples (labeled examples included in the prompt). Because of this, the two components are treated in isolation, reducing the prompt's overall quality [14].

To address this, SEE was introduced as a joint instruction and example optimization algorithm [14]. Unlike most prior work, SEE does not assume a fixed prompt structure. It optimizes the full prompt, resulting in more effective prompts through iterative refinement. The algorithm uses feedback-driven updates, guided by LLM feedback, and uses evolutionary recombination techniques to merge the strengths from previously found prompts. Despite its strengths, SEE can be computationally expensive. Each prompt in the evolving population must be evaluated using LLM calls, which becomes costly as the prompt population grows or when the validation set is large.

We build on SEE's population-based prompt search and introduce a resource-aware mechanism based on Hyperband [38]. In this extension, we develop Hyperband-SEE, which

allocates progressive evaluation budgets and applies early stopping, allowing us to explore broadly while avoiding unnecessary computation on unpromising prompts.

We combine generative intent finding with prompt optimization to produce an optimal prompt that yields intents at the preferred granularity. This process transforms a raw conversational dataset into a labeled dataset. In this labeled dataset, the intents describe the semantic function of each message within the conversations. These labeled conversations can help guide the fine-tuning process of open-weight LLMs to generate better responses. The following subsection examines several ways to adapt an LLM to a specific domain.

2.4 Specializing LLMs: Fine-Tuning and Retrieval

Supervised fine-tuning is a regularly used method for adapting pre-trained LLMs to domain-specific tasks. The model's weights are updated using a labeled task-related dataset during fine-tuning. In the past, supervised fine-tuning has been applied to emotional support conversations, for example, within the emotional support conversation (ESC) framework, using models like BlenderBot and DiabloGPT [39]. However, full fine-tuning is often computationally expensive, requiring multi-GPU setups, and has risks of catastrophic forgetting. Parameter-efficient fine-tuning (PEFT) methods have emerged to solve these problems [40]. Low-rank adaptation (LoRA) is a PEFT approach that updates only a small set of low-rank matrices within the transformer layers, keeping the main model frozen [41]. The ESC framework [39] was later extended by incorporating synthetic conversations from ChatGPT to train a Llama variant with adapters and LoRA tuning, showing improvements over traditional fine-tuning on emotional support tasks [42]. Quantized low-rank adaptation (QLoRA) [43] extends LoRA further with 4-bit quantization and paged optimizers, enabling fine-tuning large-scale models on single GPUs. QLoRA significantly reduces the amount of VRAM used without decreasing performance.

Supervised fine-tuning can take different forms depending on the supervision format. Domain-specific supervised fine-tuning adapts a model to a particular application by training on in-domain data, such as customer-assistant conversations or QA pairs, with the target being the next reply or answer [44]. Instruction tuning is a more general variant of supervised fine-tuning, where tasks are phrased as natural-language instructions paired with responses across multiple domains. This approach used in InstructGPT, improves generalization to unseen instructions [45]. A complementary strategy is CoT supervision, which augments instruction tuning or domain-specific supervised fine-tuning by including intermediate reasoning steps before the final answer, thereby enhancing multi-step reasoning ability [46].

While fine-tuning methods update parameters, another line of research adapts LLMs with-

out changing their weights. Few-shot learning leverages a small number of task examples into the prompt, without needing parameter updates. GPT-4o [23] showed strong performance across tasks using only in-context examples. FewshotQA [47] used few-shot examples inside the prompts during the fine-tuning process and showed performance improvement on masked language modeling. While effective for large models, few-shot prompting has limited success with smaller LLMs.

RAG [48], another gradient-free method, augments LLMs by retrieving context documents at inference time. This allows models to access up-to-date information without changing their parameters, with the retrieved documents providing dynamic context that reduces hallucinations and improves factual accuracy. However, RAG performance highly depends on the retrieval setup, document coverage, and the quality of retrieved documents: noisy or irrelevant documents can harm generation [49]. Retrieval models in RAG typically rely on either lexical or semantic similarity. Traditional lexical approaches such as BM25 [50] work well for exact term matching between queries and documents, but fail when queries involve synonyms or paraphrasing. In contrast, semantic embedding models like E5 (open-weight) [51] or OpenAI's ADA-003 (closed-weight) generate dense vector representations that capture semantic similarity, but have higher computational cost.

RAFT extends RAG by adding retrieved context into the training prompts during fine-tuning. In RAFT, models are fine-tuned on the query, the context, and the answer, including relevant and irrelevant documents. This teaches the model to answer questions and select which documents in the prompt provide helpful information, given the user query. Experiments show that RAFT improves noise robustness and factuality compared to standard fine-tuning or RAG alone. Furthermore, smaller models fine-tuned with RAFT outperform larger base LLMs, making RAFT attractive for domain adaptation, since it requires less GPU power to obtain the same or better results [52].

3 Methodology

In this section, we present the complete pipeline developed in this thesis. Figure 3.1 illustrates this pipeline, providing an overview of the two main stages: labeling and modeling, and the five sub-stages: LLM-based pre-processing (A), generative intent finding (B), prompt optimization (C), fine-tuning (D), and retrieval integration (E). The code for reproducing our pipeline can be found on GitHub[1].

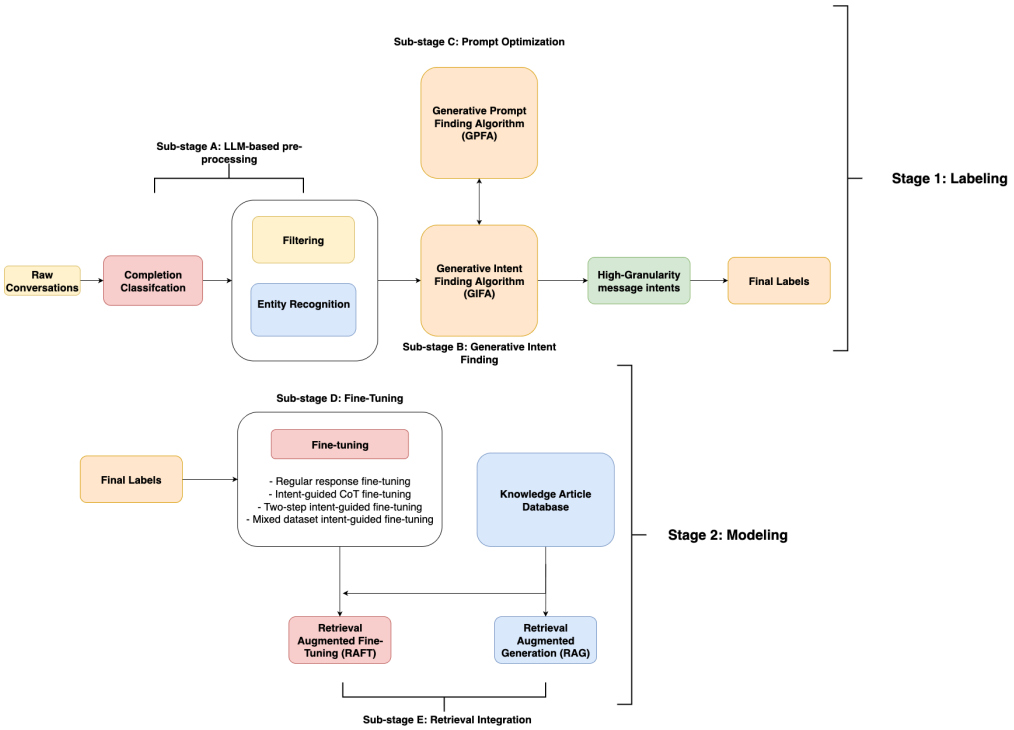

Figure 3.1: **Global framework overview.** 2-stage end-to-end framework. Stage 1: labeling, consisting of three sub-stages: (A) LLM-based pre-processing (anonymization of personal information, cleaning through message merging and redundancy removal, and completion classification to flag whether the customer's issue appears resolved); (B) GIFA LLM-based intent discovery; (C) GPFA LLM-based prompt optimization. Stage 2: modeling, consisting of two sub-stages: (D) fine-tuning strategies (response-only, CoT, mixed, 2-step); (E) retrieval integration (RAG/RAFT)

LLM-based pre-processing: Conversations from Albert Heijn are initially available in raw format. Basic and advanced pre-processing steps are required to obtain a clean

[1] `https://github.com/ramirosi/Intent-Guided-Fine-Tuning`

dataset. Specifically, an LLM is used to anonymize conversations, clean the customer queries and assistant responses, and determine whether a customer's problem was solved within the conversation. (Figure 3.1 sub-stage A, Section 3.1).

Generative Intent Finding Algorithm (GIFA): GIFA generates message-level intent labels directly from conversations. Unlike fixed-label intent classification methods, GIFA dynamically discovers intents using an LLM. Each message in a conversation is labeled with a short descriptive intent name, based on the message itself and its conversational context. These intents capture the semantic meaning of a message, which may promote generalization during fine-tuning. (Figure 3.1 sub-stage B, Section 3.2).

Generative Prompt Finding Algorithm (GPFA): GPFA adjusts the prompt used for intent discovery in GIFA. Since GIFA depends heavily on the prompt to instruct the LLM, GPFA acts as a meta-optimization loop that searches for the best-performing prompt using a labeled reference dataset. It evaluates candidate prompts based on clustering alignment with human-labeled intents and iteratively updates the candidate prompt through LLM-generated feedback. Multiple backbone search strategies are included in our work, including greedy search, random search, SEE search, and our novel Hyperband-SEE search. (Figure 3.1 sub-stage C, Section 3.3)

Fine-Tuning Setup: Multiple fine-tuning approaches are explored using the generated intents and the cleaned conversational data. These include standard next-turn generation, multi-stage learning, multi-task learning, and CoT fine-tuning. The intent annotations from GIFA are optionally used during the fine-tuning strategies to support response planning (Figure 3.1 sub-stage D, Section 3.4). Furthermore, the potential integration of RAG and RAFT is also discussed as part of the setup (Figure 3.1 sub-stage E, Section 3.5).

3.1 Conversational Dataset

This section corresponds to **sub-stage A: LLM-based pre-processing** in Figure 3.1, covering both the initial extraction/structuring of conversations and the subsequent LLM-based pre-processing steps.

3.1.1 Initial Data Pre-processing

In this work, we use a dataset of real-world customer service conversations collected from Albert Heijn. These are conversations between customers and human customer

service assistants. All conversations are stored in unstructured JSON format within a NoSQL database for a limited time. Due to privacy regulations, the conversations are automatically deleted every 30 days.

We extracted a dataset of 26,253 conversations. We filtered the selection to include only conversations involving a human assistant, excluding those resolved entirely by the chatbot, since our goal is not to fine-tune on chatbot responses.

The vast majority of digital customer questions come from the Netherlands, with a small proportion originating in Belgium. We removed the Belgian conversations because Albert Heijn has separate policy rules in both countries. All conversations are kept in Dutch since, at this time, Albert Heijn customer service does not officially provide English support.

Each conversation begins with a free-form query sent by the customer before interaction with the chatbot. This query is followed by a chatbot response, often including multiple-choice reply options. When a conversation is handed over to a human assistant, the initial customer messages are given as context. In our pre-processing pipeline, we prepend this initial query to each conversation to preserve conversational context.

Every message in the JSON is stored as an individual event with metadata identifying the sender (customer or assistant). During pre-processing, we added a "customer:" or "assistant:" prefix to each message, which is essential for the fine-tuning part and intent generation processes. Finally, we filtered out conversations containing only customer or only assistant messages, ensuring that all retained conversations were not one-way. Quantitative statistics on the dataset are provided in Table 3.1.

Statistic	**Mean**	**Standard Deviation**
Average words per conversation	182.39	130.14
Average turns per conversation	13.99	9.45
Average consecutive messages per role	1.59	0.53
Average messages per customer per conversation	7.44	5.45
Average messages per assistant per conversation	6.55	4.43

Table 3.1: Mean and standard deviation of conversation-level metrics, computed over the full Albert Heijn dataset consisting of 26,253 conversations.

3.1.2 LLM-based Data Pre-processing

After the initial data pre-processing, we apply another series of pre-processing steps to improve data quality, ensure privacy, and enable better model generalization during fine-

tuning. These steps include classifying incomplete conversations, cleaning up noisy message sequences, and masking sensitive or irrelevant information. We perform each of these pre-processing steps using GPT-4o.

Completion Classification: The current Albert Heijn setup labels a conversation as ”complete” once the chat window closes; this label does not necessarily mean the customer’s issue has been resolved. Some conversations end because the customer stops responding, the assistant times out, or no clear resolution is provided. We task the LLM with determining which conversations are resolved and which are not.

Message Merging and Cleanup: The message-level structure of raw conversations is not always well-formed. Customers often send follow-up messages that repeat earlier turns without adding new information (e.g., ”Hello?” or ”??”). In other cases, customers add minor additions (e.g., ”Please”, ”Now”, and ”Still waiting”) that should be merged into the previous message.

Using GPT-4o, we merge or concatenate similar or redundant customer messages within a short window and delete assistant system prompts such as handover notices (”My colleague is taking over”) or timeout warnings (”Are you still there?”). Transitions between assistants are removed to ensure each conversation contains interactions with only a single assistant. Empty or whitespace-only messages are discarded. Importantly, messages are never reordered—they are only merged, concatenated, or deleted.

Anonymization and Semantic Masking: Personal information and task-specific entities are masked to meet privacy requirements and prepare the dataset for fine-tuning. Fields such as names, phone numbers, addresses, and e-mails are masked, along with domain-specific entities to promote generalization. Appendix A provides a complete list of masked entities.

The three LLM-based pre-processing steps are performed using a single prompt (see Appendix A)). Anonymization and cleanup steps are applied directly to the messages, while the completion-classification task adds a Boolean label to each conversation. This label is then used to filter out cases where the customer issue remained unresolved. Figure 3.2 illustrates the LLM-based pre-processing pipeline.

3.2 Generative Intent Finding Algorithm (GIFA)

This subsection corresponds to **sub-stage B: Generative Intent Finding** in Figure 3.1. Here, we describe how GIFA annotates the conversations from sub-stage A with message-

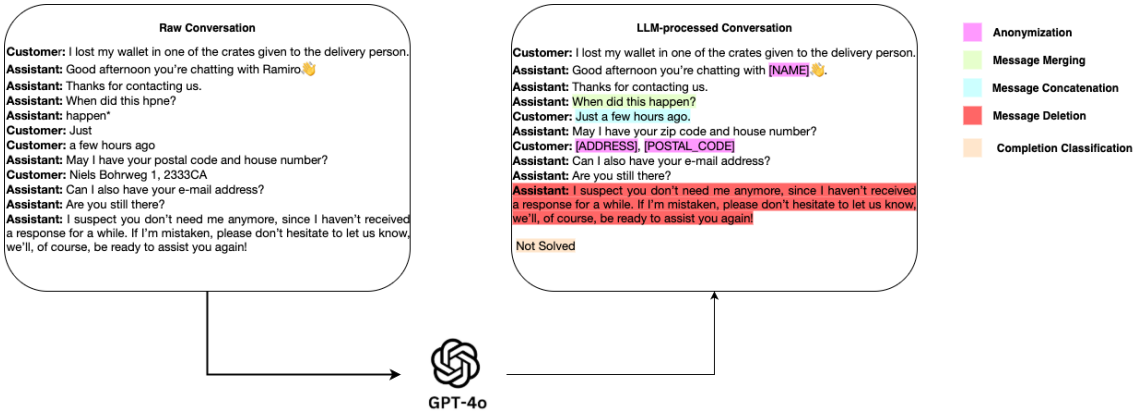

Figure 3.2: **LLM-based pre-processing example (sub-stage A).** Left: raw customer-assistant conversation. Right: the same conversation after GPT-4o pre-processing: anonymization of personal data (names, address, e-mail); message merging to collapse short back-and-forth utterances into complete turns; message concatenation to repair split sentences; message deletion of non-informative messages (e.g., time-out messages); and a completion classification flag indicating the conversation is solved or not solved. This processed output is the starting point for sub-stage B and C: intent labeling (GIFA/GPFA).

level intents.

Customer service messages often express the same intent with different wording and tone. Loss functions such as cross-entropy penalize lexical differences even when the semantics match. Furthermore, meaning is context-dependent: a message's meaning relies on the preceding messages, making labeling on the message level unreliable.

To address this, we introduce GIFA, which assigns context-aware, message-level intent labels to customer and assistant turns using a single LLM call over the whole conversation. Labels are short, human-readable descriptors (e.g., `report_missing_delivery`, `confirm_address`) that yield a structured but flexible representation that can be used for downstream tasks like fine-tuning.

GIFA maintains an expanding set of known intents, tying each new intent to the messages where it was first observed. When labeling a fresh conversation, it looks up the most similar previously labeled messages and gathers their associated intents. After removing duplicates, this forms a compact candidate intent set. This set and the whole conversation are inserted into the prompt so the LLM can consistently reuse existing labels and introduce new ones only when needed. The design promotes reuse and reduces label drift while keeping prompt length and thus cost under control. A prompt-and-output example is shown in Figure 3.3, and a high-level view of retrieval, prompt construction, labeling, and known-intent set updates appears in Figure 3.4.

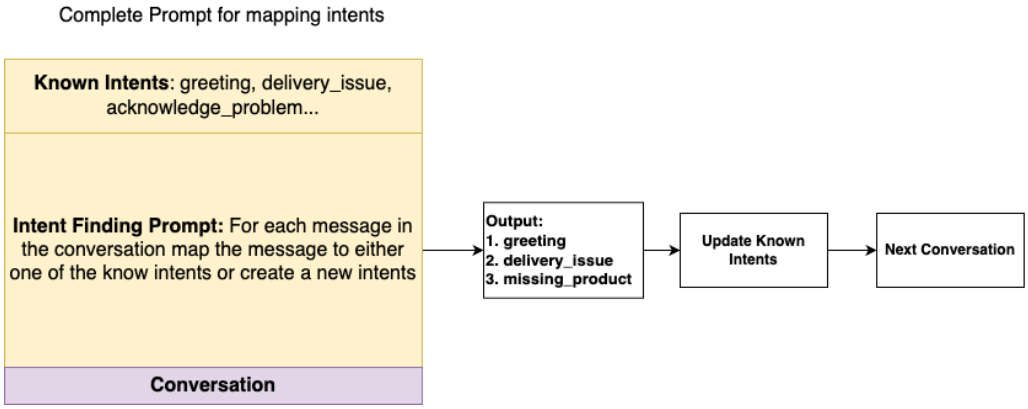

Figure 3.3: **GIFA prompt iteration (example).** A simulated conversation shows how retrieved intent names are inserted into the prompt and how message-level labels are created, updating the known-intent database each turn.

Using a retrieval mechanism, GIFA first selects relevant prior intents to construct the prompt. It retrieves semantically or lexically similar messages, based on message content alone (not prior intents). Three retrieval methods are supported:

- **Lexical (BM25):** Ranks messages based on token overlap and term frequency.
- **Semantic (ADA-003):** Embeds messages using the ADA-003 model and applies cosine similarity to identify similar messages.
- **Hybrid (RRF):** Combines the BM25 and ADA-003 scores using reciprocal rank fusion [53], aggregating lexical and semantic rankings into a single ranking.

The number of similar previous messages retrieved for each message in a new conversation is controlled by the hyperparameter k. For a conversation of x messages, each message is compared to the existing labeled database (using either lexical, semantic, or hybrid similarity). The top k most similar former messages are retrieved per new message, and from these retrieved messages, the linked intents are collected and merged into a deduplicated union of relevant intents. Finally, together with the new conversation, the relevant intents are added to the GIFA prompt. Once relevant intents are retrieved, GIFA supports three sampling strategies for how they are appended to the prompt:

- **Name:** Only the intent names are included.
- **General:** The intent names are combined with short general descriptions generated by an LLM.

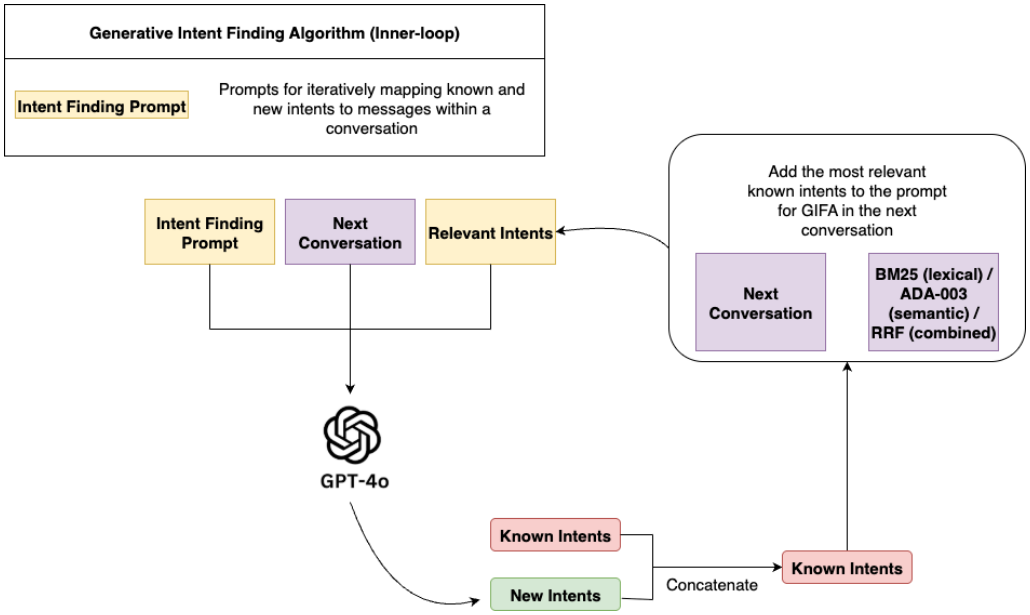

Figure 3.4: **GIFA framework.** Retrieval (BM25, ADA-003, or RRF) finds the top-k similar messages. Their linked intents are deduplicated and appended to the prompt, the LLM labels each message in the conversation, and the intent database is updated for the next conversation.

- **Example:** The prompt includes examples from previously labeled messages for each intent.

Examples of these strategies are shown in Table 3.2. Each strategy represents a different trade-off between flexibility and constraint. While a name allows for greater freedom in reuse, an example offers direct grounding but may overfit to specific phrasing, and the general strategy strikes a balance between the two. GIFA updates an auxiliary database of intent descriptions when the general sampling strategy is enabled. These descriptions are generated through a separate LLM call when a new intent is discovered.

3.3 Generative Prompt Finding Algorithm (GPFA)

This subsection corresponds to **sub-stage C: Prompt Optimization** in Figure 3.1. In this sub-stage, we refine the prompt used by GIFA (sub-stage B) to maximize intent clustering quality. We introduce GPFA and describe our backbone search strategies to explore the prompt space.

3.3.1 GPFA Framework

GPFA is a framework that automatically optimizes the prompt used by GIFA to improve intent clustering performance. It operates as a meta-level black-box search process: it

Sampling Method	Known intent examples to concatenate to the intent finding prompt
Name	`["confirm_understanding", "greeting", "offer_help"]`
General	`["confirm_understanding: Customer confirms the assistant's understanding of their request", "greeting: The assistant introduces themselves and welcomes the customer", "offer_help: Assistant offers additional assistance or resources to facilitate a process"]`
Example	`["confirm_understanding: Yes, that indeed is my problem.", "greeting: Hi, my name is [NAME] thank you for contacting customer service", "offer_help: I could send you an update by mail if I get a response from the department"]`

Table 3.2: **GIFA sampling methods.** Three ways to append retrieved intents into the prompt: *name* (intent name only), *general* (name + short description), and *example* (name + message examples), trading off flexibility and constraint.

generates candidate prompts, evaluates their effectiveness by running GIFA on a labeled dataset, and iteratively refines them. By treating prompt optimization as a search problem, GPFA removes the need for manual prompt engineering and adjusts prompt structure and phrasing to achieve the best clustering alignment with a ground-truth set of intents. Figure 3.5 provides an overview of GPFA.

The quality and behavior of GIFA are sensitive to the input prompt used. In addition to formatting rules (e.g., requiring `snake_case` for intent labels), the prompt also influences the granularity of generated intents. A prompt with overly broad rules can merge distinct messages into the same intent, while one with many constraints can produce too fine-grained intents, reducing reuse and generalization. We speculate that neither extreme is desirable for downstream tasks such as fine-tuning. The ideal prompt strikes a balance between granularity and generalizability. Designing such prompts manually is domain-dependent and costly; even small wording changes can shift GIFA's behavior. GPFA overcomes this limitation by automating the prompt optimization process and searching for prompts that generate intent clusters aligned with a labeled dataset. Because GIFA relies on a closed-weight model, GPFA treats the search as a black-box prompt optimization problem.

GPFA begins with one or more candidate prompts (depending on the backbone search

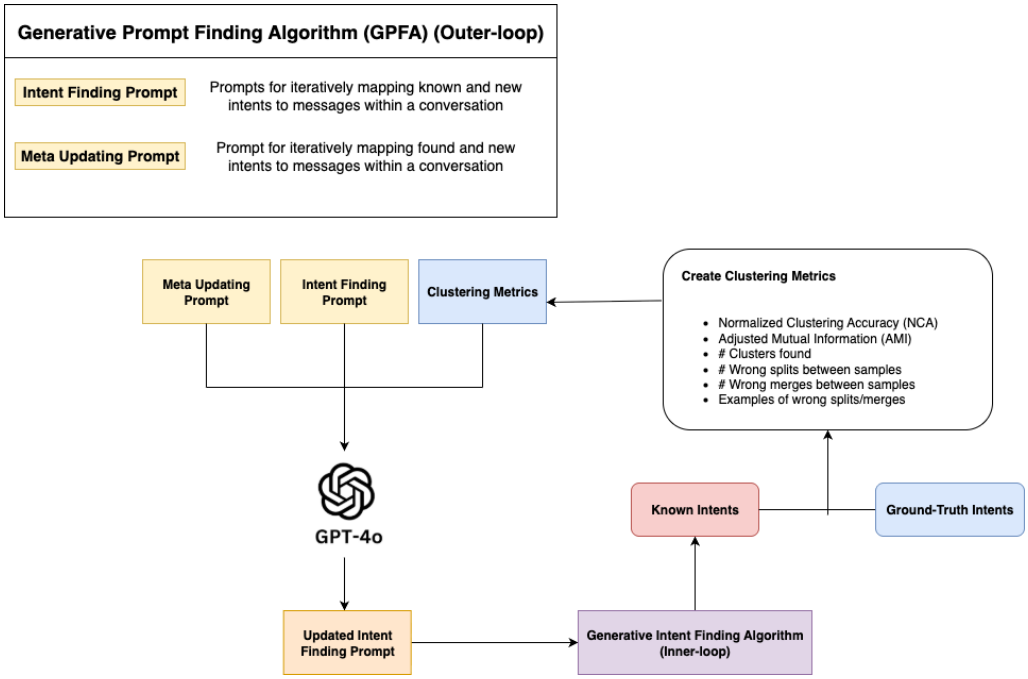

Figure 3.5: **GPFA framework.** Black-box prompt search that generates/evaluates candidate GIFA prompts on a labeled set using an LLM, and optimizes the product metric NCA$\times$AMI to align predicted clusters with ground-truth clusters.

algorithm), which may be initial guesses or refinements from earlier iterations. It evaluates each candidate by running GIFA on the labeled dataset. The LLM then generates intents for each conversation, producing clusters of messages that GPFA compares against ground-truth clusters from the human-labeled subset. We use normalized clustering accuracy (NCA) and adjusted mutual information (AMI) in this work.

GPFA incorporates the evaluation results into a meta-prompt for a second, meta-level LLM task that acts as an automatic prompt engineer. This meta-prompt contains quantitative scores (NCA and AMI), generated versus expected intent counts, and qualitative error examples. These include cases where two messages with the same ground-truth intent were assigned different predicted intents (false split) and messages with different ground-truth intents were grouped (false merge).

The meta-prompt is then given to the LLM, which can either analyze the evaluation results and suggest improvements to the prompt or directly output an updated version. A Boolean hyperparameter, $lagged\_feedback$, controls this behavior. The updated prompt is fed into the next iteration, where GIFA is rerun, evaluated on the dataset, and refined further. This loop continues for a fixed number of iterations or until no further improvements are found.

3.3.2 Backbone Algorithms for GPFA

An important choice within GPFA is how the prompt space is explored. The previous subsection showed how GPFA refines a single prompt using a feedback loop. However, it did not specify which prompts to evaluate at each iteration or how to choose among multiple candidates. To address this, GPFA can employ different search algorithms (called backbone search algorithms) to explore the prompt space. These strategies determine how new prompts are generated, which prompt(s) carry forward, and how to balance exploration (trying diverse prompts) with exploitation (focusing on the best prompts found so far). In this thesis, we explore four possible backbone algorithms.

Greedy Search

GPFA continually improves a single prompt in the greedy search configuration. The process begins with an initial prompt (e.g., a general human-made starting prompt). GIFA runs this prompt on the ground-truth dataset, and the evaluation results are used to generate LLM-based feedback. An LLM then updates the prompt based on this feedback. The updated prompt is compared against the previous best, and the higher-scoring prompt is retained for the next iteration. Figure 3.6 illustrates the greedy prompt search process.

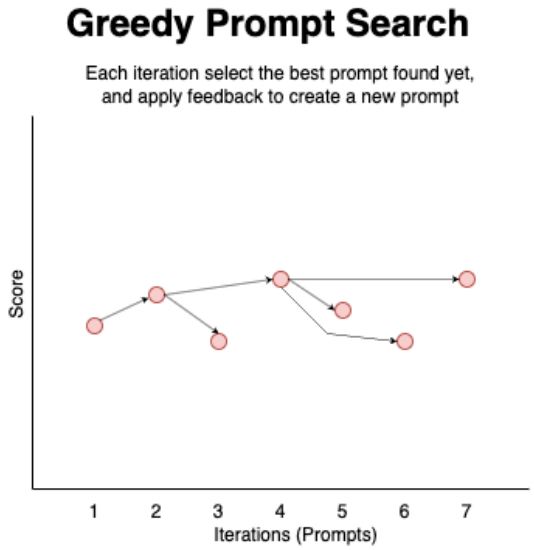

Figure 3.6: **Greedy prompt search.** Starting from a human seed prompt, iteratively keep only the best-scoring variant per round.

Random Search

Random search introduces controlled randomness into prompt optimization, encouraging exploration. Like greedy search, it begins with an initial prompt, evaluates it, and uses the results to generate LLM-based feedback for an updated prompt. The key difference lies in selecting the following prompt: instead of always keeping the best-performing prompt, random search samples from the pool of previously generated prompts. After generating a new prompt, the algorithm either continues with it or switches to an earlier prompt. This randomness promotes diversity in the search process and helps GPFA escape local optima. Over time, random search is expected to explore a wider variety of prompts than greedy search. Figure 3.7 illustrates this process.

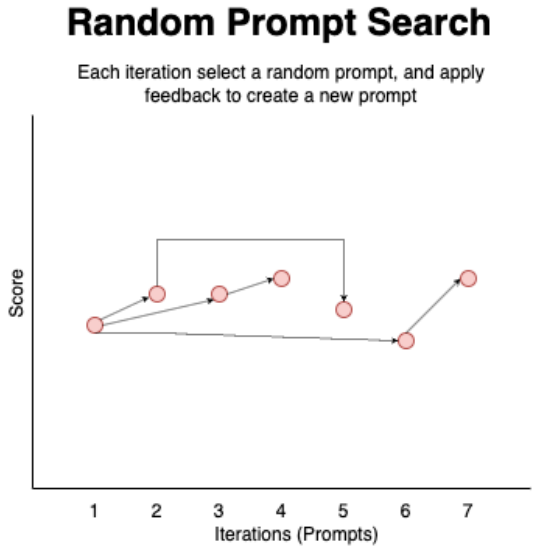

Figure 3.7: **Random prompt search** Sample and mutate prior prompts at random each iteration to explore widely.

Strategic Exploration and Exploitation (SEE)

SEE is a meta-heuristic search strategy that combines operations from genetic algorithms and Lamarckian evolution to explore the space of prompt candidates [14]. It is designed to maintain a diverse population of high-performing prompts and iteratively refine them across multiple feedback and crossover phases. SEE proceeds through three main phases, regulated by the parameter $n\_prompts$, which defines the number of prompts maintained and evaluated in each phase. Figure 3.8 provides an overview of SEE. The prompts used for the different phases in SEE are listed in Appendix E.

Lamarckian Initialization Phase: Unlike greedy or random search, which start from a

single human-written prompt, SEE generates an initial population of $n\_prompts$ through reverse engineering. A conversation is given to an LLM with instructions to produce a prompt that would generate the observed output, resulting in diverse initial prompts based on real examples rather than random constructions.

Localized Feedback Phase: In this phase, SEE uses generated feedback to refine each prompt in the current population individually. The process is applied independently to every prompt, and if multiple feedback rounds are set, all prompts from the previous round are refined.

Crossover Phase: After local refinement, SEE explores the search space further through recombination. New child prompts are generated by merging pairs of parent prompts selected from the best prompts seen so far. Unlike in the feedback phase, parents can be chosen from all prompts encountered during the Lamarckian and feedback phases. Two strategies for selecting parent pairs are supported:

- **Error-wise crossover:** Prompts are compared based on their clustering disagreement. The Hamming distance between their cluster assignments on the reference dataset is computed for each pair. Pairs exhibiting high disagreement, but still reasonably strong performance, are considered good candidates for crossover. These pairs are then merged by prompting the LLM to create a new prompt that combines the strengths of both parents.
- **Semantic-wise crossover:** Prompts are embedded using a sentence-embedding model (ADA-003), and each pair's cosine similarity is computed. Pairs with high semantic distance (dissimilar meanings) are selected to encourage diversity. The LLM is then prompted to merge the two parent instructions into a new child prompt that preserves the semantic meaning of both.

Hyperband-SEE

While SEE expands the search space through feedback and crossover, it incurs high computational costs like greedy and random search. Evaluating the clustering quality of a single prompt requires running GIFA over an entire dataset. To address this, we introduce a more resource-aware search strategy, Hyperband-SEE, which adapts the principles of the Hyperband algorithm to prompt optimization (Figure 3.9).

Hyperband [38], initially developed for hyperparameter tuning, is a bandit-based early stopping method that allocates limited resources across many configurations. The algorithm evaluates a wide set of candidates on small budgets (e.g., fewer training epochs or

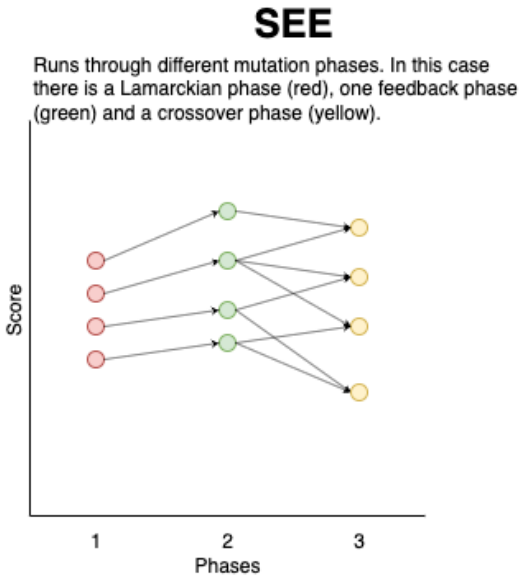

Figure 3.8: **SEE search** Population-based prompt evolution with localized feedback and recombination (semantic/error-wise crossover) and builds prompts from scratch and combines the strengths of prior candidates.

data samples) and iteratively prunes configurations that perform the worst while increasing the budget for the best-performing ones.

Hyperband-SEE applies the Hyperband principle to prompt search by evaluating many candidates on small budgets and gradually increasing resources for the most promising ones. Instead of running SEE across the complete dataset for every candidate, it incrementally increases the evaluation budget (number of conversations) while reducing the number of surviving prompts. The process unfolds as a multi-round tournament, with each round using a larger validation subset.

Each bracket in Hyperband-SEE defines a progressive allocation of validation samples across multiple rounds. In the early rounds, it evaluates prompts on small subsets of the labeled dataset, pruning many at once. In later rounds, it assigns larger budgets to the surviving prompts, increasing the reliability of the validation scores.

The motivation for using multiple brackets comes from the uncertainty in how many validation samples are needed to assess a prompt's performance accurately. If early rounds rely on too few samples, poorly performing prompts may survive simply due to poor estimates. On the other hand, using extensive validation sets too early limits the number of prompts that can be explored.

The complete SEE algorithm is executed within each bracket round, comprising the Lamarckian phase, one or more feedback phases, and crossover phases. Note, however,

that Lamarckian prompt generation occurs only in the first round of each bracket, when no prior prompts exist to inherit. In subsequent rounds, prompts that survive elimination are re-evaluated using the larger validation budget. All rounds within a bracket share the same SEE hyperparameter configuration (e.g., number of feedback iterations k_1, number of crossover phases k_2, and the chosen crossover strategy).

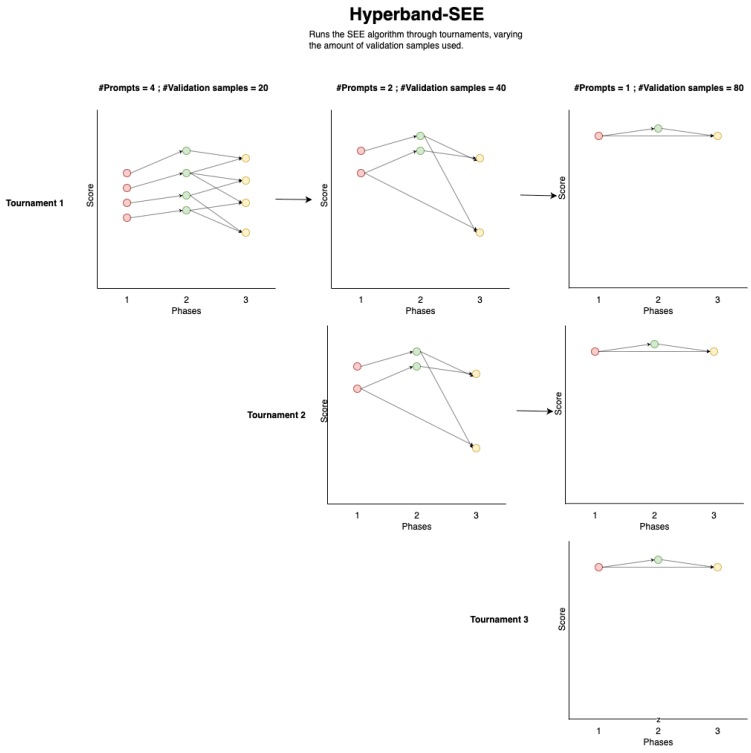

Figure 3.9: **Hyperband-SEE.** Hyperband wrapper around SEE that allocates/early-stops candidates using partial validation, pruning weaker prompts to focus budget on promising ones.

3.4 Fine-Tuning Procedure

This subsection corresponds to **sub-stage D: Fine-tuning** in Figure 3.1. We use the generated intents to fine-tune open-weight models for customer-service response generation.

3.4.1 Data Preparation

The conversational dataset we created in sub-stage A contains multiple consecutive messages from the same role (customer or assistant). To create a cleaner and more consistent training structure, we collapse all consecutive messages from the same role into single blocks. This transformation ensures that each turn contains a complete set of queries or responses.

We set up the fine-tuning process in the Albert Heijn co-assistant setting. For every training sample, we provide the model with the full ground-truth conversation history up to the last customer query. We then task the model with predicting the following assistant response or set of responses (when collapsed). This setup aligns with the real-world scenario where a customer support assistant sees the complete conversation history and must provide an appropriate response. The total number of collapsed assistant responses available for fine-tuning is 116,906.

3.4.2 LLM Choice

We use Qwen3-1.7B as the base model for fine-tuning. The choice is motivated by three primary reasons:

1. 1.7B parameters are relatively small compared to other open-weight models, and enable fine-tuning with limited GPU resources.
2. It is a multilingual model, and supports Dutch, which is a strict requirement since all of the conversations inside the dataset are in Dutch.
3. The instruction-tuned model has a thinking mode, which enables structured output (thinking before answering). The thinking mode is interesting when using fine-tuning techniques such as CoT.

3.4.3 Training Strategies

We set up five fine-tuning variants to evaluate different learning methods on the conversational dataset. Each variant represents a unique fine-tuning approach. Figure 3.10 summarizes the input structure, prompt templates per variant, the optional RAFT knowledge block, and the supervised targets for each setup.

1. **Response-Only:** Fine-tuned directly on assistant responses. The model is provided with the conversational history and learns to generate the following assistant messages. This setup is not intent-guided.
2. **CoT (Intent Chain $\rightarrow$ Response Chain):** This CoT-inspired method uses the model's built-in thinking mode. For each assistant turn, the model is trained to first predict an intent list (a list of assistant response intents) inside a thinking block, followed by the list of assistant response(s). This design encourages the model to set up a response plan before generating the responses.

3. **2-Step (Intent → Response):** A two-phase training process. First, a LoRA adapter is trained to predict intents for each assistant message, learning how to generate intents given a response. Afterwards, a second adapter is trained to directly output responses. This variant tests whether promoting intent understanding boosts response quality.

4. **2-Step (Intent → CoT):** Unlike the previous variant, the second phase trains the model to output the intent chain and responses using the CoT format. The idea here is that since intent names require a small amount of tokens compared to free-form responses, with direct CoT, it might be the case that the loss function does not penalize mistakes in the generated intent list enough. The model is first trained to generate only the intent to address this. Once it has learned this, the task is extended to include response generation, making the overall objective more challenging.

5. **Mixed Ratio (Multi-Task Learning):** In this setup, the 2-step variants are merged into one training step by mixing the dataset. A single model is fine-tuned on a 30/70 curriculum: 30% of samples train intent prediction, and 70% train CoT intent-response generation. Compared to the 2-step variants, it requires less training time and could omit catastrophic forgetting, which might occur in multiple-stage fine-tuning. A 30 / 70 split between the direct intent prediction and the CoT task was chosen, motivated by the relative difficulty of the two tasks. Since direct intent prediction is simpler, a smaller proportion of the curriculum was allocated, allowing the model to devote more training to the more challenging CoT setup. We did not perform an ablation study to optimize this ratio, but we consider it a reasonable choice for this setting.

3.5 Retrieval-Augmented Integration

This subsection corresponds to **sub-stage E: retrieval integration** in Figure 3.1. Here, we cover RAG and RAFT, which leverage external knowledge sources to improve model grounding and factual accuracy. As illustrated at the top of Figure 3.10, we implement RAFT by prepending retrieved knowledge to the conversation context with simple rules for using it.

Retrieval-based methods were incorporated into the modeling pipeline to enable more grounded and contextually accurate response generation. We focus on two mechanisms: RAG, which appends external knowledge at inference time, and RAFT, which integrates retrieval context during the training phase. Together, these approaches simulate the real-world scenario at Albert Heijn, where a co-assistant is expected to generate responses based on conversation history and relevant knowledge articles.

3.5.1 Knowledge Sources and Corpus Construction

We used two structured sources of domain knowledge to simulate the retrieval environment:

- *Frequently Asked Questions (FAQs):* A set of question-answer pairs publicly available on the Albert Heijn website.
- *Knowledge Articles:* Internal and external support articles containing structured guidance for customer-facing issues. These articles include metadata indicating validity periods, updates, and deprecation history.

While FAQs reflect the current situation, they lack the historical accuracy required to evaluate past conversations. Therefore, we used time-filtered knowledge articles as the primary source for retrieval and filtered them based on the following criteria:

- Knowledge articles must be externally visible (i.e., not internal HR or system documents).
- Content must be relevant to the Dutch market and aligned with filtered Dutch-language customer conversations.
- Knowledge articles must belong to Albert Heijn, excluding other brands under Ahold Delhaize.
- The article must be valid during the conversation, determined by matching timestamps with article metadata.

Each selected knowledge article was cleaned with regular expressions to remove HTML markup and converted into structured (question, answer) pairs. We ultimately constructed 2,221 unique QA pairs with an average answer length of 35.88 words.

3.5.2 Relevance Matching and Retrieval Pipeline

We construct a retrieval mechanism to simulate a real-world retrieval scenario based on semantic similarity between customer queries and knowledge base questions. The process consists of the following steps:

1. **Query Extraction:** The customer's first conversational turn is collapsed into a single query.

2. **Embedding:** The collapsed query and all knowledge article questions are embedded using the ADA-003 model to obtain dense semantic vectors.

3. **Similarity Computation:** Cosine similarity is computed between the query and each knowledge article entry.

4. **Top-k Selection:** The top three most similar QA pairs are retrieved and appended to the prompt.

This approach is based on Albert Heijn's current co-assistant GPT-4o RAG system, which retrieves the top three knowledge articles and appends them to the prompt.

In the RAG setup, we prepend the top retrieved QA pairs to the input prompt during inference. Since the model has not seen these entries during training, it must rely on its internal knowledge to extract relevant information from the QA pairs. This setup reflects the current co-assistant system, where relevant knowledge is added to the prompt without adapting the model parameters.

While RAG enhances generation by providing access to external context, it assumes the model can correctly determine the relevant context. RAFT addresses this limitation by incorporating the retrieved QA pairs into the training samples. This allows the model to learn when to rely on the retrieved content, how to ground its responses in it, and when to ignore irrelevant information. In this research, we implement RAFT across the best-performing fine-tuning setups to directly compare it with RAG and setups without retrieval.

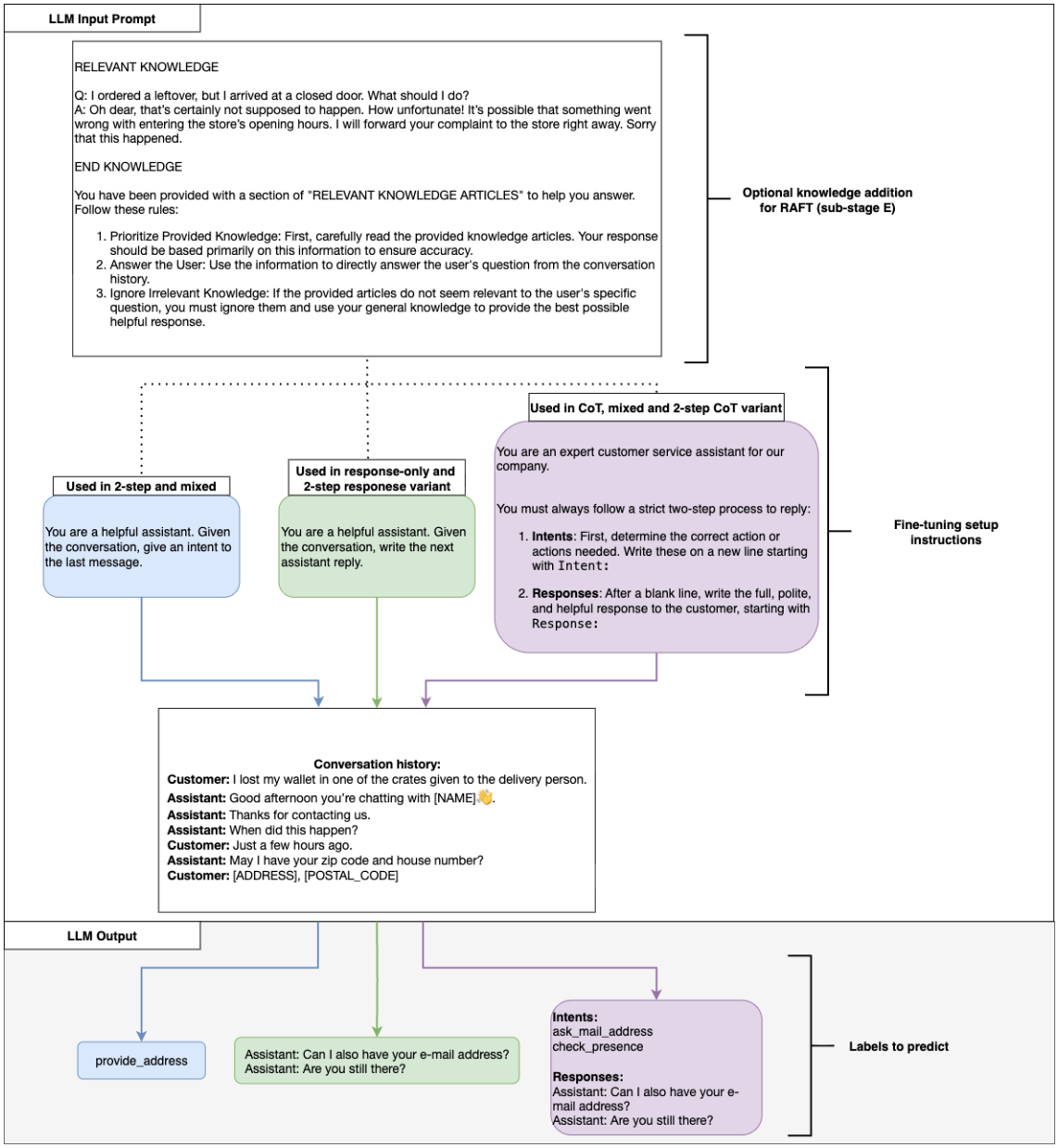

Figure 3.10: **Fine-tuning prompt formats and supervision targets (stage 2).** The model input is the LLM-processed conversation history; for RAFT (sub-stage E), an optional block of retrieved knowledge is prepended with rules to prioritize it. The instruction template varies by setup: response-only / 2-step response (green) asks for the next assistant reply; 2-step and mixed (blue) ask for a message-level intent; CoT, mixed, and 2-step-CoT (purple) enforce a chained output with `Intents:` followed by `Responses:`. The bottom panel shows the labels supervised for each setup (intent names and/or the next replies).

4 Experiments regarding GIFA & GPFA

In this section, we establish the optimal setup for generating LLM-based intent labels in stage 1, which we later use in the fine-tuning experiments of stage 2. We begin with the pre-processed Albert Heijn conversations from sub-stage A and construct a human-labeled evaluation set of 250 conversations with high-granularity intent labels. This set serves as the basis for optimizing the GIFA and GPFA settings. The effect of the pre-processing choices: anonymization, cleaning, and handling incomplete conversations is evaluated separately in subsection 5.4, since assessing their impact requires the fine-tuning setup.

We optimize the setup through a sequential process split into phases, where each phase builds on the outcomes of the previous one. In phase 1, we tune retrieval configurations for each sampling method within GIFA to maximize intent reuse (subsection 4.2). Building on these settings, phase 2 introduces GPFA to optimize prompts with different prompt search algorithms. At the same time, we investigate the effects of sampling strategies, lagged feedback, and, in the case of SEE and Hyperband-SEE, the depth and type of feedback and crossover prompt updates (subsection 4.5). Finally, in phase 3, we compare the optimized search algorithms to identify the most effective overall intent labeling strategy (subsection 4.6). We then apply the resulting prompt within GIFA to label all 26,253 pre-processed conversations, creating the dataset used in stage 2: modeling (subsection 4.7). Figure 4.1 shows a complete stage 1 experimental setup overview.

RQ coverage. This section addresses **RQ-A2**, which asks how effectively the GIFA and GPFA combination can generate intents of the right granularity, and **RQ-A3**, which asks whether Hyperband-SEE improves prompt optimization efficiency and labeling quality over SEE and simple baselines.

4.1 Human-Labeled Intent Dataset

To evaluate the performance of GIFA and GPFA, we have constructed a labeled evaluation for 250 conversations chronologically sampled from the pre-processed, cleaned conversations dataset. This intent-labeled dataset contained high-granularity intent labeling, capturing fine distinctions in message goals when appropriate.

We assigned a level of granularity to each message based on its importance. General-purpose messages such as greetings and acknowledgments were grouped under broad

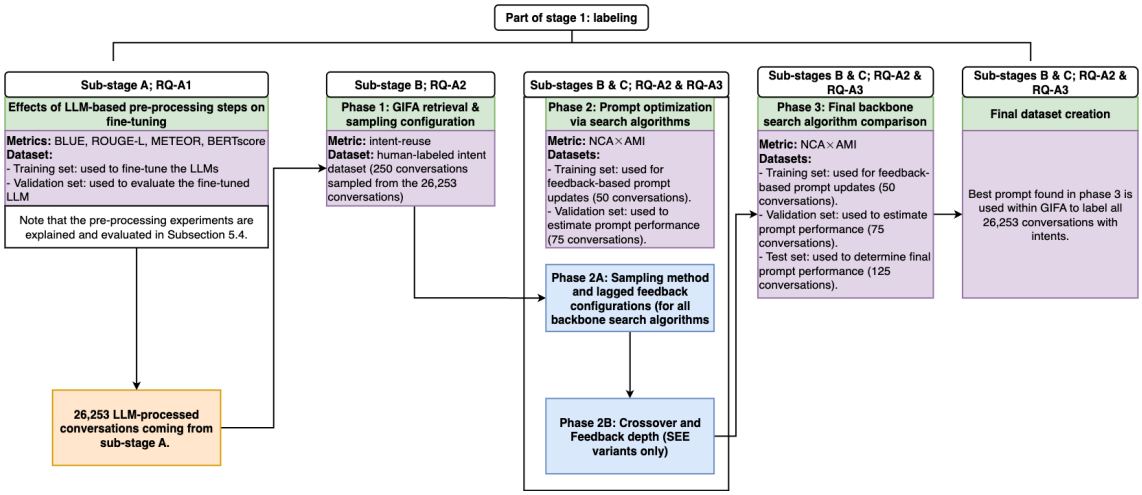

Figure 4.1: **Stage 1: labeling experiments.** Sequential setup for constructing the intent-labeled dataset. LLM-based pre-processing (evaluated in subsection 5.4); phase 1: GIFA retrieval configuration for different sampling methods (optimize intent reuse); phase 2: GIFA+GPFA prompt optimization where optimal sampling methods, lagged feedback, and the amount and type of crossover and feedback rounds for SEE/Hyperband-SEE are determined; phase-3: final comparison of the prompt search algorithms under their best configurations. The best prompt from phase 3 is then applied within GIFA to label all 26,253 conversations, producing the final dataset used in stage 2.

categories like `greeting` and `acknowledge_problem`, since additional detail offered little benefit. In contrast, subtle but meaningful distinctions, such as requesting a refund for damaged versus spilled items, were treated as distinct intents, because they required different assistant actions. Across 250 conversations consisting of 6,781 messages, the annotation produced 1,193 unique intents. Figure 4.2 illustrates the number of unique intents and their frequencies.

These high-granularity labels support the central goal of our work: enabling models to understand and express the semantic purpose behind each message rather than predicting predefined intent categories. By looking at intent prediction as a generative task rather than a classification problem, we encourage the model to learn the meaning and structure of the intents themselves, possibly allowing for more informative output.

While coarser intent groupings (i.e., medium-granularity labels) may cover more messages per intent, they can result in critical distinctions in customer goals. In an ablation study, we compared the performance on medium- and high-granularity label sets to assess whether medium-level grouping offers practical benefits. Results of our analysis, including clustering performance of generated intents, are included in Appendix B.

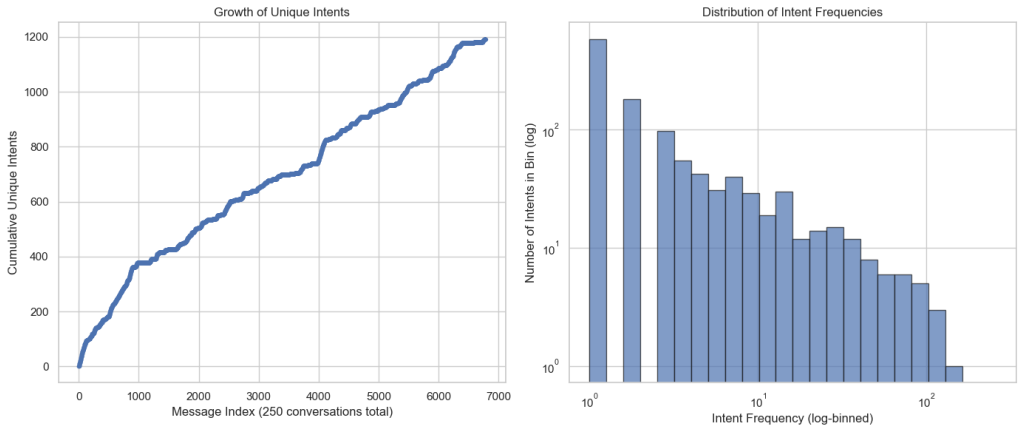

Figure 4.2: **Human-labeled intent dataset.** Intent growth and frequency. The left plot shows the semi-linear growth of unique intents over conversations. The right plot shows the long-tail distribution of intent frequencies, binned logarithmically: bin edges are spaced evenly on a $\log_{10}$ scale between the minimum and maximum observed intent frequencies (25 bins total).

4.2 Phase 1: GIFA Retrieval and Sampling Configuration

Experimental setup In the first phase, we set up an experiment to investigate how different configurations for retrieving prior intents influenced intent reuse during intent generation by GIFA. Our goal was to determine which retrieval method and depth yielded the most consistent intent labels, measured by the number of unique intents generated across the messages in the labeled dataset.

To isolate the effect of retrieval settings, we fixed two variables during this phase:

1. A static human-made prompt held constant across all retrieval settings (Appendix C).
2. A fixed sampling method (either *name*, *general*, or *example*), which determines how retrieved intents are formatted in the prompt.

This setup ensured that any change in the number of discovered intents is solely due to the retrieval configuration.

In addition, we included an extra initial experiment to evaluate the effect of using retrieval. In this experiment, we compared all retrieval configurations against a no-retrieval

baseline, where the prompt contained every previously found intent up to the token limit of GPT-4o.

We varied and tested the following parameters:

1. **Retrieval Method:**
   - Lexical: BM25-based ranking over token overlap
   - Semantic: Cosine similarity using ADA-003 sentence embeddings
   - Hybrid: A combination of both via Reciprocal Rank Fusion (RRF)
2. **Retrieval Depth (k):**
   The top-k most similar messages from the known-intent dataset were retrieved for each message in the target conversation. The set of unique intent labels associated with these messages formed the relevant intent list for that message. We tested $k \in \{1, 2\}$.
3. **Sampling Method:**
   - *Name*: Only intent names
   - *General*: Intent names plus short general descriptions
   - *Example*: Intent names plus message examples
4. **Few-Shot Examples per Intent (n):**
   Only applicable when *example* was set as the sampling method. This parameter defines how many examples are added to each relevant intent. We tested $n \in \{1, 2\}$.

The success criterion for this phase was intent reuse, measured by the total number of unique intent labels generated by GIFA over the labeled dataset of 250 conversations (6,781 messages). Since this phase was unsupervised, we compared the different search setups by their reuse rate: a lower number of unique labels indicated better reuse of known intents and fewer redundant or semantically overlapping labels.

Three retrieval methods, three sampling strategies, and two values of k (plus two values of n when *example* is used) resulted in 24 experiment configurations that had to be tested. We ran each configuration with a different random seed five times to account for LLM randomness.

The goal of this experiment was to answer the following questions. First, we assessed whether retrieval improved intent reuse compared to a no-retrieval baseline. Then, for each sampling strategy independently:

- Which retrieval method results in the highest intent reuse?
- What is the optimal value of k for retrieving relevant messages?
- For the example sampling strategy, how many few-shot examples (n) should be included per intent?

We selected the best-performing retrieval settings for each sampling strategy and fixed them for the following experimental phase, focusing on prompt optimization and sampling methods.

Results: We report outcomes in the following order: retrieval off versus on (Figure 4.3), then the optimal search setup per sampling strategy: *example* (Figure 4.4), *general* (Figure 4.5), and *name* (Figure 4.6). The metric used is the number of unique intent labels (lower = better).

Figure 4.3 compares all retrieval-enabled configurations against a no-retrieval baseline. The figure implies that turning off the search consistently resulted in more unique intents than any retrieval-based setup. This suggests that retrieval, regardless of configuration, increases intent reuse, which aligns with promoting reuse and minimizing noise within the prompts.

When we set the sampling method to *example*, we varied the retrieval method, the retrieval depth (k), and the number of examples per intent (n) (Figure 4.4), and clear patterns emerged. Configurations that included two examples per relevant intent ($n = 2$) led to better reuse (fewer intents found) than those that used only one example ($n = 1$). This suggests that richer contextual input supports more consistent reuse. Among the retrieval methods, semantic search consistently yielded the best reuse, followed by hybrid, with lexical performing worst. Furthermore, within semantic configurations, retrieving fewer intents ($k = 1$) resulted in fewer unique intents than $k = 2$, suggesting that adding more retrieved intents introduces more variation and potential noise into the prompt.

A similar pattern was observed under the *general* sampling strategy (Figure 4.5): semantic retrieval with $k = 1$ led to the fewest unique intents. Lexical retrieval performed worst, and hybrid retrieval showed intermediate performance. While hybrid retrieval and semantic retrieval with $k = 2$ performed similarly in reuse, we selected semantic retrieval with $k = 1$ as the preferred configuration because it reduced token cost and required less computation.

The results deviated slightly in the *name* sampling strategy (Figure 4.6), where only intent names were included in the prompt. Here, lexical retrieval led to the poorest reuse, and

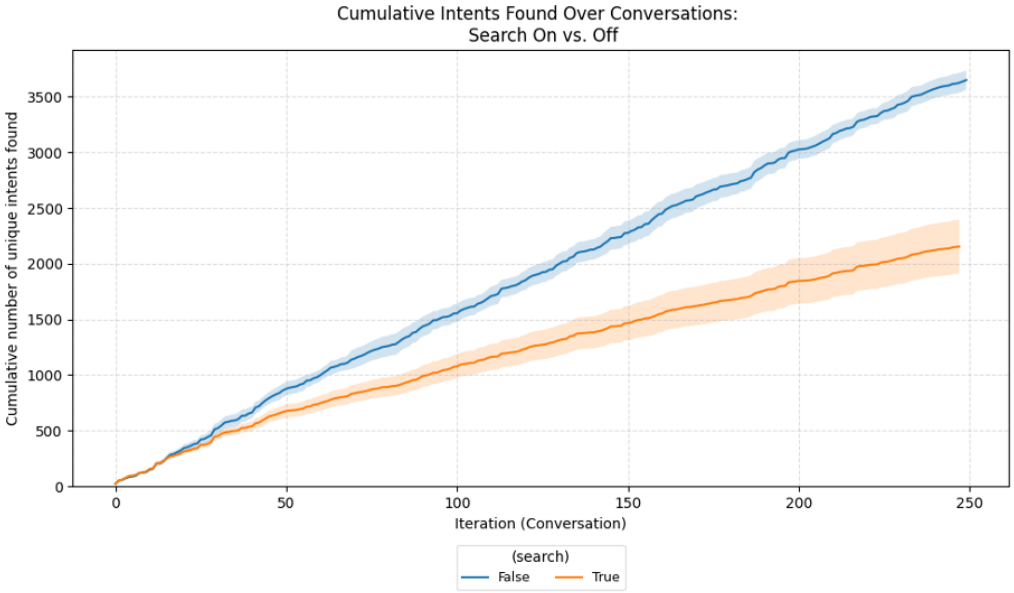

Figure 4.3: **Retrieval on vs. off for intent reuse.** On the 250-conversation labeled set, enabling retrieval (any method) yields fewer unique intents than a no-retrieval baseline (prompt listing all past intents within token limits), thus search promotes known intent reuse.

increasing k from 1 to 2 improved reuse, the opposite of what was observed with *example* and *general* sampling. This difference is likely due to the compactness of the name format: even when multiple intent names were added, the prompt remained concise. Thus, increasing the number of retrieved intents ($k = 2$) provided more relevant intent signals without overwhelming the LLM. Among these, we selected semantic retrieval with $k = 2$ as the best-performing strategy, preferring it over hybrid because it required less computation.

Finding → RQ-A2. Retrieval improved intent reuse in GIFA across all sampling strategies; the best configurations were *example*: semantic k=1, n=2; *general*: semantic k=1; *name*: semantic k=2. In conclusion, retrieval improves intent reuse across all sampling strategies. The optimal retrieval configuration is dependent on the sampling method.

We carried these best-performing setups into the next experimentation phase, refining prompt composition and sampling methods.

4.3 Clustering Evaluation Metrics

We evaluated the quality of the intent clusters generated by GIFA and GPFA using two complementary clustering metrics: Normalized Clustering Accuracy (NCA) [54] and Ad-

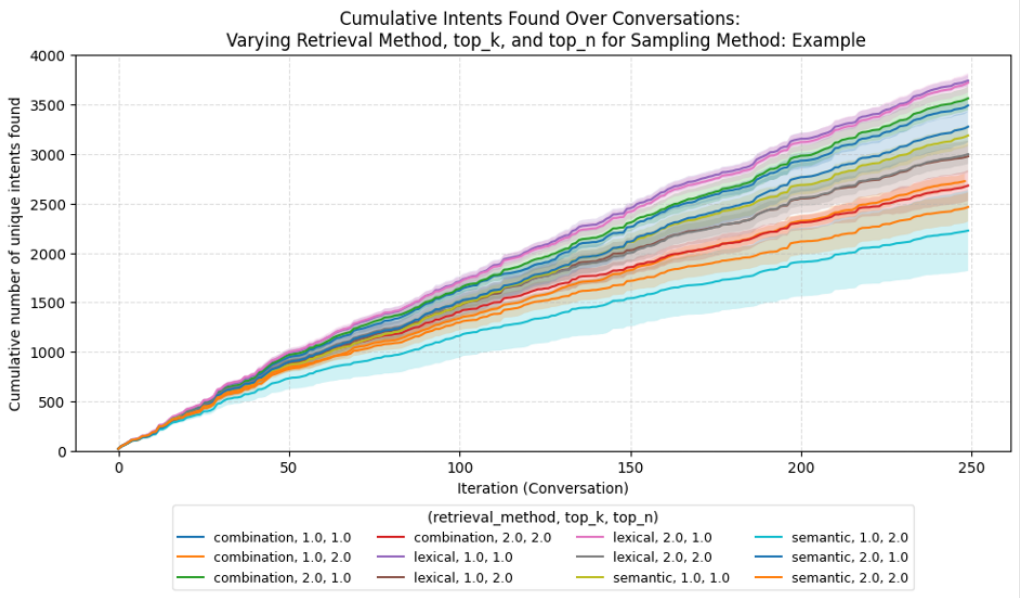

Figure 4.4: **Example sampling: retrieval method, k , and n.** With *example* sampling, semantic retrieval (ADA-003) at $k = 1$ and $n = 2$ examples per intent minimizes unique intents; hybrid (RRF) is middling, lexical (BM25) worst; $k = 2$ adds noise. Five seeds on 250 conversations. Here, k is the retrieval depth (number of top-retrieved messages whose intent labels are added to the GIFA prompt), and n is the number of message examples appended per retrieved intent in the prompt.

justed Mutual Information (AMI) [55]. Each metric captures a different aspect of clustering performance, and together they provide a balanced evaluation of both granularity and correctness.

NCA measures local cluster purity by assessing, for each ground-truth intent cluster $c_i \in C$, how well its messages are grouped within a single predicted cluster $p_j \in P$. The NCA formula is given in Equation 4.1.

$$\mathrm{NCA}(C, P) \;=\; \frac{1}{|C|} \sum_{c \in C} \max_{p \in P} \frac{|c \cap p|}{|c|} \tag{4.1}$$

Where $C = \{c_1, \ldots, c_{|C|}\}$ represents the set of true intent clusters and $P = \{p_1, \ldots, p_{|P|}\}$ the predicted clusters. NCA ranges from 0 to 1, with 1 indicating perfect alignment. NCA normalizes per ground-truth class before averaging, ensuring frequent intents (e.g., greetings or closings) do not dominate the score. This makes it particularly suitable for this project, where rare and fine-grained intents are just as, or sometimes even more, important than common ones.

Adjusted Mutual Information (AMI) is also used to complement NCA. AMI measures

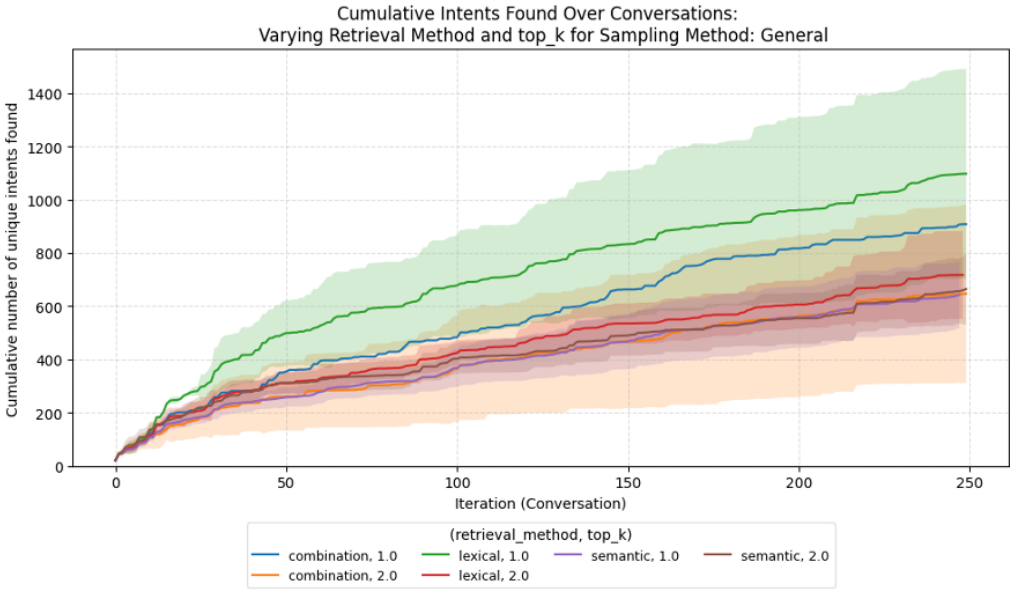

Figure 4.5: **General sampling: retrieval method and k.** Best reuse with semantic $k = 1$ and hybrid $\approx$ semantic at $k = 2$. Lexical search performs the worst. Five seeds on 250 conversations. Here, k is the retrieval depth (number of top-retrieved messages whose intent labels are added to the GIFA prompt).

global agreement between the predicted clustering and the ground truth, while correcting for chance. AMI is based on mutual information $I(C;P)$ and entropy $H(\cdot)$, and the formula can be found in Equation 4.2.

$$\mathrm{AMI}(C,P) = \frac{I(C;P) \;-\; \mathbb{E}\big[I(C;P)\big]}{\max\big\{H(C),\, H(P)\big\} \;-\; \mathbb{E}\big[I(C;P)\big]} \tag{4.2}$$

The resulting AMI scores lie in the $[-1, 1]$ range, with 1 indicating perfect clustering, 0 equivalent to random assignments, and negative values representing worse-than-random clustering. Unlike NCA, AMI is not normalized per class; instead, it assesses the overall structure of the clustering, making it helpful in detecting over-merging or over-fragmentation of clusters.

Because NCA and AMI evaluate different yet complementary aspects of clustering, local correctness, and global structure, we optimize their product during prompt optimization. This product ensures strong performance in both metrics, since it penalizes cases with high AMI but low NCA (over-generalization) or high NCA but low AMI (over-fragmentation). This encourages the model to generate semantically correct clusters with the expected granularity.

Alternative metrics such as Normalized Mutual Information (NMI) and Adjusted Rand

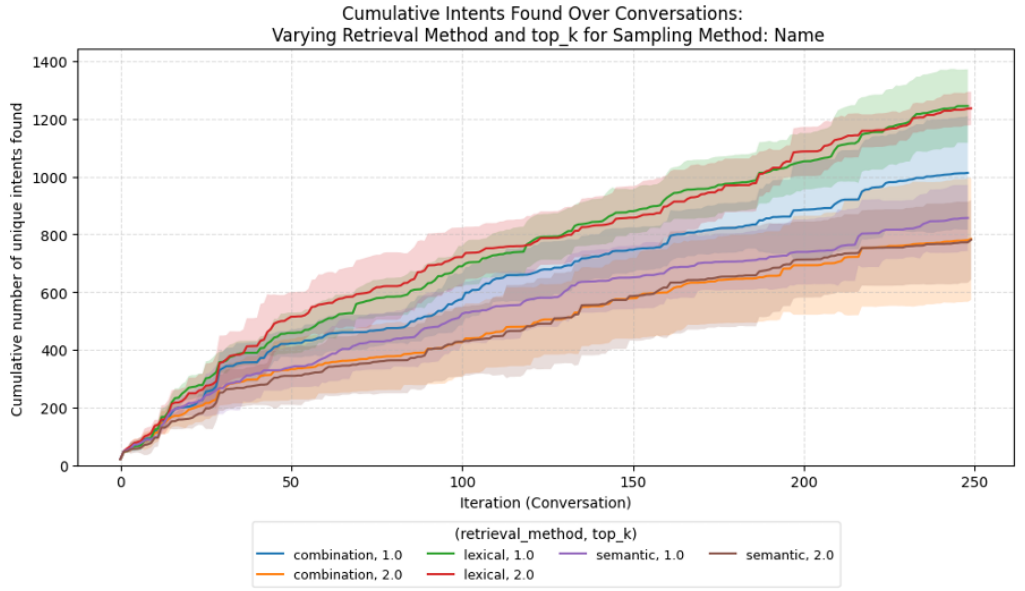

Figure 4.6: **Name sampling: retrieval method and k.** Best reuse with semantic $k = 2$ (slightly deeper context helps for names); hybrid in-between; lexical worst. Five seeds on 250 conversations. Here, k is the retrieval depth (number of top-retrieved messages whose intent labels are added to the GIFA prompt).

Index (ARI), which are often used as clustering metrics, were considered but not used. NMI lacks the chance adjustment, making it sensitive to cluster frequency imbalance. At the same time, ARI assumes equality between the number of predicted and ground-truth clusters, which is not true in our intent discovery task. Thus, we combined NCA and AMI for the following intent generation experiments.

4.4 Dataset Splitting Strategy

To ensure consistent evaluation across all prompt optimization experiments, we split the intent-labeled dataset into three distinct subsets using chronological ordering (since the human labeling was also done chronologically):

- **Training Set (50 conversations)**: Used exclusively for creating feedback during prompt optimization. Prompts may be refined based on model performance and examples belonging to this subset. A training set used during prompt optimization is different from fine-tuning an LLM. Only a small pool of conversations is needed to gather feedback; therefore, the training set is smaller than the validation and test sets.
- **Validation Set (75 conversations)**: Used throughout the optimization process to

evaluate the relative quality of candidate prompts. All hyperparameter tuning scoring relies on this set.

- **Test Set (125 conversations)**: Held out entirely during development. This set was used only to evaluate the best prompt discovered by each search algorithm (phase 3).

This separation ensured that feedback-driven refinement does not leak into evaluation and that final performance reflects generalization to unseen examples.

Note that this dataset split was not used during phase 1 of the experiments, which focused on comparing GIFA setups. The entire labeled dataset was used in phase 1 since it did not involve any optimization.

4.5 Phase 2: Prompt Optimization via Search Algorithms

After identifying the optimal configurations for intent retrieval and sampling in phase 1, we focused the second phase on evaluating and optimizing the prompt search process within GPFA. In this phase, we replaced the static prompt used in phase 1 with dynamically evolving prompts, which we optimized through black-box prompt search strategies guided by feedback on clustering quality. The aims of this phase were:

- To compare the effectiveness of different prompt search algorithms in optimizing clustering performance.
- To analyze how sampling strategies and feedback mechanisms interact with these search methods to affect final prompt quality.

Based on the outcomes of phase 1, we kept the retrieval configuration constant in this phase. In phase 2, we aimed to isolate and measure the contribution of prompt optimization techniques. We evaluated each prompt search strategy across multiple configurations, which resulted in two sub-experiments.

4.5.1 Phase 2 Sub-Experiment A: Sampling Method and Lagged Feedback

Experimental setup: In this sub-experiment, we compared four prompt search algorithms: greedy search, random search, SEE, and Hyperband-SEE. We applied each algorithm under multiple sampling methods (*name*, *general*, *example*) and with or without lagged feedback.

For each combination of:

- **Prompt search algorithm** $\in$ {greedy, random, SEE, Hyperband-SEE}
- **Sampling method** $\in$ {*name*, *general*, *example*}
- **Lagged feedback** $\in$ {True, False}

We performed a full optimization run with 24 configurations. Each run repeatedly evaluated GIFA outputs on the validation set and updated the prompts in every iteration.

This sub-experiment aimed to determine the best prompt configuration for each algorithm and to assess:

- Whether lagged feedback leads to better prompts.
- Which sampling strategy results in more effective prompt learning?

Results We evaluated the NCA$\times$AMI score under the phase 1 retrieval setup, averaging five seeds per configuration. We first compare sampling strategies across backbones and then examine the effect of lagged feedback.

The results presented in Figure 4.7 show that the *name* and *general* sampling methods consistently outperformed the *example* method across all prompt searching algorithms. The *name* strategy performed slightly better than *general*, suggesting that allowing the model to interpret intent meaning based solely on intent names enables more effective optimization. In contrast, the *example* method constrained the model by limiting intent assignment to specific examples, which reduced generalization. The *general* method, offering intent descriptions, provided a middle ground but remained less effective than using names.

Turning on lagged feedback resulted in lower scores and increased variance. This could be due to the dual LLM calls required to generate and apply feedback, which introduce additional stochasticity into the optimization process. When lagged feedback was disabled, the prompt search produced more stable and higher-quality results. This trend held across all sampling strategies.

Finding $\rightarrow$ RQ-A3. Across prompt-search backbones, *name* sampling without lagged feedback achieved the highest NCA$\times$AMI and the lowest variance; enabling lagged feedback lowered performance and increased variance (Figure 4.7).

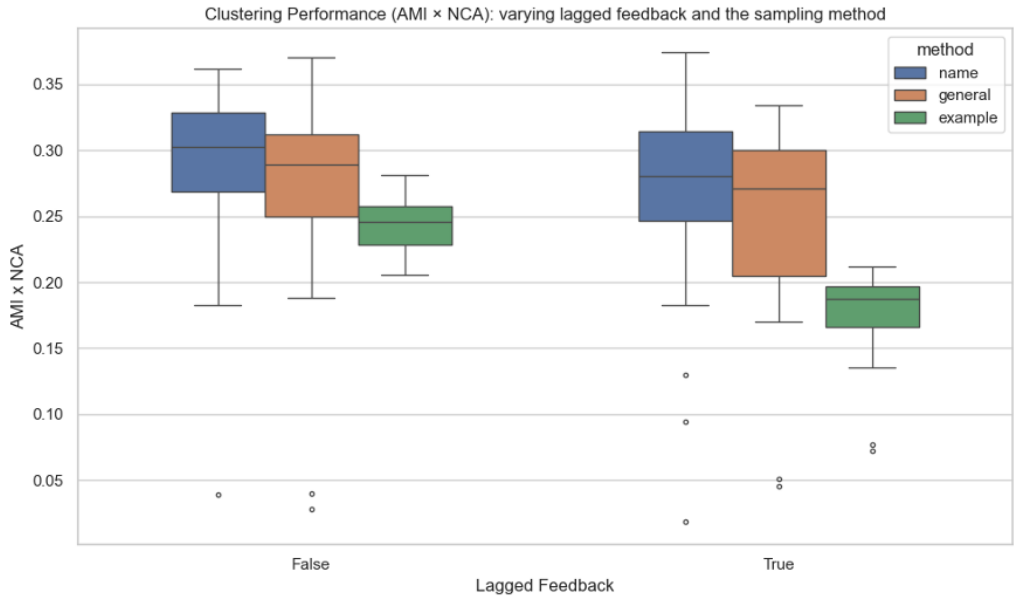

Figure 4.7: **Lagged feedback and sampling method (boxplots).** In GPFA, turning off lagged feedback improves NCA×AMI and reduces variance across seeds. Name-only sampling outperforms *general* and *example*, and no lagged feedback results in higher scores. (validation-set scores across multiple runs.)

Finding → RQ-A2. Within the GIFA/GPFA pipeline, choosing *name*-only sampling and turning off lagged feedback yielded higher-quality discovered intents (higher NCA×AMI), thus a more effective intent discovery.

We guided Sub-Experiment B by these results, fixing *name* sampling and turning off lagged feedback.

4.5.2 Phase 2 Sub-Experiment B: Crossover and Feedback Depth (SEE Variants Only)

Experimental setup: In this sub-experiment, we explored how additional evolutionary features affected performance, applying them exclusively to the SEE and Hyperband-SEE strategies. While we can compare the two, our primary goal in this phase was not cross-algorithm comparison but to analyze the relative effects of feedback and crossover within each algorithm. Since Hyperband explores more prompts under the same budget, we interpret differences between SEE and Hyperband-SEE in the context of their respective settings. With the best-performing sampling and feedback configuration fixed from sub-experiment A, we explored the following factors:

- The number of feedback rounds, $k_1 \in \{1, 2\}$

- The number of crossover rounds, $k_2 \in \{1, 2\}$, and the types of crossover rounds: error-wise or semantic-wise.

We evaluated each configuration by running the full SEE and Hyperband-SEE routines on the validation set. The goal was to examine the contribution of crossover and feedback depth to overall prompt quality and to determine whether the increased compute cost of crossover phases is justified by improved clustering performance.

Results: NCA×AMI was evaluated under *name* sampling with lagged feedback disabled (per 2A), investigating feedback depth k_1 and crossover rounds k_2 for SEE and Hyperband-SEE. We first report SEE (Figure 4.8), then Hyperband-SEE (Figure 4.9), and conclude with a seeded comparison (Table 4.1).

Figure 4.8 presents the performance of different SEE configurations across varying feedback depths ($k_1 \in [1, 2]$) and crossover strategies ($k_2 \in [0, 1, 2]$), using either error-based or semantic-based crossover types. The configuration with two feedback rounds and no crossover achieved the highest scores overall. Adding one or two crossover rounds, even with error-based strategies, did not lead to further gains and increased computational cost without consistent performance improvement. Configurations without any feedback phase performed worse, confirming the central role of feedback in guiding effective prompt optimization.

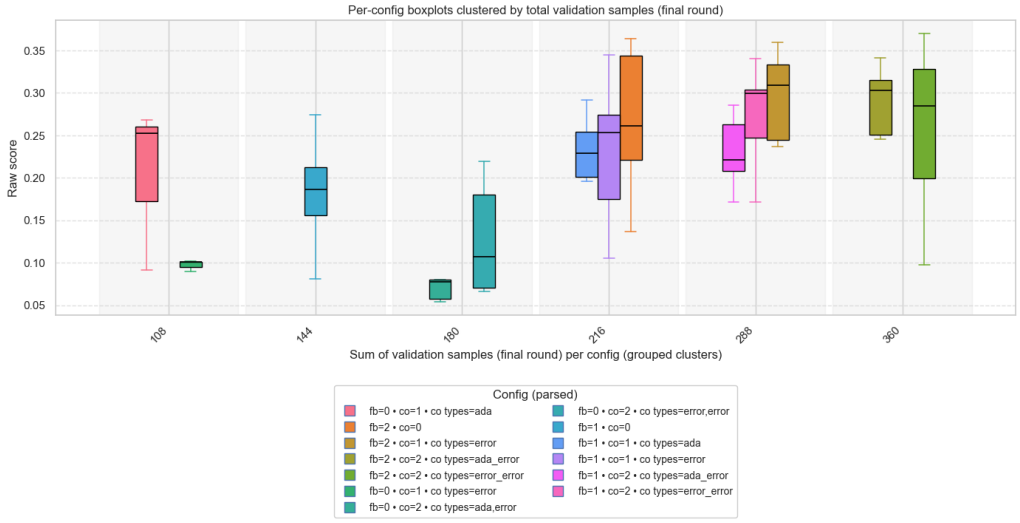

Figure 4.8: **SEE internals: feedback and crossover.** Best SEE configuration uses two feedback rounds and no crossover; error-based crossover is more stable than semantic crossover. Metric: NCA×AMI.

Among the crossover strategies, error-based crossover consistently outperformed semantic-based crossover, showing higher median scores and lower variance. Semantic crossover

introduced greater variability, likely due to its unsupervised and generative nature, while the error-based crossover directly exploits prompt-specific weaknesses, making it more stable and targeted. The two-feedback-only configuration outperformed all others while remaining resource-efficient, so we selected it as the final configuration for the SEE algorithm.

Figure 4.9 shows the corresponding Hyperband-SEE results under the same configuration space. Here, crossover became more effective when paired with feedback. Several configurations with one feedback round and one crossover round (either error or semantic) perform nearly as well as the two-feedback-round setup. This could be attributed to Hyperband's design. The algorithm incorporates more feedback-like adjustments by evaluating prompts across multiple rounds with increasing validation budget, amplifying the impact of even a single explicit feedback round.

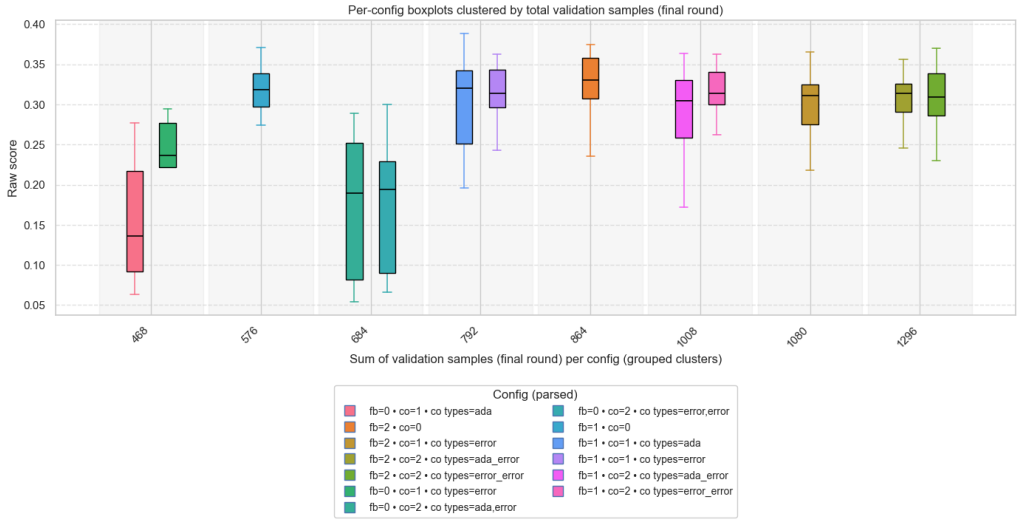

Figure 4.9: **Hyperband-SEE internals: feedback and crossover.** Under hyperband scheduling, one-feedback/one-crossover can be competitive, but two-feedback/no-crossover remains strongest at equal validation budgets. Metric: NCA$\times$AMI.

As in the standard SEE case, semantic-based crossover in Hyperband-SEE showed higher variance, while error-based crossover remained more stable and performed better.

We re-ran the four top-performing setups across five random seeds to determine the best configurations for the Hyperband-SEE algorithm. Table 4.1 shows that the configuration with two feedback rounds and no crossover achieved the highest mean score and lowest variance, significantly outperforming all others. Configurations with one feedback round and a single crossover (error or semantic) performed similarly to each other but did not match the two-feedback-only setup. We selected the two-feedback configuration without crossover as the optimal Hyperband-SEE variant based on these results.

k_1	k_2	**Crossover Type**	**NCA×AMI**
2	0	-	**0.365 ± 0.02***
1	1	Error	0.334 ± 0.03
1	1	Semantic	0.324 ± 0.03
1	0	-	0.330 ± 0.02

Table 4.1: **SEE/Hyperband-SEE (feedback and crossover).** Effect of feedback rounds (k_1) and crossover rounds (k_2) on clustering quality (NCA×AMI), comparing error-based vs. semantic (ADA-003) crossover; five seeds; best in bold; $^*p < 0.05$.

Finding → RQ-A3. SEE performed best with two feedback rounds and no crossover; Hyperband-SEE showed competitive one-feedback with one-crossover variants, but two-feedback with no-crossover remained strongest overall under equal validation budgets (Figures 4.8, 4.9; Table 4.1).

Finding → RQ-A2. Optimizing the GIFA prompt with two feedback rounds (no crossover) improved clustering quality (higher NCA×AMI), indicating more effective intent discovery in the GIFA/GPFA pipeline.

4.6 Phase 3: Final Backbone Search Algorithm Comparison

Experimental setup: In the third and final phase, we compared the best-performing settings for the prompt search strategies. Under fixed computational budgets and evaluation conditions, we quantified how well each method generalized to unseen data, using the optimal configurations identified in phase 2.

We allocated each algorithm a validation budget of 1,800 conversation evaluations to ensure a fair comparison. In our setup, this budget corresponded to roughly five hours of computation (in our setup). Each algorithm was run with five different random seeds to assess stability and variance in performance.

For each seed:

1. The prompt search algorithm was executed using the train and validation split.
2. The top-ranked prompt (based on NCA×AMI score on the validation set) was selected.

3. This prompt was evaluated using the full 125-conversation test set to obtain final clustering metrics.

We compared the different methods by scoring the final prompts they produced. For greedy search, random search, and SEE, we also examined the cumulative maximum scores over iterations. The comparison was limited to these three methods, since Hyperband-SEE uses fewer validation samples in early rounds, making its optimization process difficult to compare directly.

Results: Under the fixed validation budget, we first examined the cumulative best-of-iteration trajectories for greedy search, random search, and SEE (Figure 4.10). We then report held-out test scores for all methods (Table 4.2).

Figure 4.10 shows the cumulative maximum scores over prompt iterations for greedy search, random search, and SEE. Greedy and random search started with relatively high initial scores, beginning from a manually crafted human prompt. In contrast, SEE started from scratch and constructed prompts by itself. Despite this initial disadvantage, SEE converges to a higher final score than greedy and random search. Greedy search demonstrates early gains but converges quickly, suggesting that its exploitative behavior causes it to get trapped in local optima. Random search improves more gradually and consistently, benefiting from broader exploration. SEE, however, achieves larger cumulative gains throughout the search and outperforms both in final score and convergence behavior.

We extracted and averaged the final scores from all runs to compare them with the Hyperband-SEE method. Table 4.2 presents these results.

Configuration	**Score (NCA×AMI ± Std)**
Greedy	0.3385 ± 0.0093
Random	0.3429 ± 0.0208
SEE	0.3524 ± 0.0053
Hyperband-SEE	$\mathbf{0.4195 \pm 0.013}^{*}$

Table 4.2: **Prompt-search backbones under equal budget.** Test NCA×AMI (mean $\pm$ std) for greedy search, random search, SEE, and Hyperband-SEE search with identical validation budget (1,800 evaluations; five seeds); scores on the 125-conversations test set.; $^{*}p < 0.05$.

As shown in Table 4.2, Hyperband-SEE achieved the highest average test score compared to the others. Its early pruning mechanism concentrated resources on the most promising

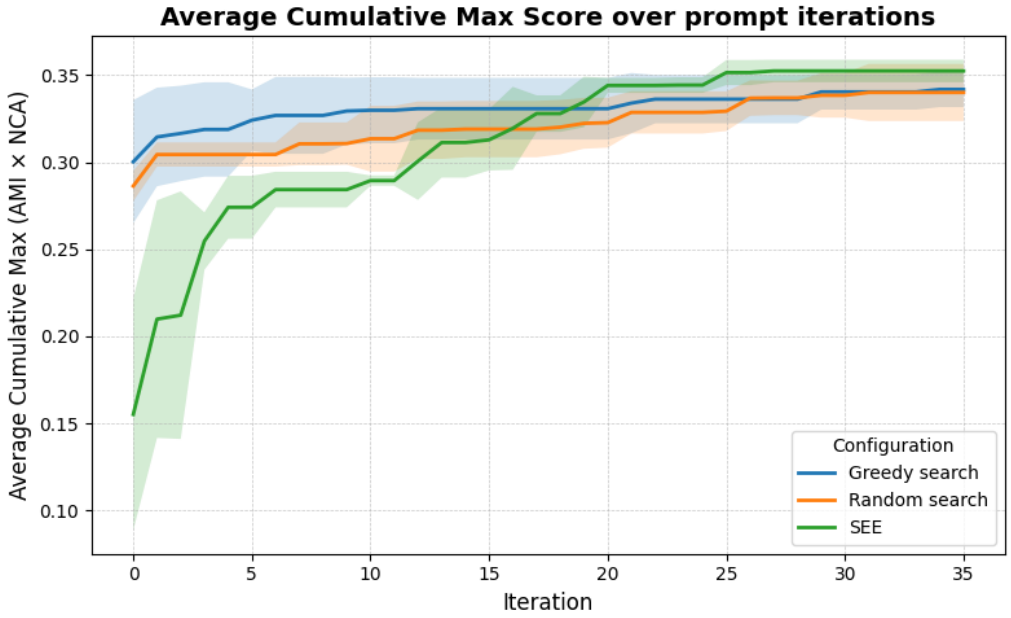

Figure 4.10: **Cumulative best NCA×AMI versus iterations.** Trajectories for greedy search, random search, and SEE under the same budget (1,800 validation evaluations; five random seeds): SEE starts without a human prompt yet surpasses greedy and random search; Hyperband-SEE was excluded here due to incomparable interim budgets.

prompts, leading to stronger overall performance. This approach, however, introduced slightly higher variance than SEE, likely due to its reliance on partial validation samples in early rounds, which could misestimate prompt quality. SEE performed nearly as well, offering relatively strong scores with lower variance. Greedy and random search scored lower on average, with random search exhibiting higher variability, possibly due to its stochastic nature.

Finding → RQ-A3. With equal validation budgets (1,800 evaluations; five random seeds), Hyperband-SEE attains the highest test NCA×AMI, outperforming SEE, greedy search, and random search. Despite starting without a human seed, SEE surpasses greedy and random search, while Hyperband's pruning delivers the best overall performance.

Finding → RQ-A2. The Hyperband-SEE optimized prompt increases GIFA intent discovery quality (higher NCA×AMI on the held-out test set).

We used the best Hyperband-SEE prompt found to generate intents across all 26,253 conversations for downstream fine-tuning.

4.7 Final Dataset Creation

Experimental setup: After determining the best GIFA and GPFA settings from the three-phased experiment plan, we selected the best Hyperband-SEE prompt and used it as input for GIFA to generate intents for all messages in the complete conversation dataset. We then analyzed the number of intents found across all 26,253 conversations and the distribution of their frequencies.

Results: Using the phase 3 top-ranked Hyperband-SEE prompt (Appendix B), which yielded an NCA×AMI score of 0.465, we applied GIFA to label the full dataset of 26,253 conversations. Below we report corpus-scale growth in unique intents and the resulting intent-frequency distribution (Figure 4.11).

Figure 4.11 illustrates the resulting distribution of intents. The number of unique intents initially increased rapidly but stabilized into a near-linear growth trend. This behavior is consistent with expectations in dynamic customer service environments, where novel customer queries and assistant responses continue to emerge over time, but at a diminishing rate. It also matches the intent growth and frequency distribution of the labeled dataset (Figure 4.2), suggesting that the GIFA algorithm correctly captured the underlying intent dynamics.

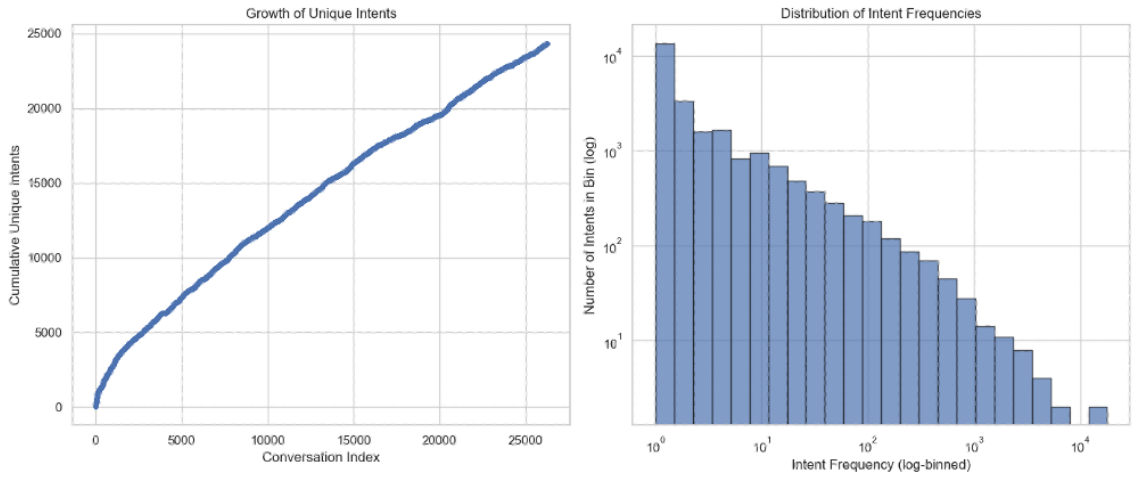

Figure 4.11: **High-granularity dataset intent growth and frequency.** Running the best Hyperband-SEE prompt with GIFA over 26,253 conversations yields 24,357 unique intents covering 367,157 messages; unique-intent growth starts steep, then linearizes. The frequency distribution is long-tailed, as shown in the histogram with bin edges spaced evenly on a $\log_{10}$ scale between the minimum and maximum observed intent frequencies (25 bins total).

The intent frequency distribution exhibits a long-tail pattern: few intents occur frequently, while the majority have only a few mapped messages. This reflects the intended granularity of the labeling scheme, where many intents are domain-specific and narrowly defined, while a few (e.g., greetings or acknowledgments) are more general. We identified 24,357 unique intents across 26,253 conversations, covering 367,157 messages.

Finding → RQ-A2. At full scale, the GIFA/GPFA pipeline yields stable, high-reuse, long-tail intent annotations (24,357 intents over 367,157 messages).

4.8 GIFA and GPFA Research Questions: Summary

RQ-A2: How effectively can intent labels be discovered using the proposed GIFA/GPFA framework?

Retrieval improves intent reuse versus no retrieval (phase 1: Figure 4.3). Best retrieval settings depend on the sampling method: *example*: semantic k=1, n=2; *general*: semantic k=1; *name*: semantic k=2 (Figure 4.4-4.6). Within GPFA, *name*-only sampling outperforms *general* and *example*, and disabling lagged feedback yields higher scores and lower variance (phase 2A; Figure 4.7). At full scale, GIFA and GPFA produced 24,357 unique intents across 367,157 messages with a long-tail frequency distribution, matching the labeled dataset. (Figure.4.11).

RQ-A3: Does Hyperband-SEE improve prompt-search efficiency and final labeling quality over SEE and simple baselines?

Under an equal validation budget (1,800 evaluations; five random seeds), Hyperband-SEE achieves the highest test NCA×AMI, outperforming SEE, greedy search, and random search (phase 3; Table 4.2). Within SEE/Hyperband-SEE, the best internal configuration is two feedback rounds, no crossover; error-based crossover is more stable than semantic, and lagged feedback degrades performance (phase 2B; Figures 4.8, 4.9; Table 4.1).

5 Experiments regarding Fine-Tuning

In this section, we evaluate modeling choices for stage 2 (sub-stages D-E) and how they interact with the intent labels produced in stage 1. We begin by revisiting sub-stage A: before comparing modeling variants, we quantify the effect of LLM-based pre-processing (anonymization, cleaning/merging, and filtering incomplete conversations) via downstream fine-tuning performance (subsection 5.4). This establishes the data preparation we carry forward into the modeling phases. All experiments were done using a chronological train/validation/test split to assess generalization under domain drift, share a fixed QLoRA and LLM hyperparameter fine-tuning setup, and we evaluated fine-tuned models using automatic metrics (BLEU, ROUGE-L, METEOR, BERTScore). The final experiment used LLM-as-a-judge evaluation (G-Eval) on helpfulness, semantic similarity, and professionalism.

The stage 2 experiments are split into four phases that build on one another. In phases 1-3, we focus on open-weight models: in phase 1, we compare intent-guided variants (CoT, 2-step, mixed) against a response-only baseline for next-reply generation (subsection 5.5); in phase 2, we investigate retrieval strategies by comparing RAG with RAFT applied to the strongest phase 1 setups (subsection 5.6); and in phase 3, we examine model size by scaling from 1.7B to 4B parameters (subsection 5.7). In phase 4, we provide the final comparison with a closed-weight LLM setup, where we evaluate the best-performing intent-guided and response-only models against each other and against a closed-weight GPT-4o RAG co-assistant baseline (subsection 5.9). Figure 5.1 summarizes the stage 2 experimental flow and shows where each research question is addressed.

RQ coverage. This section answers **RQ-A1** (impact of LLM-based pre-processing), **RQ-B1** (intent-guided versus response-only fine-tuning), **RQ-B2** (RAFT versus RAG), **RQ-B3** (model size effects), and **RQ-B4** (comparison with GPT-4o RAG).

5.1 Fine-Tuning Evaluation Data Setup

We used time-based data splitting to evaluate the performance of the fine-tuned models. Our goal is to develop a model that remains effective as new customer questions and business situations emerge. Frequent retraining is undesirable whenever business logic changes. For example, customers may begin referring to new promotions (e.g., scratch tickets, digital vouchers) or new product features.

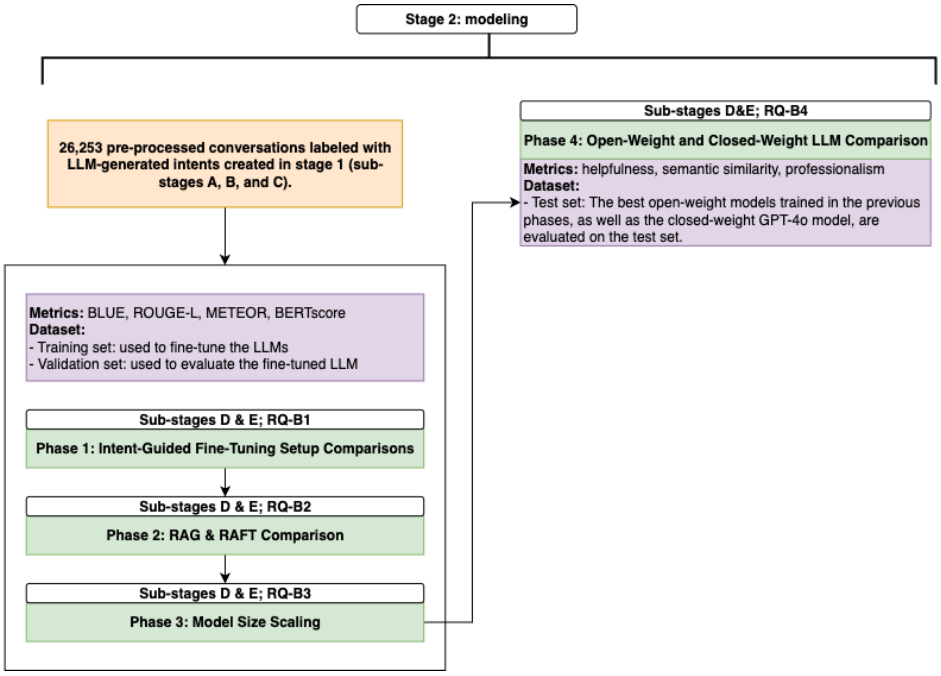

Figure 5.1: **Stage 2: modeling experiments.** Sequential setup for evaluating fine-tuning and retrieval strategies on the intent-labeled dataset from Stage 1. 26,253 pre-processed conversations labeled with GIFA+GPFA intents form the basis. Training and validation sets are used to fine-tune open-weight LLMs and assess intermediate results with automatic metrics (BLEU, ROUGE-L, METEOR, BERTScore). Phase 1 (RQ-B1) compares intent-guided fine-tuning variants against a response-only baseline. Phase 2 (RQ-B2) compares RAG with RAFT. Phase 3 (RQ-B3) studies model-size scaling from 1.7B to 4B parameters. Finally, phase 4 (RQ-B4) compares the best-performing open-weight models from earlier phases against the closed-weight GPT-4o RAG co-assistant using LLM-as-a-judge (G-Eval) on helpfulness, semantic similarity, and professionalism.

To simulate temporal changes and reward models that generalize to unseen queries, we performed a chronological data split. Training was carried out on older conversations, while evaluation targeted more recent interactions containing new or emerging topics. This setup allows us to assess how well the models handle domain drift. The dataset was divided into three splits:

1. Training set (80%, 93,524 samples): We used the earliest conversations in the dataset to fine-tune the model.
2. Validation set (10%, 11,697 samples): We selected this subset from the subsequent period to identify the best-performing model configuration.
3. Test set (10%, 11,697 samples): We held out the most recent portion of the dataset for final evaluation and to estimate real-world generalization.

5.2 Model Training Configuration

Qwen3-1.7B was used for the fine-tuning experiments and applied the QLoRA approach. To ensure comparability across experimental conditions, we applied the same hyperparameters consistently to all model variants:

- **Learning rate**: $2e{-4}$
- **LoRA rank** (r): 16
- **LoRA alpha**: 32
- **Maximum sequence length**: 2048
- **Batch size**: 32
- **Warmup ratio**: 0.05
- **Learning rate scheduler**: Cosine decay
- **Early stopping**: Training was halted if the validation loss improved by more than 0.005 over 300 steps.

In all fine-tuning experiments, we selected hyperparameters based on the recommended configurations from the QLoRA paper [43] and the settings suggested in the Hugging Face PEFT and Transformers libraries, rather than tuning them individually.

All models were trained on the time-split training set to simulate generalization to newly emerging customer issues.

5.3 Evaluation Metrics

Lexical and semantic metrics were combined to assess the quality of generated assistant responses across different fine-tuning setups. Lexical metrics measure word overlap between the model's and ground-truth responses, while semantic metrics evaluate meaning similarity.

5.3.1 Lexical Similarity

Lexical metrics yield a high score whenever a response closely matches the words and phrases from the ground-truth responses. While lexical scoring does not fully capture semantic similarity, these metrics can help measure how well the model replicates reference responses. We chose the following three lexical metrics to assess the models:

BLEU (Bilingual Evaluation Understudy): BLEU computes the proportion of n-gram matches between the generated text and one or more reference texts. BLEU scores range from 0 to 1, with higher values indicating greater overlap. BLEU rewards exact n-gram matches, especially for shorter phrases and word sequences. A limitation of BLEU is that it may penalize valid paraphrases or synonyms.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation): ROUGE-L measures the longest common subsequence between the generated response and the reference. ROUGE emphasizes recall, rewarding the models for covering key parts of the ground-truth responses regardless of exact word order.

METEOR (Metric for Evaluation of Translation with Explicit Ordering): METEOR improves upon BLEU using stemming, synonym matching, and word order penalties. METEOR accounts for semantically similar words, making it more robust to paraphrasing.

5.3.2 Semantic Similarity

In addition to the lexical metrics, we used semantic similarity metrics to assess whether the meaning of the generated assistant responses aligns with that of the ground-truth references. Compared to lexical metrics, semantic metrics capture paraphrasing and broader variations in phrasing.

We used BERTScore during the fine-tuning experiments. BERTScore computes similarity based on contextual word embeddings from a large pre-trained transformer model [56]. Specifically, we applied the Dutch-language setting of the XLM-RoBERTa-Large model.

BERTScore compares each token in the generated response to those in the reference response using contextual embeddings and calculates a similarity score based on cosine similarity. The result reflects how semantically close the generated output is to the reference, even when exact word choices differ. This metric provides a more accurate evaluation model for performance in realistic scenarios, where multiple valid versions of a correct response can exist.

5.4 Effect of LLM-Based Pre-Processing on Fine-Tuning

Experimental setup: As introduced in the methodology, we applied three LLM-based pre-processing steps to enhance the quality of training data before fine-tuning:

- **Anonymization** was used to prevent the model from learning or memorizing personal information.
- **Message correction and merging** combined or concatenated related messages, deleted redundant messages, and corrected grammar or spelling issues.
- **Conversation filtering** removed conversations that ended without resolving the customer's issue.

Rather than constructing separate labeled datasets to evaluate each pre-processing step, the impact was assessed indirectly by comparing the performance of models fine-tuned on differently pre-processed datasets.

Note that although we present this subsection after the GIFA and GPFA experiments for clarity, we conducted the pre-processing evaluation beforehand. This allowed us to select the best-performing data processing setup for the GIFA and GPFA evaluations and for all subsequent fine-tuning experiments.

The anonymization step was excluded from this comparison. In our co-assistant deployment setting, human assistants review all assistant responses before sending them, eliminating the risk of exposing sensitive information.

We compared the following three fine-tuned models:

- **Raw model:** Fine-tuned on unprocessed data (no correction, merging, or filtering).
- **Corrected model:** Fine-tuned on conversations where messages were cleaned, merged, or deleted for clarity.
- **Filtered model:** Fine-tuned only on successfully resolved conversations.

If both the corrected and filtered models outperform the raw models individually, we also evaluate a combined model that applies both correction/merging and filtering steps. Otherwise, the combined evaluation was skipped.

Results: We fine-tuned identical QLoRA configurations on three datasets: raw (26,253 conversations), corrected (26,253), and filtered (22,076), and report BLEU, ROUGE-L, METEOR, and BERTScore on the held-out validation split (Table 5.1).

Table 5.1 suggests that all metrics improved when using the corrected model, which was expected. However, the filtered model performed less well than the raw and corrected models. A possible explanation for these results could be linked to the training setup: the whole conversation history is given, and the following assistant responses must be generated. The model is not necessarily trained to solve the entire problem, but to generate the following responses correctly. When deleting many unfinished conversations, it can be the case that in early messages, the model could have learned from them. Ultimately, either an assistant or a customer stopped responding, and the message was flagged as unsolved, which threw away valuable training data.

Model	BLEU	ROUGE-L	METEOR	BERTScore
1.7B Raw Response	9.010	0.240	0.281	0.890
1.7B Corrected Response	**10.220**	**0.255**	**0.299**	**0.899**
1.7B Filtered Response	8.815	0.239	0.287	0.888

Table 5.1: **Preprocessing variants (validation).** Qwen3-1.7B (QLoRA) fine-tuned on raw versus corrected versus filtered datasets; BLEU, ROUGE-L, METEOR, and BERTScore.

Because dropping the unfinished conversations lowered performance, while cleaning the messages improved performance over the raw conversations, we did not test a combined setup. We selected the corrected messages as the basis for GIFA, GPFA, and all further fine-tuning experiments.

Finding → RQ-A1. LLM-based message correction/merging improved downstream generation across all automatic lexical and semantic metrics, while dropping unresolved conversations reduced performance; therefore, we used the corrected dataset (without filtering) for the GIFA/GPFA experiments and subsequent fine-tuning.

5.5 Phase 1: Intent-Guided Fine-Tuning Setup Comparisons

Experimental setup: In the first phase of the fine-tuning experiments, we evaluated the impact of different setups to determine which strategy produced the most accurate and semantically correct assistant responses. We tested the following fine-tuning strategies:

- **Baseline (Zero-Shot):** The unmodified Qwen3-1.7B model was evaluated on the validation set without additional training.
- **Baseline Responses Only:** Given the conversation history, the model was fine-tuned to predict the following assistant message. This represents the most straightforward supervised approach.
- **CoT (Intent Chain + Response Chain):** A CoT inspired variant in which the model was trained to first output a structured list of assistant intents in a thinking block, followed by the actual assistant responses.
- **2-step (Intent → Response):** A sequential training approach where a first adapter was trained to do intent prediction, and then a second adapter was trained to output assistant responses.
- **2-step (Intent → CoT):** This was like the above, but the second adapter was trained on both the intent chain and the assistant response using the CoT format.
- **Mixed Ratio (Intent & CoT):** A multi-task model trained on a mixed dataset consisting of 30% intent-only prediction samples and 70% CoT-style intent-response samples.

We trained and evaluated each variant on the same training and validation dataset, assessing performance with lexical and semantic metrics. In the intent-guided CoT setups, the model produced both intents and responses, but only the responses were considered during evaluation. These results determined which fine-tuning setup generated the best responses.

Results: All setups underwent training and evaluation under identical data splits and QLoRA hyperparameters. We reported BLEU, ROUGE-L, METEOR, and BERTScore on the held-out validation set (Table 5.2).

Table 5.2 shows that the zero-shot baseline, the model without any fine-tuning, performed worst across all metrics, indicating that task-specific fine-tuning was necessary for generating higher-quality assistant responses.

The response-only fine-tuned model, which learned to predict the assistant's following utterances from conversation history without any intent guidance, improved performance compared to the others. It performed on par with the CoT fine-tuned model. This suggests explicit intent structuring did not improve generation quality when used in a CoT-style single-phase training setup.

Model	BLEU	ROUGE-L	METEOR	BERTScore
1.7B Base	8.620	0.238	0.276	0.892
1.7B Response	**10.220**	**0.255**	0.299	**0.899**
1.7B CoT	10.205	**0.255**	**0.302**	0.898
1.7B 2-step Intent → Response	9.817	0.253	0.293	0.895
1.7B 2-step Intent → CoT	9.417	0.250	0.292	0.893
1.7B Mixed Intent & Response	9.255	0.243	0.285	0.892

Table 5.2: **Fine-tuning setups (validation).** Qwen3-1.7B (QLoRA) comparison of zero-shot, response-only, CoT, 2-step (Intent+Response / Intent+CoT), and mixed; BLEU, ROUGE-L, METEOR, and BERTScore.

2-step training approaches performed worse than single-step strategies. This indicates that explicitly separating the planning and generation phases may hurt overall response quality. One explanation might be that the model may overfit during intent prediction, and subsequent training fails to fully integrate this intermediate knowledge into fluent generation.

The mixed-ratio model showed the lowest performance of all fine-tuned setups. A likely reason is that this strategy applies a single LoRA adapter across mixed tasks, preventing the model from cleanly learning task-specific behavior. In contrast, 2-step setups benefit from distinct adapters per phase, allowing later training to potentially compensate for earlier limitations, an option not available in the mixed-task configuration.

Finding → RQ-B1. Intent-guided fine-tuning did not outperform direct response fine-tuning: response-only and CoT were strongest and almost tied by automatic metrics, while 2-step and mixed underperformed; we selected response-only and CoT for phase 2.

5.6 Phase 2: RAG and RAFT comparison

Experimental setup: Building on the best-performing fine-tuning configurations from phase 1, we investigated the impact of RAG during inference and RAFT during training in phase 2. We aimed to assess whether adding external knowledge improved the generated responses.

The following new configurations were evaluated:

- Baseline model with RAG applied at inference time.

- RAFT model trained using the response-only setup.
- RAFT model trained using the best intent-guided configuration from phase 1.

Each model variant was assessed with the same lexical and semantic similarity metrics as in the previous phase. The comparison focused on the contribution of RAG and RAFT, examining whether fine-tuning with retrieved context led to more accurate responses.

Results: In the previous phase, response-only fine-tuning and intent-guided CoT fine-tuning yielded comparable performance across all metrics. In phase 2, we built on these configurations to assess the impact of RAG and RAFT.

As shown in Table 5.3, adding RAG at inference time improved upon the base zero-shot model, confirming that access to external information during generation improved response quality. This performance gain was consistent when RAG was combined with the fine-tuned models, suggesting that retrieval complements fine-tuning for both responses and CoT fine-tuning.

Model	BLEU	ROUGE-L	METEOR	BERTScore
1.7B Base	8.620	0.238	0.276	0.892
1.7B Base RAG	9.620	0.245	0.280	0.894
1.7B Response-only	10.220	0.255	0.299	0.899
1.7B Response-only RAFT	**13.215**	**0.276**	0.330	0.901
1.7B CoT	10.205	0.255	0.302	0.898
1.7B CoT RAFT	13.205	0.275	**0.332**	**0.902**

Table 5.3: **RAG vs. RAFT (validation).** Retrieval-augmented configurations at 1.7B: Base/Response/CoT with and without RAFT; BLEU, ROUGE-L, METEOR, and BERTScore.

Moreover, both RAFT variants (response-only and CoT) outperformed the RAG-only model. This indicates that integrating retrieved context directly during the training phase is more effective than relying on it solely at inference time. However, no performance difference was observed between the response-only RAFT and CoT-based RAFT models. This aligned with the findings from phase 1, suggesting that CoT did not outperform the response-only model even with access to more information.

Finding → RQ-B2. RAFT outperformed RAG: both response-only RAFT and CoT RAFT exceeded base RAG and their non-RAFT fine-tuned baselines across automatic lexical and semantic metrics (Table 5.3).

Finding → RQ-B1. Intent-guided CoT did not beat response-only under retrieval: The reported automatic metrics made CoT RAFT and response-only RAFT indistinguishable.

5.7 Phase 3: Model Size Scaling

Experimental setup: The third phase explored the effect of increasing model size on generation quality. We compared the performance of the Qwen3-4B model with that of the smaller 1.7B model to assess whether larger LLMs improved generalization in standard and intent-guided fine-tuning setups.

The following 4B model configurations were evaluated:

- Baseline (4B) model with RAG
- RAFT (response-only) with 4B model
- RAFT model using the best intent-based setup

For the larger model size, the learning rate was reduced from $2e{-}4$ to $1e{-}5$ to ensure stable optimization.

Results: Building on the phase 2 findings, which showed the advantages of RAG and RAFT, we investigated whether scaling the model size further improved generation performance in this phase.

Table 5.4 shows that increasing the model size from 1.7B to 4B leads to consistent performance improvements across all configurations. The 4B baseline model with RAG outperformed the smaller 1.7B model, confirming that larger model capacity improves the ability to utilize retrieved context effectively.

Additionally, both RAFT variants, response-only and CoT-based, benefited from the increased capacity, outperforming the 4B baseline with RAG. This reinforces the earlier conclusion that RAFT is more effective than sole RAG, regardless of model size.

However, like the previous phases, no significant difference was observed between the response-only and CoT-based RAFT setups. This suggests that intent-guided CoT fine-tuning does not offer significant advantages over direct response-only fine-tuning, even with the larger model's enhanced generalization capabilities.

Finding → RQ-B3. Increasing model size from 1.7B to 4B improved all automatic metrics across base RAG, response-only RAFT, and CoT RAFT (Table 5.4).

Model	BLEU	ROUGE-L	METEOR	BERTScore
1.7B Base RAG	9.620	0.245	0.280	0.894
4B Base RAG	10.820	0.255	0.288	0.895
1.7B Response-only RAFT	10.220	0.255	0.299	0.899
4B Response-only RAFT	11.823	**0.271**	**0.315**	**0.901**
1.7B CoT RAFT	10.205	0.255	0.302	0.898
4B CoT RAFT	**11.905**	0.267	0.312	**0.901**

Table 5.4: **Parameter scaling (validation).** 1.7B vs. 4B under RAG/RAFT; BLEU, ROUGE-L, METEOR, and BERTScore.

Finding → RQ-B2. For a fixed model size (1.7B or 4B), RAFT outperformed RAG on all metrics (Table 5.4).

Finding → RQ-B1. Within RAFT, intent-guided CoT did not consistently outperform direct response-only. (Table 5.4).

5.8 LLM-as-a-judge Evaluation (G-Eval)

While lexical and semantic similarity metrics provide valuable insights into the overlap and meaning alignment between generated responses and ground-truth answers, they do not fully capture the quality, clarity, and domain appropriateness of generated outputs, particularly in real-world settings. To complement the earlier evaluation stages, we used an LLM-as-a-judge approach, where LLMs assessed the generated responses.

Specifically, we used the G-Eval framework [57] to score three key dimensions of response quality using rubric-based judgments. We computed the following metrics:

- **Helpfulness:** Measures if the response helps the customer progress towards their goal. The generated response is not compared to the ground-truth; it only assesses whether the generated response is helpful given the customer's messages.
- **Semantic Similarity:** Captures whether the intended meaning of the generated response aligns with the ground-truth. This metric is comparable to the BERTScore used in the previous phases.
- **Professionalism:** Measures the tone and appropriateness of the generated response compared to the ground-truth response.

Following a consistent rubric design, these metrics were scored on a 0-1 scale using structured evaluation prompts provided to the LLM. The G-Eval prompts used are in Appendix D.

5.9 Phase 4: Open-Weight and Closed-Weight LLM Comparison

Experimental setup: In the final phase of the fine-tuning experiments, we compared the most promising open-weight model configurations from the earlier phases to determine which setup performed best under real-world conditions. We used the G-Eval framework to evaluate generated responses on helpfulness, semantic similarity, and professionalism.

This phase assessed which configuration produced the most consistent responses in practice and compared the developed models against a closed-weight baseline similar to Albert Heijn's deployed system, based on GPT-4o combined with RAG.

The following models were included in the final comparison:

- Closed-weight GPT-4o RAG setup
- Best open-weight Qwen3 response-only fine-tuned model
- Best open-weight Qwen3 intent-guided fine-tuned model
- Ground-truth assistant responses (for helpfulness)

Results: We evaluated all models with G-Eval on three human-aligned criteria: helpfulness, semantic similarity, and professionalism, using the shared test set. The resulting scores are reported in Table 5.5.

Model	Helpfulness	Semantic Similarity	Professionalism
GPT-4o (RAG baseline)	**0.820***	0.229	0.452
4B CoT RAFT	0.304	0.275	**0.468**
4B Response-only RAFT	0.293	**0.292**	0.452
Ground-truth Responses	0.285	–	–

Table 5.5: **LLM-as-a-judge (test).** G-Eval comparison of GPT-4o RAG baseline versus 4B RAFT (response-only/CoT) and Ground-truth; helpfulness, semantic similarity, and professionalism; $^*p < 0.05$

The GPT-4o baseline remained strongest on helpfulness, showing its ability to generate functional responses across various conversations. However, the table suggests that every

other model had a higher helpfulness score than human responses, which is counterintuitive. After looking at the feedback created by G-Eval, the main reason for giving low scores was usually due to asking back questions and saying that it does not know the answer to a specific question, which in practice is desirable behavior; however, when purely looking at whether the response helped the customer, this was not the case.

Both fine-tuned Qwen3 models outperformed GPT-4o on semantic similarity, suggesting that they produce responses more closely aligned with the ground-truth response. Furthermore, the 4B response-only RAFT model achieved the highest semantic similarity score, outperforming CoT. All models performed similarly regarding professional tone, with a slight edge for the 4B CoT RAFT model. This suggests that fine-tuning did not compromise the tone or politeness of generated responses.

To better understand how the models behave across specific conversational scenarios, we compared the GPT-4o setup with the fine-tuned Qwen3 response-only model at the intent level. This analysis focused on G-Eval semantic similarity.

Figure 5.2 and Figure 5.3 present a radar chart showing the top 20 intents with the most significant performance differences between the two models. The radar plot suggests that the response-only model excelled at high-frequency and relatively easy conversational tasks, such as `greet_customer`, `inquire_customer_needs`, and `offer_to_investigate_issue`. These are usually short and simple interactions that happen quite often. The response-only model has seen many examples in the data and probably remembers these standard formats.

In contrast, GPT-4o performed better on low-frequency, more complex intents requiring domain knowledge or reasoning. These included `inform_about_cancellation_fees`, `confirm_internal_escalation`, and `response_method`. These responses are relatively harder than greeting a customer.

Interestingly, GPT-4o did underperform in verification and clarification responses, which are critical to solving customer problems. These include intents such as `request_email_for_verification`, `request_verification_information`, and `request_time_to_review_messages`. That suggests that the fine-tuned response-only model was better at identifying when such information was necessary.

Finding → RQ-B4. Compared to the closed-source GPT-4o RAG baseline, the open-weight Qwen3-4B response-only RAFT scored higher on semantic similarity, and Qwen3-4B CoT RAFT scored slightly higher on professionalism, while GPT-4o remained strongest on helpfulness.

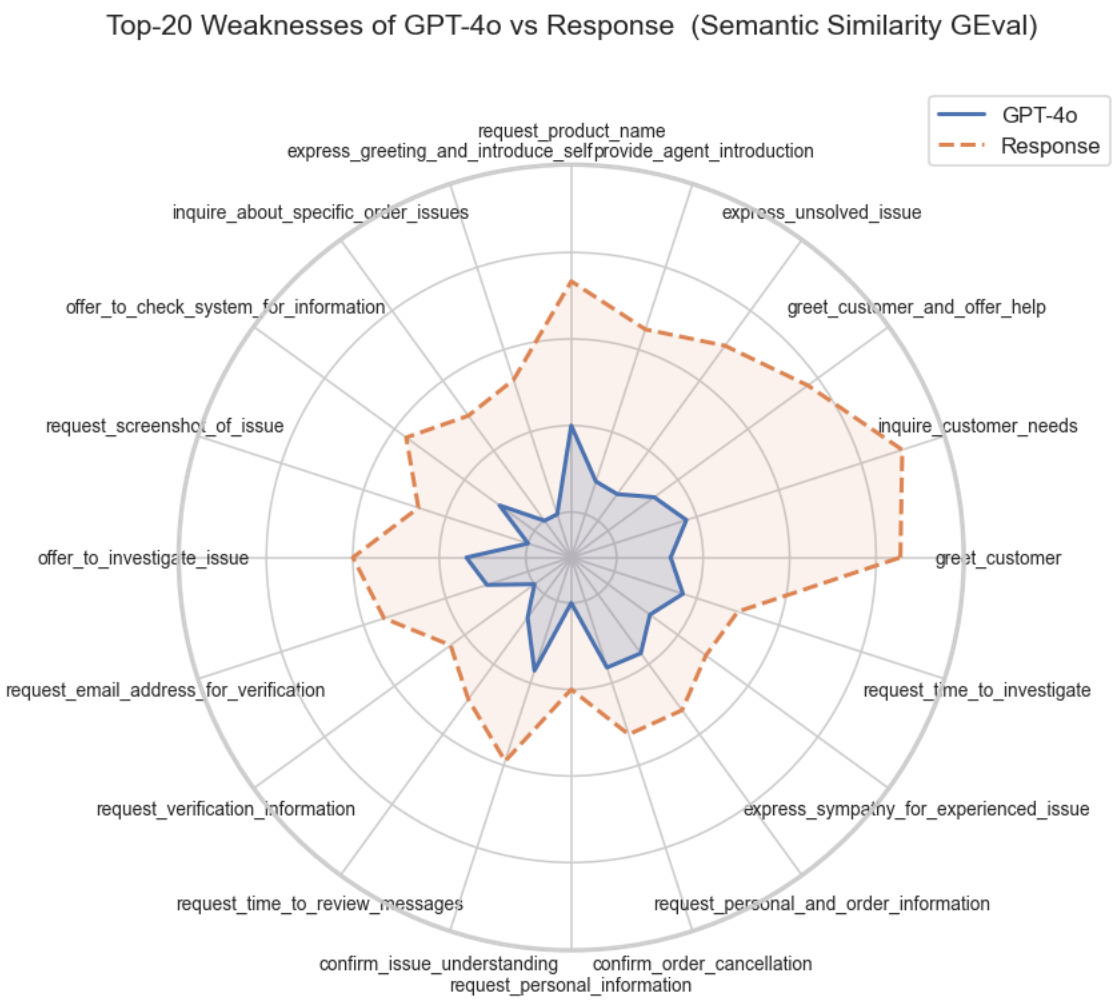

Figure 5.2: **GPT-4o weaknesses (semantic similarity, G-Eval).** Top-20 intents where Qwen3-4B response-only RAFT scores higher than GPT-4o RAG on G-Eval semantic similarity, often verification/clarification, and common operational intents.

Finding → RQ-B1. Between intent-guided and response-only fine-tuning (both RAFT, 4B), response-only RAFT yielded higher semantic similarity, whereas CoT RAFT was marginally more professional; helpfulness was similar and below GPT-4o.

5.10 Fine-Tuning Research Questions: Summary

RQ-A1: How do pre-processing choices: anonymization, cleaning, and the treatment of incomplete conversations, affect downstream label quality and fine-tuning performance?

Message correction/merging improved all reported metrics versus raw data, while filtering unresolved conversations harmed performance. (Table 5.1).

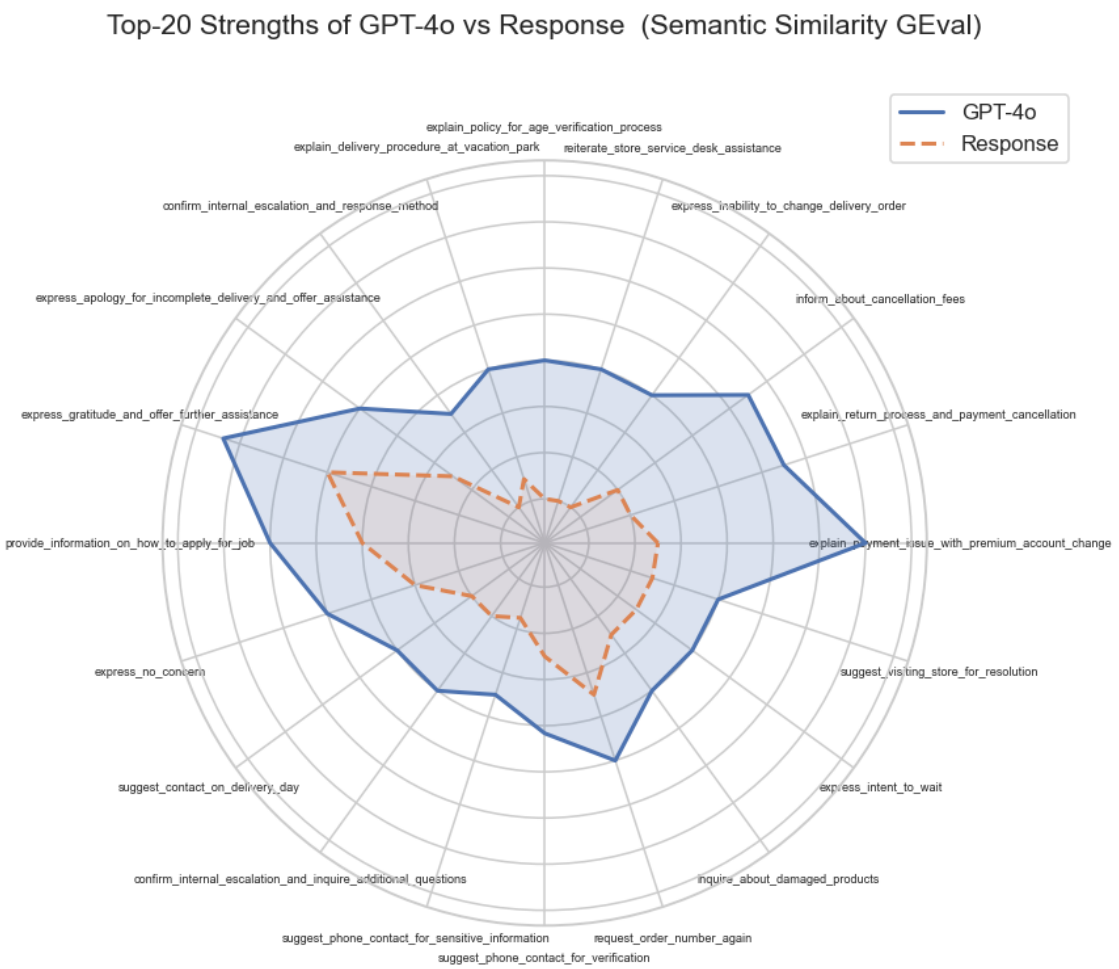

Figure 5.3: **GPT-4o strengths (semantic similarity, G-Eval).** Top-20 intents where GPT-4o RAG outperforms the 4B response-only RAFT model, typically rarer, knowledge-heavy, or policy-specific intents (e.g., cancellation fees, escalations).

RQ-B1: Do intent-guided variants (CoT, two-step, mixed) outperform response-only fine-tuning?

Intent-guided approaches did not outperform response-only: response-only and CoT were tied by automatic metrics, while 2-step and mixed underperformed (Table 5.2). Under retrieval, Response RAFT and CoT RAFT remained similar (Table 5.3); with 4B models, this pattern did not change (Table 5.4). In G-Eval, response-only RAFT scored higher on semantic similarity, and CoT RAFT scored slightly higher on Professionalism (Table 5.5).

RQ-B2: Does RAFT outperform RAG?

Both response-only RAFT and CoT RAFT exceeded base RAG and their non-RAFT baselines, and this was also true at both 1.7B and 4B scales (Table 5.3, 5.4).

RQ-B3: How does model size impact the performance of these intent-guided and retrieval-augmented training strategies?

Scaling from 1.7B to 4B improved all reported metrics across base RAG, response-only RAFT, and CoT RAFT (Table 5.4).

RQ-B4: How do the best open-weight configurations compare with the closed-weight GPT-4o RAG co-assistant across evaluation dimensions (semantic alignment, professionalism, helpfulness)?

GPT-4o RAG was strongest on helpfulness. Qwen3-4B response-only RAFT was highest on semantic similarity, and Qwen3-4B CoT RAFT was slightly higher on professionalism (Table 5.5). Furthermore, the response-only model performed better on common intents, while the GPT-4o setup performed better on rare intents.

6 Limitations and Future Work

We used 26,253 conversations in our work. In future work, we could incorporate additional conversations, allowing us to fine-tune on a broader dataset, measure whether improvements come from more data or better coverage, and better assess significant differences between model scores.

Hyperparameters for the fine-tuned models were set to sensible defaults, as complete hyperparameter optimization was out of scope. Given the known sensitivity to learning rate, warm-up, adequate batch size, sequence length, and LoRA rank/α, our approach likely limited the performance that could be achieved. A compact, budget-aware hyperparameter optimization method could be beneficial.

Model choice and capacity were constrained to the Qwen3 family. Scaling from 1.7B to 4B improved both base and fine-tuned performance, but a larger closed-weight model (GPT-4o) performed better on rarer intents, likely due to scale and pre-training breadth. The next step is a compute-matched comparison against an alternative open-weight family (e.g., Llama-3) at similar sizes, evaluating base and fine-tuned variants under the same data and hyperparameter optimization budget. Moreover, larger open-weight models could be fine-tuned and evaluated where resources allow.

For intent discovery, we explored GIFA/GPFA settings and introduced Hyperband-SEE. Although Hyperband-SEE outperformed simpler prompt search strategies, it still achieved modest NCA$\times$AMI scores; the final prompt used to label the complete high-granularity dataset reached an NCA$\times$AMI score of 0.465. Closely related intents were sometimes created instead of reused. A possible solution here would be to add an iterative merge step so that near-duplicate intents are merged rather than split.

The scalar product NCA$\times$AMI was used to guide prompt optimization, which collapses two objectives into one. Although convenient for guiding optimization, this choice oversimplifies the trade-off. A more comprehensive approach would be to analyze the Pareto front across the two metrics, providing insight into the trade-offs between prompts.

Using intents to guide fine-tuning did not outperform straightforward response-only fine-tuning. Instead of utilizing intents to guide fine-tuning, intents could also be used as an evaluation method. Intents could be treated as groupings that expose weaknesses, then update prompts or retrieval rules for those clusters; use them for routing (e.g., choose a fine-tuned open-weight model for specific intents and the GPT-4o setup otherwise);

and trigger targeted retrieval when an intent strongly predicts the needed knowledge (for example, automatically fetching delivery information for an `ask_about_delivery` intent).

Because we generated intent labels with LLMs (via GIFA/GPFA), the dataset may embed prior knowledge or biases of the labeling model. Fine-tuned open-weight models trained on these labels could therefore partly inherit this influence, limiting the extent to which results demonstrate independent generalization. This is particularly relevant when comparing against closed-weight systems such as GPT-4o, which also shaped the labeling process.

Finally, we relied heavily on LLM-as-a-judge (G-Eval) and automatic similarity metrics for evaluation. While useful for large-scale comparison, LLM-based evaluation is not ground truth and therefore represents a limitation in our work. Indeed, even human reference responses sometimes scored unexpectedly on the helpfulness metric. To align metrics more closely with real-world performance, G-Eval should be complemented with human preference evaluation and online A/B tests using outcome-based metrics (e.g., which model's response was actually used by human assistants).

7 Conclusion

Modern customer-service automation often relies on closed-weight LLMs (e.g., GPT-4o), which deliver quality but at the cost of transparency and customization. This work investigated whether historical conversations, labeled with LLM-generated intents, can fine-tune open-weight models to compete with a closed-weight GPT-4o RAG setup. We designed a two-stage pipeline with five sub-stages (A-E). Stage 1 (A-C) covered LLM-based pre-processing (A), intent generation with GIFA (B), and prompt optimization with GPFA and Hyperband-SEE (C). Stage 2 (D-E) evaluated fine-tuning strategies (D) and retrieval integration (E).

In stage 1 (labeling; sub-stages A-C), we prepared the data and generated intent labels. In sub-stage A, we applied three LLM-based pre-processing steps: anonymization to protect sensitive information, cleaning to remove noise, and completion classification to identify incomplete conversations. While anonymization and cleaning improved the dataset's consistency, filtering out conversations classified as incomplete negatively impacted fine-tuning performance. This suggests that partially complete conversations still contain valuable examples for training. For subsequent experiments, we retained incomplete conversations while applying anonymization and cleaning, and this processed dataset was used both for response-only fine-tuning and for the intent-guided experiments.

The intent discovery stage was motivated by the limited coverage of existing human labels and the cost of scaling manual annotation for emerging intents. In sub-stages B and C, we therefore used a small human-labeled subset for reference and automated large-scale labeling with GIFA (optimized via GPFA and Hyperband-SEE). Retrieval during labeling improved intent reuse, with low-depth semantic search giving the best balance of coverage and duplication control (measured by reuse rate). To refine target granularity, we used Hyperband-SEE to find prompts whose GIFA clusters best matched a labeled reference set, outperforming SEE by a 19% performance increase in NCA$\times$AMI. Within GIFA/GPFA, concise intent representations, short feedback cycles, and direct (non-lagged) feedback proved most effective, avoiding the noise introduced by longer or more complex optimization routines. We applied the best-found prompt to all available customer-service conversations (26,253 conversations, 367,157 messages), which generated over 24,000 unique intents with a controlled long-tail distribution, containing common as well as rare intents, matching the distribution of the labeled intent dataset.

Together, these findings answer **RQ-A:** *Can we automatically generate reusable, appro-*

priately granular message-level intents from noisy, multi-turn customer-service conversations?

We showed that it is feasible to automatically generate reusable, appropriately granular message-level intents from noisy, multi-turn conversations. LLM-based pre-processing improved data consistency without discarding incomplete but valuable conversations. GIFA, optimized via GPFA and Hyperband-SEE, produced over 24,000 intents with a controlled long-tail distribution, covering both frequent and rare cases in line with the human-labeled intents. However, the discovered intents still included near-duplicates, and clustering quality remained modest in absolute terms (NCA$\times$AMI score of 0.465). Future work could explore iterative merge steps or alternative multi-objective optimization to refine granularity and reuse further.

In stage 2 (modeling; sub-stages D-E), we evaluated models using automatic metrics: lexical (BLEU, ROUGE-L, METEOR) and semantic (BERTScore). CoT was the strongest of the intent-guided variants but did not outperform the response-only baseline. We therefore evaluated RAFT for both CoT and response-only. Adding retrieved knowledge during training improved both, and both RAFT variants outperformed the base open-weight model with RAG, indicating that the model learns to use information more effectively when exposed to it during training, with RAFT, CoT and response-only remained comparable, and scaling from 1.7B to 4B improved scores without changing this pattern.

After selecting the best open-weight setups, we conducted a final comparison against a closed-weight GPT-4o RAG system using G-Eval to assess helpfulness, semantic similarity, and professionalism. The best open-weight RAFT models matched or slightly exceeded GPT-4o on semantic alignment and professionalism, while GPT-4o maintained an advantage in helpfulness. For rare, knowledge-heavy intents, assessed here only via semantic similarity, GPT-4o performed better, whereas RAFT was stronger on specific, common intents such as greetings and verification requests. Although explicit intent labels did not directly improve fine-tuning performance, they remain valuable as an evaluation and diagnostic tool: by revealing which intents a model handles poorly, they can guide targeted prompt or retrieval optimization for both open- and closed-weight systems.

Together, these findings answer **RQ-B:** *Given conversations labeled with intents, how do intent signals and retrieval strategies affect the quality/efficiency of an open-weight model trained to generate the next assistant reply, and how does it compare to the GPT-4o RAG baseline?*

Intent signals did not improve fine-tuning performance over response-only setups, but retrieval did: RAFT consistently outperformed RAG by integrating external knowledge

during training. Scaling models from 1.7B to 4B further improved quality without changing this pattern. Compared to GPT-4o RAG, the best open-weight RAFT models matched or slightly exceeded GPT-4o on semantic alignment and professionalism. However, GPT-4o remained stronger on helpfulness and rare, knowledge-heavy intents (assessed here via semantic similarity). This shows that open-weight RAFT models can outperform GPT-4o performance for routine intents, while GPT-4o remains stronger on rare intents.

Although intent guidance did not directly improve fine-tuning performance, it remains valuable as an evaluation and optimization signal. By identifying underperforming intent clusters, we can adapt GPT-4o prompts and configure intent-specific RAG pipelines that strengthen coverage for knowledge-heavy requests. These same intent-driven refinements, both in prompt design and retrieval targeting, can be applied when fine-tuning larger open-weight models with a smaller generalization gap, helping them close the remaining gap with closed-weight systems. This way, improvements to the GPT-4o pipeline will contribute to advancing open-weight RAFT models toward parity on routine and knowledge-intensive customer-service tasks.

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the International Conference on Neural Information Processing Systems, NeurIPS*, pp. 5998–6008, 2017.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Proceedings of the International Conference on Neural Information Processing Systems, NeurIPS*, pp. 1877–1901, 2020.

[3] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. C. Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz, D. Livshits, D. Wyatt, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Dinan, E. M. Smith, F. Radenovic, F. Guzmán, F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Thattai, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. A. Ibarra, I. Kloumann, I. Misra, I. Evtimov, J. Zhang, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. van der Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. V. Alwala, K. Prasad, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, K. El-Arini, K. Iyer, K. Malik, K. Chiu, K. Bhalla, K. Lakhotia, L. Rantala-Yeary, L. van der Maaten, L. Chen, L. Tan, L. Jenkins, L. Martin, L. Madaan, L. Malo, L. Blecher, L. Landzaat, L. de Oliveira, M. Muzzi, M. Pasupuleti, M. Singh, M. Paluri, M. Kardas, M. Tsimpoukelli, M. Oldham, M. Rita, M. Pavlova, M. Kambadur, M. Lewis, M. Si, M. K. Singh, M. Hassan, N. Goyal, N. Torabi, N. Bashlykov, N. Bogoychev, N. Chatterji, N. Zhang, O. Duchenne, O. Çelebi, P. Alrassy, P. Zhang, P. Li, P. Vasic, P. Weng, P. Bhargava, P. Dubal, P. Krishnan, P. S. Koura, P. Xu, Q. He, Q. Dong,

R. Srinivasan, R. Ganapathy, R. Calderer, R. S. Cabral, R. Stojnic, R. Raileanu, R. Maheswari, R. Girdhar, R. Patel, R. Sauvestre, R. Polidoro, R. Sumbaly, R. Taylor, R. Silva, R. Hou, R. Wang, S. Hosseini, S. Chennabasappa, S. Singh, S. Bell, S. S. Kim, S. Edunov, S. Nie, S. Narang, S. Raparthy, S. Shen, S. Wan, S. Bhosale, S. Zhang, S. Vandenhende, S. Batra, S. Whitman, S. Sootla, S. Collot, S. Gururangan, S. Borodinsky, T. Herman, T. Fowler, T. Sheasha, T. Georgiou, T. Scialom, T. Speckbacher, T. Mihaylov, T. Xiao, U. Karn, V. Goswami, V. Gupta, V. Ramanathan, V. Kerkez, V. Gonguet, V. Do, V. Vogeti, V. Albiero, V. Petrovic, W. Chu, W. Xiong, W. Fu, W. Meers, X. Martinet, X. Wang, X. Wang, X. E. Tan, X. Xia, X. Xie, X. Jia, X. Wang, Y. Goldschlag, Y. Gaur, Y. Babaei, Y. Wen, Y. Song, Y. Zhang, Y. Li, Y. Mao, Z. D. Coudert, Z. Yan, Z. Chen, Z. Papakipos, A. Singh, A. Srivastava, A. Jain, A. Kelsey, A. Shajnfeld, A. Gangidi, A. Victoria, A. Goldstand, A. Menon, A. Sharma, A. Boesenberg, A. Baevski, A. Feinstein, A. Kallet, A. Sangani, A. Teo, A. Yunus, A. Lupu, A. Alvarado, A. Caples, A. Gu, A. Ho, A. Poulton, A. Ryan, A. Ramchandani, A. Dong, A. Franco, A. Goyal, A. Saraf, A. Chowdhury, A. Gabriel, A. Bharambe, A. Eisenman, A. Yazdan, B. James, B. Maurer, B. Leonhardi, B. Huang, B. Loyd, B. D. Paola, B. Paranjape, B. Liu, B. Wu, B. Ni, B. Hancock, B. Wasti, B. Spence, B. Stojkovic, B. Gamido, B. Montalvo, C. Parker, C. Burton, C. Mejia, C. Liu, C. Wang, C. Kim, C. Zhou, C. Hu, C.-H. Chu, C. Cai, C. Tindal, C. Feichtenhofer, C. Gao, D. Civin, D. Beaty, D. Kreymer, D. Li, D. Adkins, D. Xu, D. Testuggine, D. David, D. Parikh, D. Liskovich, D. Foss, D. Wang, D. Le, D. Holland, E. Dowling, E. Jamil, E. Montgomery, E. Presani, E. Hahn, E. Wood, E.-T. Le, E. Brinkman, E. Arcaute, E. Dunbar, E. Smothers, F. Sun, F. Kreuk, F. Tian, F. Kokkinos, F. Ozgenel, F. Caggioni, F. Kanayet, F. Seide, G. M. Florez, G. Schwarz, G. Badeer, G. Swee, G. Halpern, G. Herman, G. Sizov, Guangyi, Zhang, G. Lakshminarayanan, H. Inan, H. Shojanazeri, H. Zou, H. Wang, H. Zha, H. Habeeb, H. Rudolph, H. Suk, H. Aspegren, H. Goldman, H. Zhan, I. Damlaj, I. Molybog, I. Tufanov, I. Leontiadis, I.-E. Veliche, I. Gat, J. Weissman, J. Geboski, J. Kohli, J. Lam, J. Asher, J.-B. Gaya, J. Marcus, J. Tang, J. Chan, J. Zhen, J. Reizenstein, J. Teboul, J. Zhong, J. Jin, J. Yang, J. Cummings, J. Carvill, J. Shepard, J. McPhie, J. Torres, J. Ginsburg, J. Wang, K. Wu, K. H. U, K. Saxena, K. Khandelwal, K. Zand, K. Matosich, K. Veeraraghavan, K. Michelena, K. Li, K. Jagadeesh, K. Huang, K. Chawla, K. Huang, L. Chen, L. Garg, L. A, L. Silva, L. Bell, L. Zhang, L. Guo, L. Yu, L. Moshkovich, L. Wehrstedt, M. Khabsa, M. Avalani, M. Bhatt, M. Mankus, M. Hasson, M. Lennie, M. Reso, M. Groshev, M. Naumov, M. Lathi, M. Keneally, M. Liu, M. L. Seltzer, M. Valko, M. Restrepo, M. Patel, M. Vyatskov, M. Samvelyan, M. Clark, M. Macey, M. Wang, M. J. Hermoso, M. Metanat, M. Rastegari, M. Bansal, N. Santhanam, N. Parks, N. White, N. Bawa, N. Singhal, N. Egebo, N. Usunier, N. Mehta, N. P. Laptev,

N. Dong, N. Cheng, O. Chernoguz, O. Hart, O. Salpekar, O. Kalinli, P. Kent, P. Parekh, P. Saab, P. Balaji, P. Rittner, P. Bontrager, P. Roux, P. Dollar, P. Zvyagina, P. Ratanchandani, P. Yuvraj, Q. Liang, R. Alao, R. Rodriguez, R. Ayub, R. Murthy, R. Nayani, R. Mitra, R. Parthasarathy, R. Li, R. Hogan, R. Battey, R. Wang, R. Howes, R. Rinott, S. Mehta, S. Siby, S. J. Bondu, S. Datta, S. Chugh, S. Hunt, S. Dhillon, S. Sidorov, S. Pan, S. Mahajan, S. Verma, S. Yamamoto, S. Ramaswamy, S. Lindsay, S. Lindsay, S. Feng, S. Lin, S. C. Zha, S. Patil, S. Shankar, S. Zhang, S. Zhang, S. Wang, S. Agarwal, S. Sajuyigbe, S. Chintala, S. Max, S. Chen, S. Kehoe, S. Satterfield, S. Govindaprasad, S. Gupta, S. Deng, S. Cho, S. Virk, S. Subramanian, S. Choudhury, S. Goldman, T. Remez, T. Glaser, T. Best, T. Koehler, T. Robinson, T. Li, T. Zhang, T. Matthews, T. Chou, T. Shaked, V. Vontimitta, V. Ajayi, V. Montanez, V. Mohan, V. S. Kumar, V. Mangla, V. Ionescu, V. Poenaru, V. T. Mihailescu, V. Ivanov, W. Li, W. Wang, W. Jiang, W. Bouaziz, W. Constable, X. Tang, X. Wu, X. Wang, X. Wu, X. Gao, Y. Kleinman, Y. Chen, Y. Hu, Y. Jia, Y. Qi, Y. Li, Y. Zhang, Y. Zhang, Y. Adi, Y. Nam, Yu, Wang, Y. Zhao, Y. Hao, Y. Qian, Y. Li, Y. He, Z. Rait, Z. DeVito, Z. Rosnbrick, Z. Wen, Z. Yang, Z. Zhao, and Z. Ma, "The Llama 3 herd of models," *CoRR*, vol. abs/2407.21783, 2024.

[4] J. Weizenbaum, "ELIZA-A computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[5] K. M. Colby, S. Weber, and F. D. Hilf, "Artificial paranoia," *Artificial Intelligence*, vol. 2, no. 1, pp. 1–25, 1971.

[6] X. Ni, Y. Wang, T. Feng, L. X. Lu, Y. Wang, and C. Zhou, "Generative AI in action: Field experimental evidence on worker performance in E-commerce customer service operations," *SSRN*, 2024.

[7] E. Brynjolfsson, D. Li, and L. Raymond, "Generative AI at work," *The Quarterly Journal of Economics*, vol. 140, no. 2, pp. 889–942, 2025.

[8] J. Liu, Y. Li, and M. Lin, "Review of intent detection methods in the human-machine dialogue system," *Journal of Physics: Conference Series*, vol. 1267, no. 1, p. 012059, 2019.

[9] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan,

Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, and Z. Qiu, "Qwen3 technical report," *CoRR*, vol. abs/2505.09388, 2025.

[10] V. Hanke, T. Blanchard, F. Boenisch, I. E. Olatunji, M. Backes, and A. Dziedzic, "Open LLMs are necessary for current private adaptations and outperform their closed alternatives," in *Proceedings of the International Conference on Neural Information Processing Systems, NeurIPS*, pp. 1220–1250, 2025.

[11] J. A. Rodríguez, N. Botzer, D. Vázquez, C. Pal, M. Pedersoli, and I. H. Laradji, "IntentGPT: Few-shot intent discovery with large language models," *CoRR*, vol. abs/2411.10670, 2024.

[12] M. D. Raedt, F. Godin, T. Demeester, and C. Develder, "IDAS: Intent discovery with abstractive summarization," in *Proceedings of the Workshop on NLP for Conversational AI, NLP4ConvAI*, pp. 71–88, 2023.

[13] X. Wang, C. Li, Z. Wang, F. Bai, H. Luo, J. Zhang, N. Jojic, E. Xing, and Z. Hu, "PromptAgent: Strategic planning with language models enables expert-level prompt optimization," in *Proceedings of the International Conference on Learning Representations, ICLR*, 2024.

[14] W. Cui, J. Zhang, Z. Li, H. Sun, D. Lopez, K. Das, B. A. Malin, and S. Kumar, "SEE: Strategic exploration and exploitation for cohesive in-context prompt optimization," in *Proceedings of the Association for Computational Linguistics, ACL*, pp. 29575–29627, 2025.

[15] R. E. Banchs and H. Li, "IRIS: A chat-oriented dialogue system based on the vector space model," in *Proceedings of the Association for Computational Linguistics, ACL*, pp. 37–42, 2012.

[16] E. Adamopoulou and L. Moussiades, "Chatbots: History, technology, and applications," *Machine Learning with Applications*, vol. 2, p. 100006, 2020.

[17] C.-H. Li, K. Chen, and Y.-J. Chang, "When there is no progress with a task-oriented chatbot: A conversation analysis," in *Proceedings of the International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI*, pp. 59:1–59:6, 2019.

[18] K. Mitsuda, R. Higashinaka, T. Li, and S. Yoshida, "Investigating person-specific errors in chat-oriented dialogue systems," in *Proceedings of the Association for Computational Linguistics, ACL*, pp. 464–469, 2022.

[19] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, "Building Watson: An overview of the DeepQA project," *AI Magazine*, vol. 31, no. 3, pp. 59–79, 2010.

[20] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 1724–1734, 2014.

[23] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, A. Madry, A. Baker-Whitcomb, A. Beutel, A. Borzunov, A. Carney, A. Chow, A. Kirillov, A. Nichol, A. Paino, A. Renzin, A. T. Passos, A. Kirillov, A. Christakis, A. Conneau, A. Kamali, A. Jabri, A. Moyer, A. Tam, A. Crookes, A. Tootoochian, A. Tootoonchian, A. Kumar, A. Vallone, A. Karpathy, A. Braunstein, A. Cann, A. Codispoti, A. Galu, A. Kondrich, A. Tulloch, A. Mishchenko, A. Baek, A. Jiang, A. Pelisse, A. Woodford, A. Gosalia, A. Dhar, A. Pantuliano, A. Nayak, A. Oliver, B. Zoph, B. Ghorbani, B. Leimberger, B. Rossen, B. Sokolowsky, B. Wang, B. Zweig, B. Hoover, B. Samic, B. McGrew, B. Spero, B. Giertler, B. Cheng, B. Lightcap, B. Walkin, B. Quinn, B. Guarraci, B. Hsu, B. Kellogg, B. Eastman, C. Lugaresi, C. Wainwright, C. Bassin, C. Hudson, C. Chu, C. Nelson, C. Li, C. J. Shern, C. Conger, C. Barette, C. Voss, C. Ding, C. Lu, C. Zhang, C. Beaumont, C. Hallacy, C. Koch, C. Gibson, C. Kim, C. Choi, C. McLeavey, C. Hesse, C. Fischer, C. Winter, C. Czarnecki, C. Jarvis, C. Wei, C. Koumouzelis, D. Sherburn, D. Kappler, D. Levin, D. Levy, D. Carr, D. Farhi, D. Mely, D. Robinson, D. Sasaki, D. Jin, D. Valladares, D. Tsipras, D. Li, D. P. Nguyen, D. Findlay, E. Oiwoh, E. Wong, E. Asdar, E. Proehl, E. Yang, E. Antonow, E. Kramer, E. Peterson, E. Sigler, E. Wallace, E. Brevdo, E. Mays, F. Khorasani, F. P. Such, F. Raso, F. Zhang, F. von Lohmann, F. Sulit, G. Goh, G. Oden, G. Salmon, G. Starace, G. Brockman, H. Salman, H. Bao, H. Hu, H. Wong, H. Wang, H. Schmidt, H. Whitney, H. Jun, H. Kirchner, H. P. de Oliveira Pinto, H. Ren, H. Chang, H. W. Chung, I. Kivlichan, I. O'Connell, I. O'Connell, I. Osband, I. Silber, I. Sohl, I. Okuyucu, I. Lan, I. Kostrikov, I. Sutskever, I. Kanitscheider, I. Gulrajani, J. Coxon, J. Menick, J. Pachocki, J. Aung, J. Betker, J. Crooks, J. Lennon, J. Kiros, J. Leike, J. Park, J. Kwon, J. Phang, J. Teplitz, J. Wei, J. Wolfe, J. Chen, J. Harris, J. Varavva, J. G.

Lee, J. Shieh, J. Lin, J. Yu, J. Weng, J. Tang, J. Yu, J. Jang, J. Q. Candela, J. Beutler, J. Landers, J. Parish, J. Heidecke, J. Schulman, J. Lachman, J. McKay, J. Uesato, J. Ward, J. W. Kim, J. Huizinga, J. Sitkin, J. Kraaijeveld, J. Gross, J. Kaplan, J. Snyder, J. Achiam, J. Jiao, J. Lee, J. Zhuang, J. Harriman, K. Fricke, K. Hayashi, K. Singhal, K. Shi, K. Karthik, K. Wood, K. Rimbach, K. Hsu, K. Nguyen, K. Gu-Lemberg, K. Button, K. Liu, K. Howe, K. Muthukumar, K. Luther, L. Ahmad, L. Kai, L. Itow, L. Workman, L. Pathak, L. Chen, L. Jing, L. Guy, L. Fedus, L. Zhou, L. Mamitsuka, L. Weng, L. McCallum, L. Held, L. Ouyang, L. Feuvrier, L. Zhang, L. Kondraciuk, L. Kaiser, L. Hewitt, L. Metz, L. Doshi, M. Aflak, M. Simens, M. Boyd, M. Thompson, M. Dukhan, M. Chen, M. Gray, M. Hudnall, M. Zhang, M. Aljubeh, M. Litwin, M. Zeng, M. Johnson, M. Shetty, M. Gupta, M. Shah, M. Yatbaz, M. J. Yang, M. Zhong, M. Glaese, M. Chen, M. Janner, M. Lampe, M. Petrov, M. Wu, M. Wang, M. Fradin, M. Pokrass, M. Castro, M. O. T. de Castro, M. Pavlov, M. Brundage, M. Wang, M. Khan, M. Murati, M. Bavarian, M. Lin, M. Yesildal, N. Soto, N. Gimelshein, N. Cone, N. Staudacher, N. Summers, N. LaFontaine, N. Chowdhury, N. Ryder, N. Stathas, N. Turley, N. Tezak, N. Felix, N. Kudige, N. Keskar, N. Deutsch, N. Bundick, N. Puckett, O. Nachum, O. Okelola, O. Boiko, O. Murk, O. Jaffe, O. Watkins, O. Godement, O. Campbell-Moore, P. Chao, P. McMillan, P. Belov, P. Su, P. Bak, P. Bakkum, P. Deng, P. Dolan, P. Hoeschele, P. Welinder, P. Tillet, P. Pronin, P. Tillet, P. Dhariwal, Q. Yuan, R. Dias, R. Lim, R. Arora, R. Troll, R. Lin, R. G. Lopes, R. Puri, R. Miyara, R. Leike, R. Gaubert, R. Zamani, R. Wang, R. Donnelly, R. Honsby, R. Smith, R. Sahai, R. Ramchandani, R. Huet, R. Carmichael, R. Zellers, R. Chen, R. Chen, R. Nigmatullin, R. Cheu, S. Jain, S. Altman, S. Schoenholz, S. Toizer, S. Miserendino, S. Agarwal, S. Culver, S. Ethersmith, S. Gray, S. Grove, S. Metzger, S. Hermani, S. Jain, S. Zhao, S. Wu, S. Jomoto, S. Wu, Shuaiqi, Xia, S. Phene, S. Papay, S. Narayanan, S. Coffey, S. Lee, S. Hall, S. Balaji, T. Broda, T. Stramer, T. Xu, T. Gogineni, T. Christianson, T. Sanders, T. Patwardhan, T. Cunninghman, T. Degry, T. Dimson, T. Raoux, T. Shadwell, T. Zheng, T. Underwood, T. Markov, T. Sherbakov, T. Rubin, T. Stasi, T. Kaftan, T. Heywood, T. Peterson, T. Walters, T. Eloundou, V. Qi, V. Moeller, V. Monaco, V. Kuo, V. Fomenko, W. Chang, W. Zheng, W. Zhou, W. Manassra, W. Sheu, W. Zaremba, Y. Patil, Y. Qian, Y. Kim, Y. Cheng, Y. Zhang, Y. He, Y. Zhang, Y. Jin, Y. Dai, and Y. Malkov, "GPT-4o system card," *CoRR*, vol. abs/2410.21276, 2024.

[24] S. H. Kim, S. Schramm, L. C. Adams, R. Braren, K. K. Bressem, M. Keicher, P.-S. Platzek, K. J. Paprottka, C. Zimmer, D. M. Hedderich, and B. Wiestler, "Benchmarking the diagnostic performance of open source LLMs in 1933 Eurorad case reports," *npj Digital Medicine*, vol. 8, no. 1, p. 97, 2025.

[25] T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol, "DIET: lightweight language understanding for dialogue systems," *CoRR*, vol. abs/2004.09936, 2020.

[26] J. Zhang, K. Hashimoto, W. Liu, C.-S. Wu, Y. Wan, P. Yu, R. Socher, and C. Xiong, "Discriminative nearest neighbor few-shot intent detection by transferring natural language inference," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 5064–5082, 2020.

[27] L. Shu, H. Xu, and B. Liu, "DOC: Deep open classification of text documents," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 2911–2916, 2017.

[28] C. Shi, Q. Chen, L. Sha, S. Li, X. Sun, H. Wang, and L. Zhang, "Auto-Dialabel: Labeling dialogue data with unsupervised learning," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 684–689, 2018.

[29] H. Zhang, H. Xu, T.-E. Lin, and R. Lyu, "Discovering new intents with deep aligned clustering," in *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, pp. 14365–14373, 2021.

[30] R. Kumar, M. Patidar, V. Varshney, L. Vig, and G. Shroff, "Intent detection and discovery from user logs via deep semi-supervised contrastive clustering," in *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL*, pp. 1836–1853, 2022.

[31] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 3045–3059, 2021.

[32] Z. Wang, R. Panda, L. Karlinsky, R. Feris, H. Sun, and Y. Kim, "Multitask prompt tuning enables parameter-efficient transfer learning," in *Proceedings of the International Conference on Learning Representations, ICLR*, 2023.

[33] A. Prasad, P. Hase, X. Zhou, and M. Bansal, "GrIPS: Gradient-free, edit-based instruction search for prompting large language models," in *Proceedings of the European Chapter of the Association for Computational Linguistics, EACL*, pp. 3845–3864, 2023.

[34] R. Pryzant, D. Iter, J. Li, Y. Lee, C. Zhu, and M. Zeng, "Automatic prompt optimization with "gradient descent" and beam search," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 7957–7968, 2023.

[35] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," in *Proceedings of the International Conference on Learning Representations, ICLR*, 2023.

[36] Q. Guo, R. Wang, J. Guo, B. Li, K. Song, X. Tan, G. Liu, J. Bian, and Y. Yang, "Connecting large language models with evolutionary algorithms yields powerful prompt optimizers," in *Proceedings of the International Conference on Learning Representations, ICLR*, 2024.

[37] C. Fernando, D. Banarse, H. Michalewski, S. Osindero, and T. Rocktäschel, "Promptbreeder: Self-referential self-improvement via prompt evolution," in *Proceedings of the International Conference on Machine Learning, ICML*, pp. 541:1–541:64, 2024.

[38] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, pp. 6765–6816, 2017.

[39] S. Liu, C. Zheng, O. Demasi, S. Sabour, Y. Li, Z. Yu, Y. Jiang, and M. Huang, "Towards emotional support dialog systems," in *Proceedings of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing, ACL-IJCNLP*, pp. 3469–3483, 2021.

[40] B. Liao, Y. Meng, and C. Monz, "Parameter-efficient fine-tuning without introducing new latency," in *Proceedings of the Association for Computational Linguistics, ACL*, pp. 4242–4260, 2023.

[41] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *Proceedings of the International Conference on Learning Representations ICLR*, 2022.

[42] Z. Zheng, L. Liao, Y. Deng, and L. Nie, "Building emotional support chatbots in the era of LLMs," *CoRR*, vol. abs/2308.11584, 2023.

[43] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," in *Proceedings of the International Conference on Neural Information Processing Systems, NeurIPS*, pp. 10088–10115, 2023.

[44] W. Lu, R. K. Luu, and M. J. Buehler, "Fine-tuning large language models for domain adaptation: Exploration of training strategies, scaling, model merging and synergistic capabilities," *npj Computational Materials*, vol. 11, no. 1, p. 84, 2025.

[45] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," in *Proceedings of the International Conference on Neural Information Processing Systems, NeurIPS*, pp. 2011–2025, 2022.

[46] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-Thought prompting elicits reasoning in large language models," in *Proceedings of the International Conference on Neural Information Processing Systems, NeurIPS*, pp. 1800–1813, 2022.

[47] R. Chada and P. Natarajan, "FewshotQA: A simple framework for few-shot learning of question answering tasks using pre-trained text-to-text models," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 6081–6090, 2021.

[48] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proceedings of the International Conference on Neural Information Processing Systems, NeurIPS*, pp. 793–808, 2020.

[49] J. Chen and J. Mueller, "Quantifying uncertainty in answers from any language model and enhancing their trustworthiness," in *Proceedings of the Association for Computational Linguistics, ACL*, pp. 5186–5200, 2024.

[50] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[51] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Text embeddings by weakly-supervised contrastive pre-training," *CoRR*, vol. abs/2212.03533, 2022.

[52] T. Zhang, S. G. Patil, N. Jain, S. Shen, M. Zaharia, I. Stoica, and J. E. Gonzalez, "RAFT: adapting language model to domain specific RAG," *CoRR*, vol. abs/2403.10131, 2024.

[53] G. V. Cormack, C. L. A. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms Condorcet and individual rank learning methods," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*, pp. 758–759, 2009.

[54] M. Gagolewski, "Normalised clustering accuracy: An asymmetric external cluster validity measure," *Journal of Classification*, vol. 42, no. 1, pp. 2–30, 2025.

[55] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.

[56] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating text generation with BERT," in *Proceedings of the International Conference on Learning Representations, ICLR*, 2020.

[57] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "G-Eval: NLG evaluation using GPT-4 with better human alignment," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 2511–2522, 2023.

A LLM-Based Pre-Processing Details

The prompt in this appendix was used for the three LLM-based pre-processing steps: cleaning, anonymization, and completion classification (subsection 3.1.2).

Furthermore, the exact fields that were anonymized can be seen in the following list:

- Names (customers and assistants) → `[NAME]`
- Phone numbers → `[PHONE]`
- Email addresses → `[EMAIL]`
- Order numbers → `[ORDER_ID]`
- Bank account numbers → `[BANK_ACCOUNT]`
- Loyalty card numbers → `[LOYALTY_CARD]`
- Address details (street names, postal codes, cities) → `[ADDRESS]`, `[POSTAL_CODE]`, `[CITY]`

In addition, several domain-specific entities are masked to promote generalization during fine-tuning:

- Product or brand names (e.g., ”Heinz ketchup”) → `[PRODUCT]`
- Prices (e.g., ”€3.50”) → `[PRICE]`
- Quantity references (e.g., ”two bottles of milk”) → `[COUNT]`
- Delivery windows (e.g., ”between 4 and 5 PM”) → `[TIME_RANGE]`
- Specific times (e.g., ”09:00”) → `[TIME]`
- Dates (e.g., ”tomorrow,” ”November 7”) → `[DATE]`

System Prompt for Preprocessor

Task:

You are a preprocessing assistant for Dutch grocery-store customer service conversations. Your goal is to prepare these conversations clearly and consistently for fine-tuning an LLM to handle customer service tasks generically.

Task Overview:

- **Mask sensitive information** (personal and grocery-related).
- **Merge or delete repetitive messages** that don't add meaningful information.
- **Split messages containing clearly distinct customer intents**.
- Clearly indicate if a customer issue remains unresolved at the end.
- **Mark each message with `important:` `true`** if it contains key steps.

Instructions:

1. Merge or Delete repetitive messages

- Merge customer follow-ups that:
  - repeat questions without new info (”Hallo?”, ”??”)
  - add urgency (”Pfff”, ”Gezeik”)
- Delete assistant system messages:
  - Assistant switches: ”Mijn collega neemt over”
  - Timeouts, surveys, etc.
- Keep only the first assistant intro, mask name as `[NAME]`.

2. Mask sensitive personal information

- Names → `[NAME]`
- Phone numbers → `[PHONE]`
- Emails → `[EMAIL]`
- Order numbers → `[ORDER_ID]`
- Bank account numbers → `[BANK_ACCOUNT]`
- Loyalty card numbers → `[LOYALTY_CARD]`

- Addresses → `[ADDRESS]`, `[POSTAL_CODE]`, `[CITY]`

3. Mask grocery-specific sensitive information

- Product or brand names → `[PRODUCT]`
- Prices → `[PRICE]`
- Quantities → `[COUNT]`
- Time ranges → `[TIME_RANGE]` or `[TIME]`
- Dates → `[DATE]`

Keep generic words like "melk", "fruit", "brood".

4. Split messages with distinct intents

- Example intents: `request_order_cancellation`, `inquire_delivery_time`, `complain_product_quality`, etc.
- Example: `"Ik wil mijn bestelling annuleren en weten wanneer mijn boodschappen komen."` → two messages

5. Indicate unresolved issues

- Add `"text": "not solved"` if:
  - Conversation ends with no resolution
  - Assistant fails to respond meaningfully
- Do not add if:
  - Assistant acknowledges limitation or clarifies valid reasoning

6. Annotate important messages

- **Mark as `important: true`** if message contributes to resolution.
- Includes: complaints, order info, assistant clarifications, status updates, etc.
- Mark greetings, small talk, or system closings as `important: false`.

Constraints:

- Do **not** reorder messages.
- Only delete, split, or merge messages.

- Every output must map to input message(s).

Output Format:

```
[
  {
    "speaker": "customer",
    "text": "Ik wil mijn bestelling annuleren.",
    "important": true
  },
  {
    "speaker": "assistant",
    "text": "Je kunt bellen naar [PHONE].",
    "important": true
  },
  {
    "speaker": "assistant",
    "text": "not solved",
    "important": false
  }
]
```

Now process this conversation: {conversation}

B Medium-Granularity Dataset

To validate our claim made in subsection 4.1 that the generated intents for the conversational dataset should be high-granularity, transforming intent detection from a classification task to a generative task, we also did an ablation experiment with a labeled intent dataset of 250 conversations, where the granularity was kept at a medium level. Hyperband-SEE combined with GIFA was applied to the labeled dataset, and the prompt can be seen in Figure B.1.

Over the 26,253 conversations, the medium-intent prompt found 1,234 intents. The amount of unique intents found over conversations and the intent frequency distribution can be found in Figure B.1. The intents found over time show a similar trend as the high-granularity dataset: over the first conversations, many unique intents are found, and after a certain amount of conversation, the trend becomes linear.

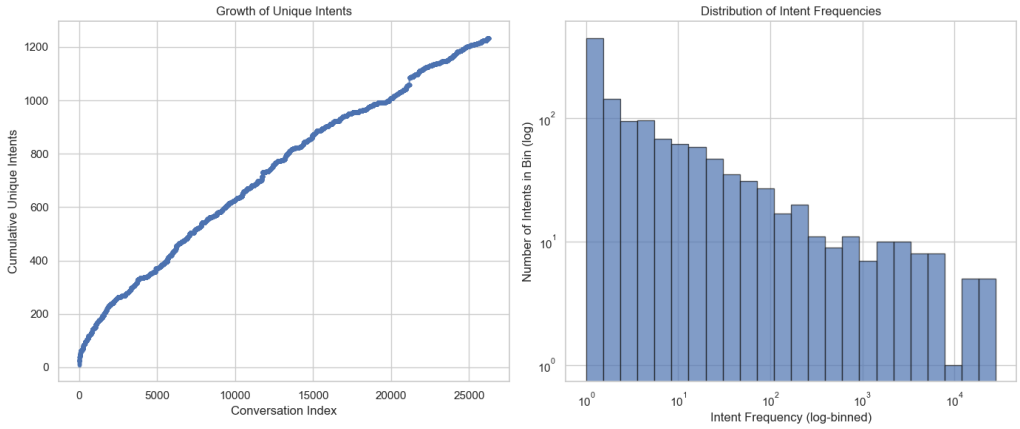

Figure B.1: **Medium-granularity ablation.** With the medium-granularity intent finding prompt, intent discovery over 26,253 conversations finds 1,234 intents; unique-intent growth mirrors the high-granularity case. The frequency distribution is long-tailed, shown as a histogram with bin edges spaced evenly on a $\log_{10}$ scale between the minimum and maximum observed intent frequencies (25 bins total).

The same fine-tuning setup was kept as in the phase 1 fine-tuning experiments, where a Qwen3-1.7B model was fine-tuned on all intent-driven fine-tuning setups. We compared the best-found intent-guided setup from phase 1, CoT, across both intent granularity datasets. Automatic evaluation metrics were computed and are shown in Table B.1.

Model	BLEU	ROUGE-L	METEOR	BERTScore
1.7B High CoT	**10.205**	**0.255**	**0.302**	**0.898**
1.7B Medium CoT	8.875	0.240	0.281	0.893

Table B.1: **Medium versus high intent granularity (CoT).** Qwen3-1.7B CoT fine-tuning on medium- versus high-granularity intent datasets; BLEU, ROUGE-L, METEOR, and BERTScore.

Table B.1 suggests that fine-tuning the Qwen3-1.7B using a CoT setup is more effective when using the high-granularity conversational dataset than the medium-granularity dataset. This suggests that treating the intent prediction as a generative task instead of a classification task is preferred within the CoT setup. A possible explanation could be that the mix between classification for intents and generation for responses confuses the model. The model has to switch from a restricted choice between concise intents to a free-form generation of responses.

Below are the final prompts found with the Hyperband-SEE search algorithm for the medium- and high-granularity datasets.

Optimal High Granularity prompt for Intent Annotation

Task:

You are an annotation assistant for Dutch grocery-store customer service conversations. Your goal is to assign each input message a clear and specific intent label, ensuring consistency and granularity for fine-tuning intent classification models.

Task Overview:

- **Assign exactly one intent label** to each message.
- **Disambiguate subtle differences** in phrasing that indicate distinct intents.
- **Maintain contextual consistency** across similar system/assistant messages.
- **Handle multi-intent conversations** by labeling each message separately.
- **Use fine-grained intent labels** that reflect message nuance.
- **Ensure consistency in acknowledgments and gratitude** across messages.

Annotation Guidelines:

1. Identify Unique Intents

- Do not merge intents unless they are identical in meaning and context.
- Example:
  - "customer: Koopzegels uitbetaling niet ontvangen"
  - "customer: Heb gisteren een deel van mijn koopzegels uitbetaald, tot heden niks ontvangen.."

  Both → `report_issue_with_bonus_card_balance_issue`

2. Disambiguate Similar Phrases

- Differentiate expressions with subtle differences.
- Example:
  - "Dank je voor bericht." → `express_gratitude`
  - "Dank je voor bericht. Ik hoop t" → `express_gratitude_and_hope_for_resolution`

3. Contextual Consistency

- Ensure system/assistant messages with similar meaning share the same label.
- Example:
  - "Ik stuur je vraag even intern door" → `transfer_chat`
  - "Deborah is niet langer beschikbaar. Het gesprek wordt doorgezet naar een collega." → `transfer_chat`

4. Handle Mixed Conversations

- Label each message separately, even in multi-intent dialogues.
- Example:
  - "Bedankt voor je berichtje. Nadine heeft deze chatsessie beëindigd." → `close_chat_session`
  - "Deborah is niet langer beschikbaar. Het gesprek wordt doorgezet naar een collega." → `transfer_chat`

5. Granularity of Intent Labels

- Use specific labels to capture nuance.

- Example:
  - "Als je nu even inlogt kan mijn collega zometeen met je mee kijken" → `suggest_logging_in`
  - "Heb je verder nog andere vragen voor mij?" → `inquire_about_additional_questions`

6. Consistency in Acknowledgments and Gratitude

- Group acknowledgments consistently.
- Example:
  - "Komt goed, dankjewel voor het nakijken" → `express_gratitude`
  - "Graag gedaan hoor." → `express_gratitude`

Constraints:

- Each message must have exactly one intent label.
- Labels should remain consistent across similar wording.
- Do not invent new intents if an existing one matches appropriately.

Output Format:

```
[
  {
    "speaker": "customer",
    "text": "Koopzegels uitbetaling niet ontvangen",
    "intent": "report_issue_with_bonus_card_balance_issue"
  },
  {
    "speaker": "assistant",
    "text": "Ik stuur je vraag even intern door",
    "intent": "transfer_chat"
  }
]
```

Now assign intents for this conversation: {`conversation`}

Optimal Medium Granularity prompt for Intent Annotation

Task:

You are building an intent set for grocery-store customer service chats. For every incoming message, assign **exactly one** intent label.

Task Overview:

- Each message (customer or assistant) must receive exactly one intent.
- Intent names are short, consistent, and formatted `lowercase_with_underscores`.
- Always reuse existing labels if they fit (80% similarity) instead of inventing new ones.

Examples:

Greeting / Closings

- `greet_customer` → "Hi, je chat met [NAME]"; "Goedemiddag! Waarmee kan ik helpen?"
- `express_gratitude` → "Dankjewel!"; "Super, bedankt voor uw reactie."
- `close_chat_session` → "Fijne avond en tot gauw!"; "Bedankt voor de chat, dag!"

Identity / Verification

- `request_verification_details` → "Mag ik je e-mail en postcode?"; "Kunt u ter controle het bestelnummer doorgeven?"
- `provide_verification_details` → "Mijn e-mail is . . . "; "Bestelnummer: [ORDER_ID]"

Order / Delivery Problems

- `report_incomplete_order` → "Ik mis twee producten uit mijn bestelling."; "De krat is geleverd zonder [PRODUCT]."
- `request_missing_items_details` → "Welke items ontbreken precies?"

Guidelines:

1. Assign **exactly one** intent per message.

2. Do not merge different purposes into one label.
3. Use `lowercase_with_underscores` only.
4. Always label both `assistant:` and `customer:` messages.
5. Keep labels short and meaningful; avoid mixing Dutch/English.
6. Reuse existing intents before inventing new ones.

Full Conversation Example:

```
customer: "Iets mis met een geleverde bestelling"
        report_incomplete_order
assistant: "Ik kijk dit meteen voor je na."
        acknowledge_understanding
assistant: "Het is in mindering gebracht op factuur ..."
        confirm_refund_processed
assistant: "Heb je verder nog vragen?"
        inquire_additional_questions
assistant: "Mag ik je e-mail en postcode?"
        request_verification_details
customer: "[BANK_ACCOUNT]"
        provide_bank_account
assistant: "In dit geval jammer genoeg niet."
        deny_request
```

Output Format:

```
[
  {
    "speaker": "customer",
    "text": "Ik mis twee producten uit mijn bestelling.",
    "intent": "report_incomplete_order"
  },
  {
    "speaker": "assistant",
    "text": "Welke items ontbreken precies?",
    "intent": "request_missing_items_details"
  }
]
```

Now assign intents for this conversation: {`conversation`}

C Greedy and Random Search Prompts

The system prompt in this appendix was used as the starting prompt for the greedy and random prompt optimization discussed in subsection 4.2.

System Prompt for Intent Labeling

You are analyzing a customer service conversation to identify the underlying intentions behind each message. Assign a structured intent label to each message.

- Use an existing intent label if it already covers the message.
- Create a new intent only if the message introduces a distinct issue not covered by existing intents.

D G-Eval Prompts

The prompts in this appendix were input for G-Eval (subsection 5.8) to determine the helpfulness, semantic similarity, and professionalism scores.

System Prompt for Helpfulness Rating

Rate how helpful the actual output is in progressing the user toward their goal.

System Prompt for Semantic Similarity Rating

Assess how semantically similar the actual output is to the expected output.

System Prompt for Professionalism Rating

Assess if the response maintains the same tone and domain-appropriate language as the expected output.

E SEE Phase Prompts

This appendix shows the different prompts used within the phases of SEE (subsection 3.3.2).

Direct Feedback Prompt

You are an advanced AI system tasked with refining our intent discovery prompt for clustering customer service messages. Our evaluation of the current clustering approach has produced the following diagnostic results:

---

1. Global Clustering Metrics - Neighborhood Clustering Accuracy (NCA): {NCA} - Adjusted Mutual Information (AMI): {AMI} - Adjusted Rand Index (ARI): {ARI} - Fowlkes{Mallows Index (FMI): {FMI}

---

2. Error Analysis - False Splits: {False Splits} *(Same true intent → assigned to different predicted clusters.)* - **False Merges:** {False Merges} *(Different true intents → assigned to the same predicted cluster.)*

---

3. Intent Count - Predicted number of intents: {Predicted Amount} - Expected (true) number of intents: {True Amount}

Task

Based on this analysis, **refine and improve the existing intent discovery prompt** to enhance clustering performance. Return *only* the updated prompt|no extra commentary.

Error Examples

Each tuple includes two messages, their true and predicted intents, and the error type: {Interesting Examples}

Lagged Feedback Prompt

You are an advanced AI system assigned to improve an intent discovery prompt for clustering customer service messages.

Below is an evaluation summary of the current clustering performance:

{global_metrics_text}
Error Examples Each of the following pairs shows two messages, their true and predicted intents, and the type of error: {Interesting Examples}

Task
Based on this evaluation, generate a **list of specific feedback** to improve the existing intent discovery prompt.
You may:

- Adjust specificity based on the metrics above.
- Include examples from the list above to illustrate critical issues.
- Propose ways to rephrase instructions, add disambiguation cues, or encourage more/less specificity.

Avoid re-summarizing the metrics. Focus on **feedback that would help rewrite the intent discovery prompt.**

Lamarckian Extraction Prompt

I gave a friend an instruction and some inputs.
The friend read the instruction and wrote an output for every one of the inputs.
Give back ONLY instruction that was given such that it can be used as a prompt.
Here are the input-output pairs: ## Example ## {pairs}

Feedback Improver Prompt

You are a quick improver. Given an existing prompt and feedback on how it should improve. Create an improved version based on the feedback. ## Existing Prompt ## {existing_prompt} ## Feedback ## {feedback} ## Improved Prompt ##

Crossover Mutation Prompt

```
You are a mutator who is familiar with the concept of
cross-over in genetic algorithm, namely combining the genetic
information of two parents to generate new offspring.
Given two parent prompts, you will perform a cross-over to
generate an offspring prompt that covers the same semantic
meaning as both parents.
RETURN ONLY THE NEW PROMPT.
# Example Parent prompt 1:  Now you are a categorizer, your
mission is to ascertain the sentiment of the provided text,
either favorable or unfavorable
Parent prompt 2:  Assign a sentiment label to the given
sentence from ['negative','positive'] and return only the
label without any other text.
Offspring prompt:  Your mission is to ascertain the sentiment
of the provided text and assign a sentiment label from
['negative','positive'].
## Given ## Parent prompt 1:  {prompt1} Parent prompt 2:
{prompt2}
Offspring prompt:
```