CLARIGEN: PIPELINE FOR QUERY AMBIGUITY DETECTION AND CLARIFICATION

CLARIGEN: A MULTI-STAGE PIPELINE FOR DETECTING AND CLARIFYING AMBIGUOUS QUERIES

BY
CHI YU CHAN

A Report
submitted to the Department of Computing and Software
and the School of Graduate Studies
of McMaster University
in partial fulfilment of the requirements
for the degree of
Master of Engineering

Master of Engineering (2026)	McMaster University
(Department of Computing and Software)	Hamilton, Ontario, Canada

TITLE:	ClariGen: A Multi-Stage Pipeline for Detecting and Clarifying Ambiguous Queries
AUTHOR:	Chi Yu Chan B.Eng. Information Engineering, City University of Hong Kong
SUPERVISOR:	Dr. Rong Zheng
NUMBER OF PAGES:	ix, 78

Lay Abstract

Many users submit vague questions to search tools, which makes it hard for these systems to understand what they really mean. For instance, a query such as "Amazon" could refer either to the company or the river, while "exercises for balance" might relate to home workouts or structured exercise classes. Such unclear queries increase the risk of incorrect or unhelpful answers, and this risk is especially serious in high-stakes contexts, such as health information for older adults.

This project develops ClariGen, an AI system that detects unclear questions, figures out why they are unclear, asks a simple clarifying question, and then refines the original query more precisely before searching.

Tests showed that a lightweight instruction-tuned LLM can quickly detect unclear questions reliably, and a stronger model creates better clarifying questions when guided by a list of common ambiguity types and step-by-step reasoning. Real examples with questions about preventing falls in seniors demonstrated how the system successfully clarifies details like exercise setting or fear of falling, leading to more accurate search results.

Overall, ClariGen is an efficient way to make AI assistants smarter at handling unclear questions without needing of training data. It improves health information tools and other search systems where accurate interpretation of user intent is critical.

Abstract

Ambiguous user queries pose a significant challenge to retrieval-augmented generation (RAG) systems, often resulting in unclear or irrelevant responses. This report presents ClariGen, an end-to-end Large Language Model (LLM)-driven pipeline for interactive ambiguity resolution in Question-Answering (QA) systems. The system employs a dual-model "fast-slow" architecture: a lightweight instruction-tuned model (Llama-3.1-8B) performs low-latency binary ambiguity detection, while a larger model (Llama-3.3-70B) conducts in-depth ambiguity reasoning using a modified CLAMBER taxonomy and generates targeted clarifying questions.

Experiments demonstrate that zero-shot prompting achieves usable ambiguity detection, outperforming few-shot approaches prone to classification bias. For clarification generation, ambiguity-aware Chain-of-Thought (CoT) prompting yields the highest semantic similarity to human references (BERTScore F1 up to 0.9187 on ClariQ and 0.9093 on QULAC). End-to-end case studies on fall prevention queries for older adults illustrate effective resolution of lexical and semantic ambiguities through interactive reformulation.

ClariGen offers a scalable, prompt-based clarification module that balances computational efficiency with reasoning depth, providing a foundation for improving downstream retrieval performance in ambiguity-sensitive domains such as healthcare.

To my parents...

Acknowledgements

I would like to express my sincere thanks to my supervisor, Rong Zheng, for her patient guidance and unwavering support throughout my study journey at WiSeR Lab at McMaster. I am truly honored to have worked under her mentorship and am grateful for the trust and encouragement she provided at every stage of my project.

I am also deeply thankful to my family in Hong Kong for their constant support and confidence in me. Even across the distance, your messages, calls, and care gave me strength. Thank you for supporting my decision to pursue a master's degree and for always believing in me.

Abbreviations

AI	Artificial intelligence
AmbigNQ	Ambiguous Questions for Open-Domain QA (dataset)
AT-CoT	Ambiguity-Type Chain-of-Thought (prompting strategy)
AT-Standard	Ambiguity-Type Standard (prompting strategy)
BERTScore	BERT-based semantic similarity score
CLAMBER	Clarifying Ambiguous Query benchmark taxonomy
ClariQ	Clarification Question dataset (conversational search)
CoT	Chain-of-Thought
F1	F1 score
FSM	Finite State Machine
IR	Information Retrieval
LLMs	Large Language Models
MIMIC-CIR	MIMIC Clinical Information Retrieval (dataset)

QA	Question Answering
RAG	Retrieval-Augmented Generation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In modern Information Retrieval (IR) and QA systems, the effectiveness of responses largely depends on the clarity of user queries [11, 5]. Users frequently submit short, informal, or underspecified queries, such as "How to prevent falls?" or "Apple support", which omit crucial details like location, time period, product type, or specific intent. These queries often result in the retrieval of irrelevant documents or suboptimal answers. [3, 26].

Research in web search logs indicates that a substantial proportion of queries exhibit ambiguity, contributing to user dissatisfaction and requiring multiple reformulations to achieve satisfactory results. For instance, studies on query classification have shown that ambiguous queries can account for up to 16% of real-world search traffic, based on supervised models trained on labelled data [33].

Recently, Retrieval-Augmented Generation (RAG) systems have become popular in open-domain QA. These systems combine large language models with document retrieval to provide answers that are grounded in real evidence. Therefore, they can reduce the problem of hallucinations compared with pure language models [22]. However, even with RAG, ambiguous queries still create difficulties. When the query is not specific enough, the retrieval step may bring back documents that do not match well. Consequently, the final answer can still lack accuracy or relevance, even though grounding is used.

For this reason, it is useful to detect and resolve ambiguity at the very beginning, before the retrieval process starts. By asking targeted clarifying questions or reformulating the query, the system can obtain refined query parameters [37, 21]. This approach helps improve retrieval precision and overall performance in RAG systems while keeping the interaction simple and efficient for users.

1.2 Problem Statement

User queries in RAG systems often contain lexical, semantic, or contextual ambiguity. This ambiguity can reduce system performance in two ways. First, *retrieval failure* occurs when a single query cannot represent all possible user intentions, leading to missing or irrelevant documents. Second, *generation failure* occurs when the system combines information from different meanings and produces unclear or incorrect answers.

The main problem addressed in this project is *how to design efficient and modular components that can detect ambiguous queries and improve retrieval quality*. The proposed solution is *a clarification module placed before the RAG retrieval process*, which helps resolve ambiguity through query transformation and user clarification.

1.3 Objectives

The primary aim of this project is to design, implement, and assess ClariGen, a modular clarification pipeline that detects query ambiguity, generates clarifying questions, and reformulates queries to enhance downstream RAG performance. Specific objectives include:

1. To develop a lightweight ambiguity detection module using small language models, utilizing zero-shot and few-shot prompting techniques on benchmark datasets such as ClariQ [1] (a conversational clarification benchmark) and AmbigNQ [26] (an ambiguity-focused subset of Natural Questions).

2. To create a clarification question generation module informed by the CLAMBER taxonomy [39], comparing prompting strategies including Ambiguity-Type Chain-of-Thought (AT-CoT) [34], Ambiguity-Type Standard (AT-Standard), and Vanilla prompting (direct question generation without taxonomy guidance).

3. To implement a query reformulation component that integrates user clarifications to produce disambiguated queries, thereby improving retrieval precision and generation quality in RAG pipelines.

4. To conduct qualitative analyses via case studies on real-world queries from aging populations related to fall prevention, assessing the pipeline's applicability in domain-specific scenarios where ambiguity can have practical consequences, such as in health information retrieval.

1.4 Scope

The project focuses on initial, short, and single-turn queries rather than extended multi-turn dialogues. Evaluation utilizes public datasets, such as QULAC [3], ClariQ [1], and AmbigNQ [26], supplemented by curated real-user queries for broader applicability.

The pipeline emphasizes *a single clarification step* followed by user's clarification and reformulation, without pursuing model fine-tuning or full interactive conversational agents. Implementation prioritizes modularity, employing smaller LLMs for detection and larger models for generation to accommodate varying resource constraints.

The final reformulated query is constrained to an established RAG system that focuses specifically on the domain of fall prevention knowledge for older adults.

1.5 Report Structure

This report is organized as follows:

- **Chapter 2: Related Work** surveys existing approaches to query ambiguity detection, clarification generation, and interactive QA systems. It reviews key datasets (e.g., ClariQ, QULAC), taxonomies (e.g., CLAMBER), and LLM-based methods, highlighting gaps in efficiency, reasoning depth, and domain-specific applications that ClariGen addresses.

- **Chapter 3: Methodology** describes the overall architecture of ClariGen, including the dual-model "fast-slow" pipeline, ambiguity taxonomy, Orchestration Layer, and implementation details.

- **Chapter 4: Benchmark Evaluation of ClariGen** experimentally validates the pipeline. It compares zero-shot vs. few-shot ambiguity detection on AmbigNQ, ClariQ, and evaluates clarification question generation strategies (vanilla, AT_Standard, AT_CoT) on ClariQ and QULAC using BERTScore, confirming superior performance of ambiguity-aware Chain-of-Thought prompting and the fast-slow architecture.

- **Chapter 5: Case Study: Mobility Inquiries from Older Adults** presents qualitative analysis on a domain-specific set of fall prevention queries for older adults. It examines ambiguity detection stability and demonstrates interactive resolution through case studies.

- **Chapter 6: Conclusion** summarizes key findings, discusses limitations, and proposes directions for future work.

Appendices, include full prompts, additional results and implementation scripts.

Chapter 2

Related Work

This section reviews the literature on query ambiguity in IR and QA systems, focusing on detection methods, clarification question generation, query reformulation, integration with RAG, and relevant datasets. It contextualizes the ClariGen pipeline by highlighting foundational and recent advancements, drawing from surveys, empirical studies, and benchmarks.

2.1 Query Ambiguity in IR and QA Systems

Ambiguous queries are those that can be interpreted in multiple ways [10]. They often lack sufficient context or contain terms with more than one meaning [31, 14]. Early studies found that about 16% of real search queries are ambiguous [33]. Such ambiguity makes retrieval difficult, because search systems may retrieve irrelevant documents or return generic answers. In conversational or QA systems, vague queries can lead to unhelpful responses like "I don't know" or even hallucinated answers [36]. In domains like medical or aging-related queries, ambiguity is especially risky:

a vague symptom query from an elderly patient could trigger the wrong information. Research has shown that understanding and resolving ambiguity is crucial for effective retrieval and user satisfaction [33, 37].

2.2 Ambiguity Detection Methods

Ambiguity detection is crucial for enabling proactive clarification and resolution [32]. Earlier work [8, 2, 17, 19] typically relies on supervised classifiers trained on labelled datasets, while more recent approaches increasingly leverage LLMs via zero-shot and few-shot prompting [39].

For temporal ambiguity, one method prompts an LLM to generate disambiguated query variants by inserting explicit years, and then compares the answers returned for each variant using efficient search strategies such as skip-lists. This approach achieves an F1 score of 0.699 on TempAmbiQA [30]. In benchmarks such as CLAMBER [39], LLMs are instead prompted to directly classify whether a query is ambiguous, using settings such as zero-shot chain-of-thought prompting or few-shot demonstrations to evaluate how well they identify ambiguity in users' information needs. Challenges persist in semantic and epistemic misalignments, where LLMs struggle with unfamiliar entities or conflicting knowledge [32].

2.3 Clarification Question Generation

Generating clarifying questions resolves ambiguities efficiently. Techniques include rule-based templates, crowdsourcing, and LLM prompting, with taxonomies guiding quality [18]. In information retrieval research, Zamani et al. [37] studied clarifying questions in search and argued that asking such questions "closes the gap" between simple query-response and interactive search. User studies [15] show users welcome a question back when the query is unclear. Systems can generate clarifying questions using templates, heuristics, or learned models. For example, Clarinet [9] uses a LLM conditioned on the retrieval results to generate informative questions. Another approach is diversity-first: generate candidate interpretations or sub-questions, then verify them. The Tree of Clarifications model [16] creates a tree of disambiguated sub-questions via few-shot prompting and retrieval.

2.4 Query Reformulation and RAG Integration

Query reformulation integrates clarifications to disambiguate inputs, enhancing retrieval augmented generation [7]. Surveys on query expansion techniques categorise methods into global, ontology based, and local, feedback based, analysis, using sources like WordNet [4]. In retrieval augmented generation, RQ-RAG refines queries through rewriting, decomposition, and disambiguation, outperforming baselines by 1.9% on the single-turn Question Answering [7] task.

2.5 Datasets

- ClariQ contains roughly 2,000 queries and focuses on conversational clarification, with data sourced from StackExchange domains (see `https://github.com/aliannejadi/ClariQ`).
- QULAC varies in size and focuses on asking clarifying questions when a user's request lacks clarity in open-domain conversational settings (see `https://github.`

com/aliannejadi/qulac).

- AmbigNQ contains 14,042 questions and focuses on capturing multiple ambiguous interpretations, using a subset of Natural Questions (see https://github.com/shmsw25/AmbigQA).

- TempAmbiQA contains 8,162 questions and focuses specifically on temporal ambiguity, with data drawn from ArchivalQA and SituatedQA (see https://github.com/DataScienceUIBK/TempAmbiQA).

Chapter 3

Methodology

3.1 System Overview

The ClariGen system is designed as an end-to-end pipeline that bridges the gap between ambiguous user queries and clear, intent-aligned RAG search requests. The system utilizes a dual-model approach to balance computational efficiency with linguistic reasoning. This section outlines the high-level architecture, the core design philosophy, and the sequential stages of the query lifec ycle.

3.1.1 High-Level Architecture

The ClariGen architecture is composed of four primary layers that work in concert to resolve ambiguity(see Figure 3.1):

- **Interface Layer (Frontend/API)**: Provides both a web-based Streamlit application for demo human interaction and a FastAPI-based RESTful interface for programmatic integration. It manages the user session and handles the

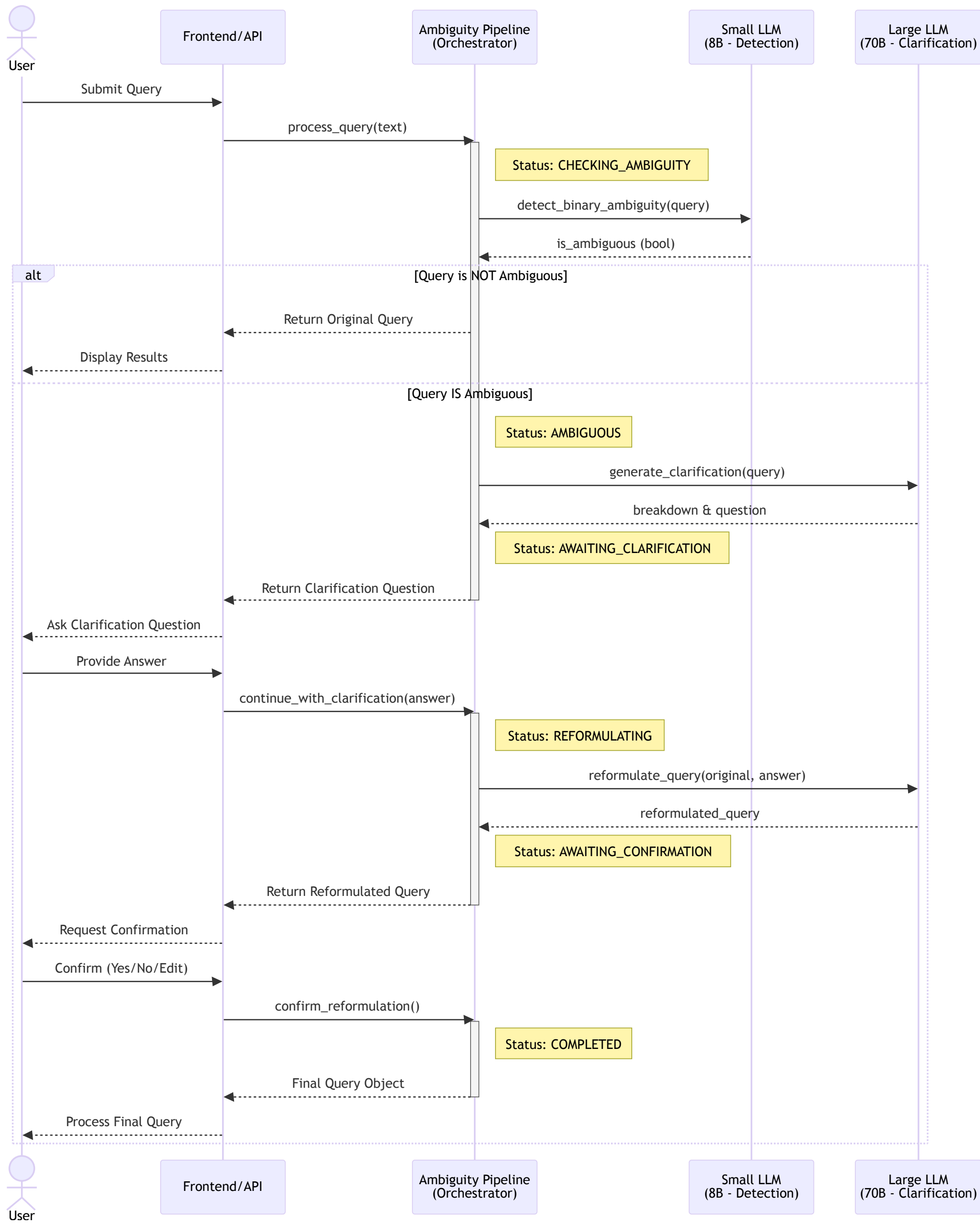

Figure 3.1: ClariGen Sequence Diagram

display of clarifying questions and results.

- **Orchestration Layer (Ambiguity Pipeline)**: The central controller (AmbiguityPipeline) that manages the state transitions of a query. It coordinates the execution flow, ensuring that a query moves logically from detection to resolution while maintaining the context of the conversation.
- **Efficiency Layer (Small LLM Client)**: An interface to a lightweight, high-throughput model (e.g., Llama-3.1-8B). This layer is optimized for low-latency classification tasks.
- **Reasoning Layer (Large LLM Client)**: An interface to a more capable, high-parameter model (e.g., Llama-3.3-70B). This layer is reserved for tasks requiring deep semantic understanding, such as ambiguity categorization and query reformulation.

3.1.2 Design Philosophy: The "Fast-Slow" Strategy

A core design principle of ClariGen is the application of the "Fast-Slow" cognitive paradigm to a software pipeline. Recognizing that many user queries are inherently clear, ClariGen avoids the unnecessary latency and cost of invoking a large model for every request.

- **Gated Detection:** The system uses the 8B model as a "gatekeeper". By performing binary ambiguity detection at this level, ClariGen can process clear queries in milliseconds.
- **Escalated Reasoning:** Only when a query is flagged as ambiguous is the task escalated to the 70B model. This ensures that the system's most expensive

"reasoning cycles" are spent precisely where they are needed: on understanding why a query is unclear and how to fix it.

3.1.3 Pipeline Stages and Query Life cycle

The life cycle of a query within the system is managed through a formal state machine, ensuring robustness and traceability. (see Figure 3.2 for the state diagram):

1. **Initial Detection** (`CHECKING_AMBIGUITY`): The small model performs a binary check. If the query is clear, it skips directly to the final state.

2. **Ambiguity Breakdown** (`AMBIGUOUS`): For flagged queries, the large model analyzes the text to identify specific ambiguity types with CLAMBER taxonomy [39], and generates a targeted clarifying question.

3. **Interaction Loop** (`AWAITING_CLARIFICATION`): The system pauses and requests clarification from the user. This interactive step prevents the system from making incorrect assumptions regarding user intent.

4. **Synthesis** (`REFORMULATING`): Once feedback is received, the large model synthesizes the original query and the user's clarification into a single, unambiguous search query.

5. **Validation & Finalization** (`AWAITING_CONFIRMATION / COMPLETED`): The user provides a final "Yes/No" or edit to the reformulated query before it is finalized for downstream retrieval tasks.

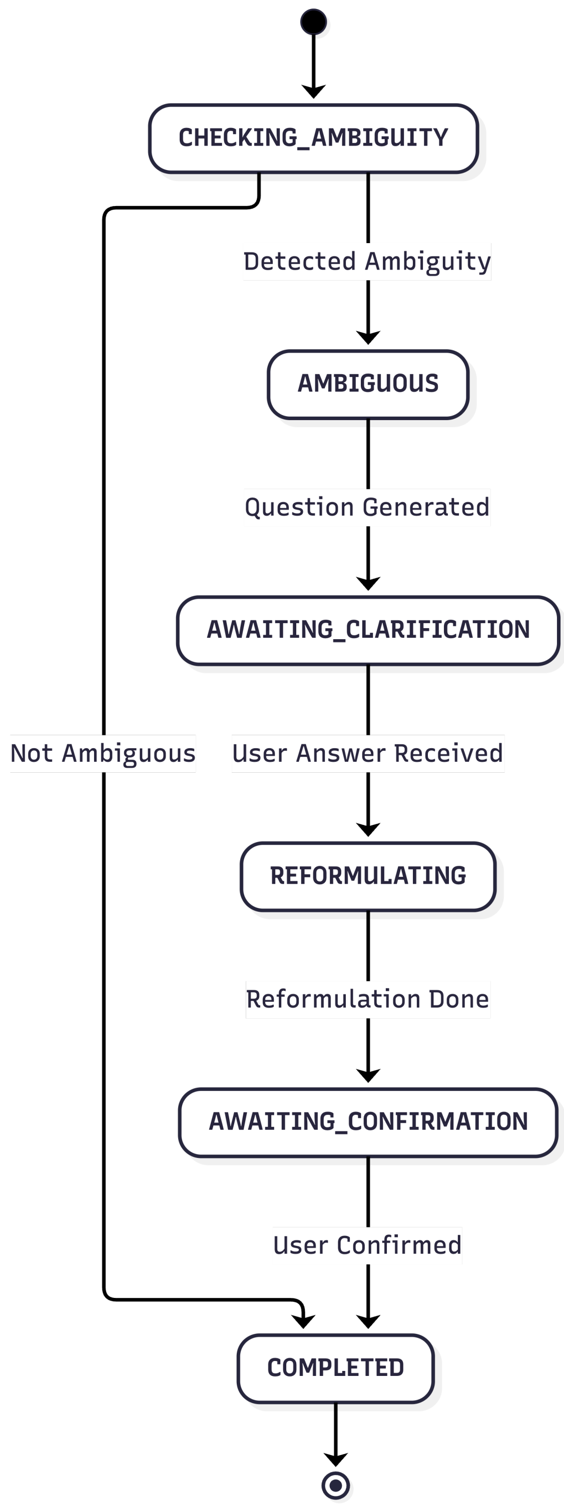

Figure 3.2: ClariGen State Diagram

3.2 Ambiguity Detection Module

The Ambiguity Detection Module serves as the primary gateway of the ClariGen system. Its objective is to efficiently perform a rapid binary classification of the incoming query to determine if further clarification is required. By filtering out unambiguous queries at this early stage, the system maintains high throughput and reduces unnecessary computational overhead on the larger reasoning models.

3.2.1 Objective and Model Selection

The primary goal of this module is to achieve high recall in detecting ambiguity while maintaining a low inference latency. For this task, we selected Llama-3.1-8B-Instruct. The justification for this specific model choice is threefold:

- **Inference Efficiency**: As the first stage in every query's life cycle, this module must be near-instantaneous. The 8B model's smaller parameter count allows for high tokens-per-second throughput, ensuring that clear queries are processed and returned to the user without a perceptible "wait time".

- **Instruction-Following**: Despite its size, the instruction-following model exhibits strong instruction-following capabilities. This is critical for adhering to our strict system prompt and the binary decision rules required for reliable pipeline branching. `IFEval` dataset [40] evaluates instruction-following via programmatically verifiable constraints (e.g., length, format, required keywords). In the Llama 3.1 8B Instruct model card, the reported score is 80.4 [24]. This result suggests strong instruction-following capability for an 8B-parameter model, supporting its use as an efficient backbone for our project.

- **Structured Output Compatibility**: The model's ability to consistently generate valid JSON allows for seamless integration with our Pydantic-based orchestration layer, preventing the common "formatting errors" seen in smaller or less-refined models.

3.2.2 Binary Classification

The detection process is governed by a rigorous definition of ambiguity. The system prompt instructs the model to classify a query as ambiguous if "two or more distinct, reasonable interpretations would lead to materially different answers."

The decision logic provided to the model includes specific rules for complex edge cases:

- **Dominant Interpretation Role:** If one interpretation is clearly dominant and alternatives are unlikely, the query is marked as clear (`False`).
- **Missing Constraints:** Omissions of timeframe (e.g., "in 2023"), location, or version/edition (e.g., "iPhone 15 vs 14") are explicitly treated as ambiguity when the omission could change the final answer.

3.2.3 Prompting Strategies: Zero-Shot vs. Few-Shot

While the system provides infrastructure for both zero-shot and few-shot prompting, our final implementation utilizes the zero-shot strategy. The prompts used are in Appendix B.

During the development and evaluation phase, we observed that few-shot prompting introduced a significant **classification bias towards ambiguity**. By providing the model with multiple examples of ambiguous queries, even though it saw an equal number of unambiguous queries, the model became overly sensitive and frequently flagged clear but complex queries as ambiguous (false positives).

By adopting a zero-shot approach, we rely on the model's inherent semantic understanding of the provided definition. This resulted in a more balanced classification that effectively filters clear queries while still maintaining the sensitivity required to trigger the clarification pipeline for truly uncertain requests.

3.2.4 Structured Output and Validation

To facilitate machine readability, the module utilizes the defined "BinaryDetectionResponse" schema to constrain the model's output. The model is constrained to return a strictly formatted JSON object containing:

- `is_ambiguous`: Whether the query is ambiguous (True) or clear (False)

3.3 Clarification Generation Module

Once a query is identified as ambiguous by the detection layer, it is escalated to the Clarification Generation Module. This module is responsible for the most semantically complex task in the pipeline: **diagnosing the specific nature of the ambiguity and formulating a natural language question** that will elicit the missing information from the user.

3.3.1 The Reasoning Model

While the detection task can be handled by a smaller model, generating a high-quality clarifying question requires a deep understanding of linguistic nuance and world knowledge. For this reason, we employ the Llama-3.3-70B-Instruct-FP8 model [27].

The transition to a 70B parameter model at this stage is a critical architectural decision. The larger model is significantly better at:

- **Semantic Nuance**: Distinguishing between subtle interpretations that a smaller model might conflate.
- **Reasoning Depth**: Following complex instructions to categorize the ambiguity before generating a question.
- **Conversational Fluency**: Generating questions that feel natural and targeted rather than generic or robotic.

3.3.2 Ambiguity Categorization and Definitions

A key feature of the system is its ability to perform simultaneous classification and generation. The model is provided with a formal taxonomy of ambiguity types. (See the Table 3.1). The taxonomy is a modified version of CLAMBER [39], and we unified the original `WHEN, WHO, WHERE AND WHAT` categories to `REFERENCE` for simplicity.

By grounding the model in these definitions, we force it to explicitly identify the “root cause” of the ambiguity. This taxonomy-driven approach ensures that the generated question is anchored in a logical diagnosis of why the original query was unclear.

Ambiguity Type	Explanation	Example
`UNFAMILIAR`	Query contains unfamiliar entities or facts	Find the price of Samsung Chromecast.
`CONTRADICTION`	Query contains self-contradictions	Output 'X' if the sentence contains [category withhold] and 'Y' otherwise. The critic is in the restaurant.>X. The butterfly is in the river.>Y. The boar is in the theatre.>Y.
`LEXICAL`	Query contains terms with multiple meanings	Tell me about the source of Nile.
`SEMANTIC`	Query lacks context, leading to multiple interpretations	When did he land on the moon?
`REFERENCE`	Query contains ambiguous references (pronouns, temporal, spatial, or object references) where multiple interpretations are possible	John told Mark he won the race. (who won?)

Table 3.1: Ambiguity type definitions with explanations and examples.

3.3.3 Prompting Strategies for Clarification

To investigate the impact of reasoning-based prompting on clarification quality, the system supports **three distinct strategies**, derived from the four prompting schemes proposed by Tang et al. [34]. The prompts used are in Appendix B.

- **Vanilla** (`vanilla`): A control group strategy where the model is simply asked to generate a clarifying question without being briefed on the ambiguity taxonomy or specific reasoning steps. This serves as a baseline for measuring the effectiveness of our categorization-led approach.

- **AT-Standard** (`at_standard`): The model is provided with ambiguity definitions and asked to return the identified types along with the clarifying question in a single structured JSON response.

- **AT-CoT** (`at_cot`): This strategy leverages Chain-of-Thought (CoT) reasoning. The model is explicitly instructed to "think step-by-step": first analyzing the query against the ambiguity types, then explaining its reasoning, and finally formulating the question. This introspection often leads to more precise and relevant questions.

3.3.4 Structured Output and Schema Enforcement

To ensure the Orchestrator can reliably handle the model's complex reasoning, we utilize a Pydantic-enforced JSON schema (`ClarificationResponse`). Regardless of the internal reasoning path (CoT or Standard), the model must output a structured object containing:

- `ambiguity_types`: A list of categories identified from the taxonomy
- `reasoning`: A brief textual explanation of the model's plan
- `clarifying_question`: The final question presented to the user

This implementation effectively turns a black-box LLM into a structured reasoning engine that provides both the *answer* (the question) and the *evidence* (the ambiguity types and reasoning) for its decisions.

3.4 Orchestration and State Management

The Orchestration Layer acts as the control plane for ClariGen, coordinating asynchronous communication between user requests and model inference tasks. It uses a finite state machine (FSM) to guide each query through a clear set of steps, so the system behaves predictably. This design helps the system handle non-linear flows, such as branching or retries, in a reliable way.

3.4.1 State-Driven Execution Flow

The core logic of the system is encapsulated in a state-driven execution model (see Figure 3.2). Rather than a procedural script, the pipeline treats a query as a transitioning entity that moves through discrete processing stages. This design decouples the processing logic from the user session, enabling the system to handle long-running transactions (such as waiting for user clarification) without maintaining persistent server connections. The life cycle is governed by the following key state transitions:

- **Binary Filter State (`CHECKING_AMBIGUITY`).** The optimized fast-path entry point where the initial `is_ambiguous` predicate is evaluated.
- **Logic Branching.**
  - *Negative path:* Queries deemed unambiguous transition immediately to a terminal `COMPLETED` state.
  - *Positive path:* Ambiguous queries transition to `AMBIGUOUS`, triggering the clarification workflow.
- **Interaction Suspension (`AWAITING_CLARIFICATION`).** A critical suspension point where the system's execution halts, serializing the current context (ambiguity types, reasoning, generated metadata) while awaiting external input.
- **Synthesis and Convergence (`REFORMULATING`).** Upon receiving the clarification signal, the system resumes execution, synthesizing the original context with the new constraints into a `REFORMULATING` state.

- **Final Verification (`AWAITING_CONFIRMATION`).** A human-in-the-loop validation step that precedes the final `COMPLETED` commit state.

3.4.2 Modular Strategy Design

To ensure the system remains flexible for research, the orchestrator is built with a modular design pattern. This means the core logic that manages the workflow is strictly separated from the specific strategies used to prompt the AI models.

Ideally, the system should support switching between different reasoning methods, such as standard prompting versus "chain-of-thought reasoning" without rewriting the entire system. By abstracting these strategies, the pipeline allows researchers to easily plug in new experimental approaches. This separation is crucial for fair evaluation, as it allows us to test different prompting techniques in isolation while keeping the rest of the system constant.

3.5 Query Reformulation and User Interaction

The final substantive phase of the pipeline is the synthesis of the user's corrective feedback into a functional search query. This stage closes the loop, transforming the initial ambiguous input into a precise, retrieval-ready format.

3.5.1 Contextual Reformulation

Once the user responds to the clarifying question, the system transitions to the `REFORMULATING` state. At this point, the orchestrator aggregates four key pieces of information:

- **Original Query:** The ambiguous starting point.
- **Ambiguity Diagnosis:** The specific types identified (e.g., "Ambiguous Reference").
- **Clarifying Question:** The question generated by the system.
- **User Answer:** The new constraint provided by the user.

These inputs are fed into the `Llama-3.3-70B` model via the `QueryReformulationPrompt`. The model's task is not merely concatenation; it must perform semantic rewrites. For example, if the user answers "the river" to a question about "Amazon," the model must resolve the entity reference and produce "Amazon River history" rather than "Amazon generic history the river."

3.5.2 The "Human-in-the-Loop" Confirmation

Automated reformulation is powerful but not perfect. To mitigate the risk of "hallucinated constraints", where the model infers too much from a brief user answer, ClariGen implements a mandatory confirmation step.

After reformulation, the system presents the new query to the user (e.g., "Did you mean: 'Revenue of Apple Inc. in 2023'? "). The user has two options:

- **Confirm (`Yes`):** The query is finalized and marked `COMPLETED`.
- **Reject and Edit (`No`):** The user can provide an alternative phrasing. This fallback mechanism ensures that the user always retains final agency over the search intent, correcting any misunderstandings introduced by the LLM.

3.5.3 Handling "No Ambiguity" False Positives

A robust system must handle its own errors. In cases where the initial detection module falsely flags a clear query as ambiguous, the user may respond with confusion (e.g., "I don't understand" or "Just search it").

The 70B reformulation model is instructed to handle such cases gracefully. If the user's clarification indicates that the original query was already clear, the model is prompted to revert to the original query text rather than forcing an unnecessary modification.

3.6 Implementation Details

To bridge the gap between the system and the end-user, an interface layer was implemented to transform abstract Python logic into a functional service. The system utilizes a standard client-server architecture, effectively decoupling the heavy inference backend from the lightweight user interface. Both components are hosted on the RAG server (rag.cas.mcmaster.ca), utilizing ports 8370 for the backend and 8371 for the frontend.

3.6.1 API Backend (FastAPI)

The core logic is exposed via a RESTful API built with FastAPI. This choice enables asynchronous request handling, which is especially useful for I/O-bound workloads (e.g., waiting on external services such as LLM calls) because other requests can be processed while one request is awaiting I/O.

The API exposes three primary endpoints that correspond to the life cycle stages of the `AmbiguityPipeline`:

- **`POST /v1/query`:** The entry point. It accepts raw text and returns either a direct result (for clear queries) or a clarification object containing a `context` blob (for ambiguous ones).
- **`POST /v1/clarify`:** Accepts the user's answer along with the serialized `context` blob from the previous turn, triggering the reformulation logic.
- **`POST /v1/confirm`:** Finalizes the transaction by accepting user validation, returning the definitive query string ready for search execution.

This API design allows ClariGen to be easily integrated into larger search platforms independent of any specific frontend technology.

3.6.2 Interactive Frontend

For demonstration and evaluation purposes, we developed an interactive web application using Streamlit (http://rag.cas.mcmaster.ca:8371/). This frontend acts as a reference implementation of the API client, visualizing the entire "ambiguity resolution loop".

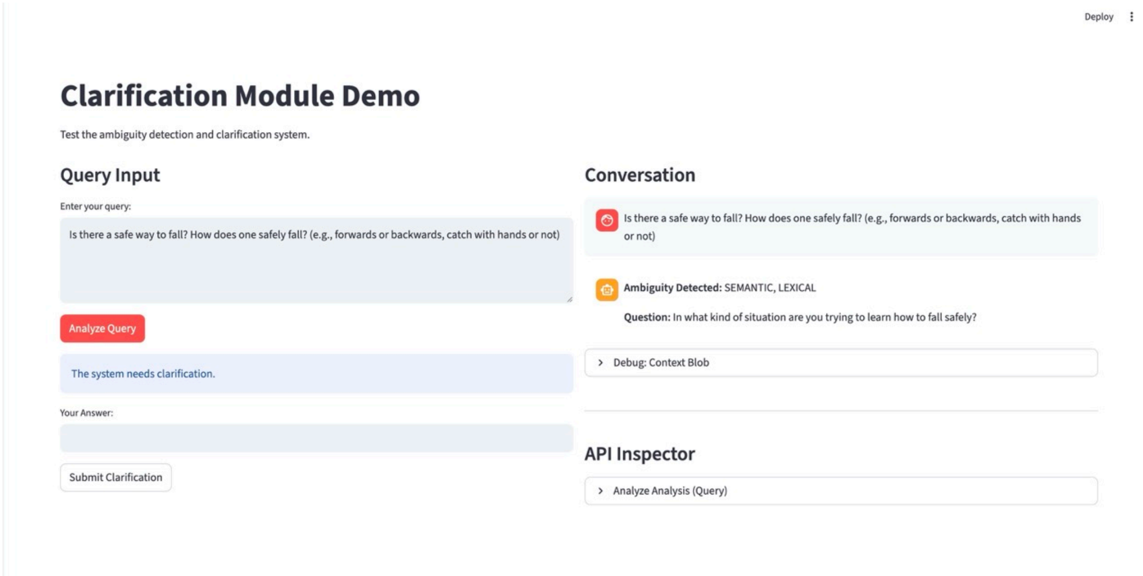

Figure 3.3: Binary classification and detection flow

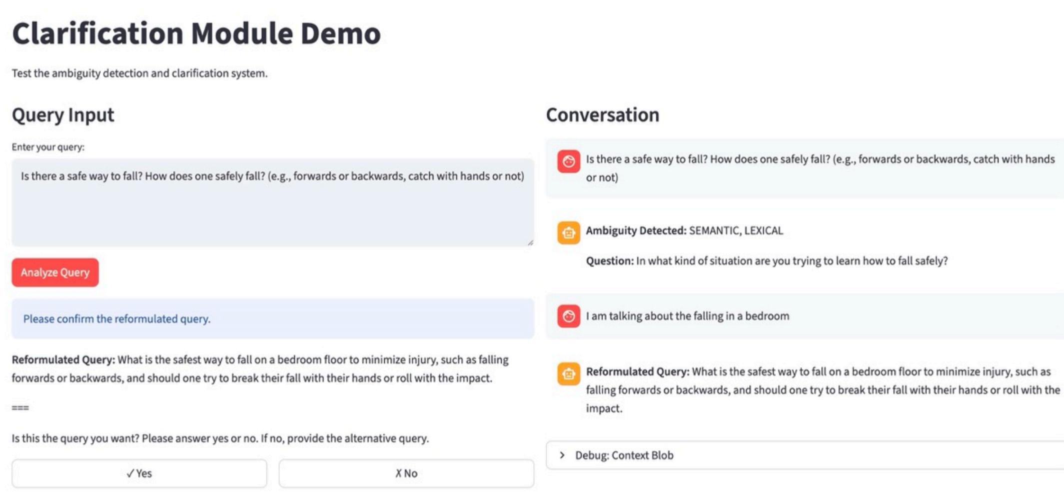

Figure 3.4: Clarification and reformulation flow

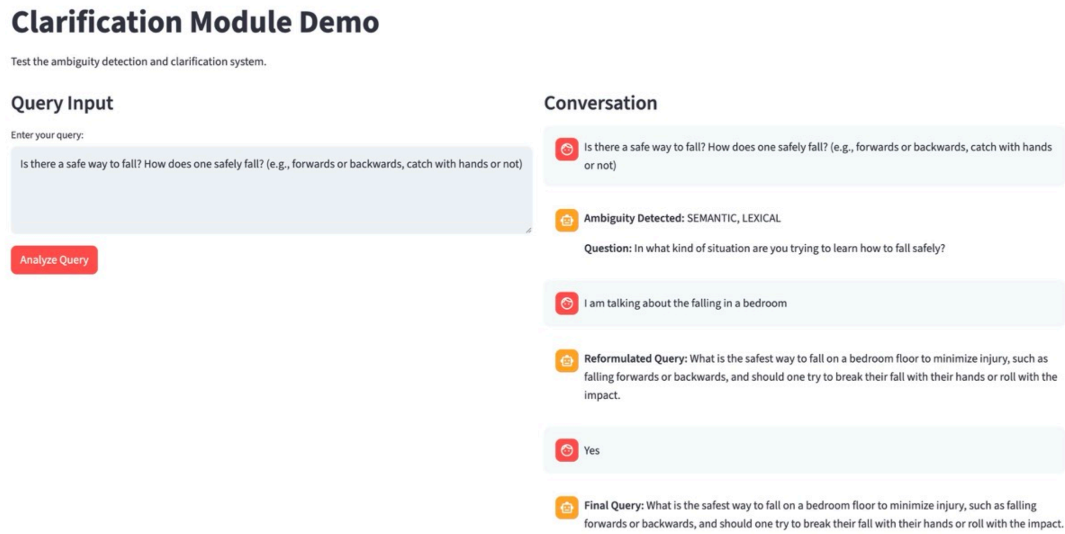

Figure 3.5: Reformulation and confirmation flow

To ensure the system meets latency requirements for real-time interaction, a dedicated model-serving layer was implemented to bypass the inefficiencies of standard sequential inference pipelines. The infrastructure is built on `vLLM`, a high-throughput and memory-efficient engine for LLM serving, which enables effective deployment of our multi-stage architecture.

3.6.3 Dual-Model Deployment Architecture

The system deploys two distinct model servers concurrently so that the heavy computational load of reasoning does not impact the speed of detection. As illustrated in the deployment script (`serve_models.sh`), resources are segregated by task complexity:

Fast Server (Port 8368).

- Model: `meta-llama/Llama-3.1-8B-Instruct`

- Hardware allocation: Dedicated single GPU
- Role: Optimized for low-latency queries. This server exclusively handles binary ambiguity detection invoked by `SmallModelClient`, ensuring that the gate-keeping phase adds negligible overhead to the user experience.

Reasoning Server (Port 8369).

- Model: `nvidia/Llama-3.3-70B-Instruct-FP8`
- Hardware allocation: Distributed across two GPUs using tensor parallelism
- Role: Handles deep semantic tasks, including ambiguity classification, clarification question generation, and query reformulation.

3.6.4 Key Technical Optimizations

Serving a 70 billion parameter model in an interactive loop introduces significant systems challenges. We addressed these constraints using the following techniques:

FP8 Quantization We deployed the FP8 (8-bit floating point) quantized variant of Llama-3.3-70B. This reduced the VRAM footprint by approximately 50% relative to FP16 deployments, allowing the model to fit within available hardware without degrading reasoning quality.

Tensor Parallelism To mitigate memory bottlenecks and improve throughput, the 70B model is served with tensor parallel size 2 (i.e., `--tensor-parallel-size 2`). This shards weight matrices across GPUs, parallelizes matrix multiplications, and increases token generation speed.

PagedAttention We utilize `vLLM`'s PagedAttention to dynamically manage KV-cache memory. This improves concurrency and enables multiple simultaneous clarification dialogues without KV-cache fragmentation failures.

3.6.5 Summary

This chapter detailed the implementation of ClariGen, a framework for interactive ambiguity resolution. We presented a dual-model architecture that balances efficiency with reasoning depth: a lightweight 8B model filters clear queries ("Fast Path") while a 70B model handles complex ambiguity diagnosis and query reformulation ("Reasoning Path").

The system relies on a stateless Orchestration Layer to manage the multi-turn query life cycle (Detection → Clarification → Reformulation → Confirmation). Supported by a robust FastAPI backend and vLLM-based serving infrastructure utilizing FP8 quantization, this design provides a scalable foundation for evaluating the efficacy of LLM-driven clarification in information retrieval.

Chapter 4

Benchmark Evaluations

4.1 Ambiguity Detection Evaluation

4.1.1 Introduction

This chapter evaluates instruction-tuned LLMs using zero-shot and few-shot prompting for binary ambiguity classification ("ambiguous" vs. "clear"), bypassing the need for fine-tuning or labelled data. This lightweight approach supports deployable clarification modules in the system.

4.1.2 Methodology

This section details the prompt-based evaluation, output parsing, and metric computation for ambiguity detection. Llama-3.1-8B-Instruct was selected as the baseline model due to its balanced performance and efficiency [13]. Queries were processed in batches, with models generating "yes" (ambiguous) or "no" (clear) responses; invalid outputs defaulted to "no" for binary labelling.

4.1.3 Experimental Setup

To evaluate zero/few-shot ambiguity detection across model scales and prompting strategies, we ran Python 3.10 with vLLM [20], which uses paged attention to improve inference throughput for large-model batching. To ensure fair and reproducible comparisons, we fixed inference settings: max new tokens = 10 (binary output), temperature = 0.0, and seed = 42, following common LLM classification practice. Experiments used four NVIDIA H100 PCIe GPUs (80 GB each), driver 580.105.08, and CUDA 13.0. We used one GPU for 1B – 8B models (batch = 64 – 128) and 4-GPU tensor parallelism for 27B – 70B (batch = 32). GPU memory utilization was capped at 50% to avoid OOM errors. For Llama-3.3-70B, NVIDIA FP8 quantization reduced memory (~ 35 GB vs. 140 GB FP16) with $< 1\%$ accuracy loss [25], enabling large-scale evaluation (e.g., ~ 2000 AmbigNQ queries).

Prompt Design

Prompts prioritized clarity and task alignment for binary classification. Zero-shot prompts positioned the model as an "expert query ambiguity classifier," followed by the query alone. Few-shot prompts incorporated CLAMBER taxonomy definitions (e.g., unfamiliar, contradictory) and five unambiguous examples to enable in-context learning [39, 6]. Full prompts appear in Appendix A, with lengths optimized to suit smaller models.

Datasets

Evaluation of question ambiguity was conducted using two public QA datasets, with labels standardized to a binary scheme (0 for unambiguous and 1 for ambiguous) to ensure consistency with prior zero-shot evaluations [23]. All queries were lowercased, deduplicated, and exported to TSV format for uniform processing and comparability. **AmbigNQ** builds on Natural Questions and contains 14,042 questions annotated for multiple plausible answers. After preprocessing, our working subset (n=2,002) includes 830 unambiguous (41.46%) and 1,172 ambiguous (58.54%) instances. This mild class imbalance probes models' robustness to lexical and syntactic ambiguity in open-domain settings. **ClariQ** extends QULAC with 299 conversational queries labeled by clarification need on a 1–4 scale [2]. Following prior work, level 1 is mapped to unambiguous (0), while levels 2–4 are mapped to ambiguous (1). Merging all splits yields 37 unambiguous (12.37%) and 262 ambiguous (87.63%) cases. This pronounced imbalance highlights underspecification in dialogue-based QA and makes ClariQ apt for evaluating clarification strategies [23].

binary_label	**count**	**percent**
0 (unambiguous)	830	41.46
1 (ambiguous)	1172	58.54

Table 4.1: Label distribution in the preprocessed AmbigNQ subset

binary_label	**count**	**percent**
0 (unambiguous)	37	12.37
1 (ambiguous)	262	87.63

Table 4.2: Label distribution in the preprocessed ClariQ dataset

Models

A curated set of instruction-tuned LLMs spanning 1B–70B parameters have been assesed to examine the effects of scale and architecture on ambiguity detection and clarification. Instruction-tuned variants were prioritized for classification performance.

Model Series	Reference
Phi-3 (Mini 128k Instruct)	[12]
Llama-3 Series: Llama-3.1-8B, Llama-3.2-1B, Llama-3.2-3B, Llama-3.3-70B	[13]
Gemma-3 Series: Gemma-3-1B, Gemma-3-4B, Gemma-3-27B	[35]
GPT-4.1 Series: GPT-4.1, GPT-4.1-mini, GPT-4.1-nano	[28]

Table 4.3: Model series and references.

Open-weight models were included to ensure reproducibility, with Llama-3.1-8B serving as a latency-conscious baseline suitable for real-time deployment. GPT-4.1 provides a reference point to contextualize open-model results.

Classification Metrics

This section defines the prevalence-weighted metrics used for ambiguity detection. Because the datasets are imbalanced (AmbigNQ: 58.54% ambiguous; ClariQ: 87.63% ambiguous), we report weighted scores that account for class proportions n_c/N, where $c \in \{0 = \text{unambiguous}, 1 = \text{ambiguous}\}$, n_c is the number of instances in class c, and N is the total number of instances. All metrics use scikit-learn implementations [29].

$$\mathrm{P}_{\text{weighted}} = \sum_{c\in\{0,1\}} \left(\frac{n_c}{N} \cdot \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c} \right) \tag{4.1.1}$$

Weighted precision balances performance across classes and reduces unnecessary clarifying questions.

$$\mathrm{R}_{\text{weighted}} = \sum_{c\in\{0,1\}} \left(\frac{n_c}{N} \cdot \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FN}_c} \right) \tag{4.1.2}$$

Weighted recall emphasizes correctly detecting ambiguous queries, which is important for conversational QA safety.

$$\mathrm{F1}_{\mathrm{weighted}} = \sum_{c \in \{0,1\}} \left(\frac{n_c}{N} \cdot 2 \cdot \frac{\mathrm{Precision}_c \cdot \mathrm{Recall}_c}{\mathrm{Precision}_c + \mathrm{Recall}_c} \right) \tag{4.1.3}$$

Weighted F1 is the main metric, as it balances precision and recall under class imbalance.

4.1.4 Results

	ClariQ (%)				**AmbigNQ (%)**			
Methods	**P**	**R**	**F1**	**Latency (ms)**	**P**	**R**	**F1**	**Latency (ms)**
Llama-3.1-8B (zero-shot)	81.45	68.23	73.14	2.1	52.66	53.95	52.98	1.7
Gemma-3-1B (few-shot)	76.78	87.63	81.85	1.5	34.27	**58.54**	43.23	0.8
Gemma-3-1B (zero-shot)	76.75	87.29	81.68	1.1	55.01	**58.54**	43.41	0.6
Gemma-3-27B (few-shot)	83.84	66.89	72.40	7.3	50.95	47.80	47.65	8.4
Gemma-3-27B (zero-shot)	80.60	77.59	78.98	8.3	52.99	53.60	53.22	9.4
Gemma-3-4B (few-shot)	76.75	87.29	81.68	2.2	48.07	58.39	43.43	1.3
Gemma-3-4B (zero-shot)	76.78	87.63	81.85	1.1	34.27	**58.54**	43.23	1.1
Llama-3.1-8B (few-shot)	84.95	50.17	57.61	3.4	50.55	47.55	47.48	2.6
Llama-3.2-1B (few-shot)	76.67	86.62	81.34	0.7	55.10	**58.54**	44.94	0.8
Llama-3.2-1B (zero-shot)	77.89	61.20	67.55	0.8	52.42	57.94	46.22	0.6
Llama-3.2-3B (few-shot)	83.18	68.23	73.36	1.1	49.91	47.25	47.35	1.0
Llama-3.2-3B (zero-shot)	**89.41**	**87.96**	**82.64**	1.0	52.70	58.49	43.57	0.8
Llama-3.3-70B (few-shot)	84.81	69.90	74.79	18.5	53.75	51.65	51.94	18.5
Llama-3.3-70B (zero-shot)	83.16	72.58	76.51	17.5	53.92	54.35	**54.09**	16.8
Phi-3 (few-shot)	82.01	74.92	77.83	1.4	52.73	56.49	51.00	1.6
Phi-3 (zero-shot)	76.78	87.63	81.85	1.9	34.26	58.49	43.21	1.3
gpt-4.1 (few-shot)	84.75	48.83	56.29	-	56.68	51.45	50.39	-
gpt-4.1 (zero-shot)	84.41	67.56	72.96	-	**61.40**	54.20	52.57	-
gpt-4.1-mini (few-shot)	85.67	63.21	69.49	-	56.04	47.90	43.83	-
gpt-4.1-mini (zero-shot)	84.61	71.24	75.77	-	58.01	50.90	48.64	-
gpt-4.1-nano (few-shot)	81.79	31.77	36.67	-	53.89	45.05	38.11	-
gpt-4.1-nano (zero-shot)	81.33	58.19	65.32	-	54.50	53.65	53.93	-

Table 4.4: Classification performance for ambiguity detection across datasets.

Precision (P), recall (R), F1, and latency across ClariQ and AmbigNQ were reported. Metrics are averaged over all instances in each dataset. Latency is measured per query in milliseconds for open-weight models; API models are reported without latency due to provider variability.

ClariQ yields higher F1 scores (36.67%–82.64%) than AmbigNQ (38.11%–54.09%). This gap likely reflects ClariQ's focused conversational ambiguities versus AmbigNQ's broader, open-domain ambiguity types. Zero-shot prompting often matches or exceeds few-shot, particularly for smaller models (e.g., Llama-3.2-3B zero-shot on ClariQ). Latencies are consistently low for open-weight models (0.6–18.5 ms), indicating suitability for interactive deployment. Taken together, these findings support prompt-based ambiguity detection for real-time clarification. However, the modest F1 on AmbigNQ and the mixed few-shot gains suggest that few-shot prompting does not reliably outperform zero-shot.

4.1.5 Discussion

The results of this study indicate that **zero-shot prompting is sufficient and often more effective than few-shot prompting for detecting ambiguity**. Although it is commonly believed that adding examples helps models understand tasks better, our data shows that these extra examples can sometimes introduce noise or bias. This suggests that modern instruction-tuned models are already capable enough to understand the concept of ambiguity directly. Therefore, using zero-shot prompts is not only simpler but also more reliable for practical applications.

Furthermore, it is clear that **model size does not always lead to better performance**. In many tests, medium-sized models that were specifically tuned for instructions performed just as well as, or even better than, much larger models with higher parameter counts. This proves that the way a model is aligned with a task is more important than the total number of parameters. For developers, this is an important finding because it means they can use smaller, faster models to achieve high-quality results without requiring expensive computational resources.

Finally, **while this prompt-based method is ready for real-time use, its performance depends heavily on the dataset**. Open-weight models are fast enough for real-time applications. However, their accuracy changes depending on the type of ambiguity. For instance, the models found it easier to detect conversational ambiguity in the ClariQ dataset but struggled more with the complex, open-domain questions in AmbigNQ. This shows that while the technology is technically ready for deployment, further research is needed to improve how models handle different kinds of informational gaps.

4.2 Clarification Question Generation Evaluation

4.2.1 Introduction

The primary objective of this experiment is to evaluate the efficacy of varying prompting strategies in enhancing the quality of generated clarification questions. Specifically, this experiment investigates whether explicitly defining ambiguity types and incorporating Chain-of-Thought (CoT) reasoning can foster higher semantic similarity to human-generated references, inspired by recent work on ambiguity-aware prompting for clarification generation [34]. It is hypothesized that (1) providing the model with a structured taxonomy of ambiguity will outperform generic zero-shot prompting, and that (2) enforcing a step-by-step reasoning process will yield the optimal performance.

4.2.2 Methodology

This section describes the core prompting strategies and the evaluation metric used to assess clarification question quality.

Prompting Strategies

The experiment leverages an instruction-tuned LLM as a clarification question generator. Three prompting strategies (Vanilla, AT_Stadnard and AT_CoT) defined in the prior chapter are compared to evaluate the impact of explicit ambiguity awareness.

Evaluation Metric

Clarification question quality is assessed using a BERTScore-based max-matching approach [38], following the methodology in Tang et al. [34]. For each query, the generated question is compared against multiple human-authored reference questions. A similarity matrix is constructed using BERTScore F1 scores, and the per-query score S_{query} is the maximum value in this matrix:

$$S_{query} = \max(M) \tag{4.2.1}$$

The final system-level score is the mean across all queries:

$$S_{\text{final}} = \frac{1}{N} \sum_{k=1}^{N} S^{(k)} \tag{4.2.2}$$

This metric prioritizes semantic similarity, making it suitable for evaluating diverse yet equivalent clarifications.

4.2.3 Experimental Setup

This section details the datasets and implementation used for empirical validation.

Datasets

Evaluation is conducted on two public datasets with human-annotated clarification questions:

Dataset	Purpose	# Queries	# Clarification Questions
ClariQ [1]	Ambiguous queries for clarification need	62 (test)	4,499
QULAC [3]	Conversational search (TREC facets)	198	10,277

Table 4.5: Summary of datasets used in this work.

Model and Inference

Experiments use an instruction-tuned large language model (Llama-3.3-70B-Instruct-FP8). Inference settings include structured JSON enforcement via prompting.

4.2.4 Results

Table 4.6 summarizes the quantitative performance of the three prompting strategies. There is a clear upward trend in BERTScore F1 as the prompts become more structured. The Vanilla baseline performs the worst (0.9019 for ClariQ; 0.8960 for QULAC), likely due to a lack of specific guidance on handling ambiguity. In contrast, the AT_Standard method shows significant improvements, reaching 0.9174 and 0.9080 respectively. Finally, AT_CoT achieves the highest scores across both datasets. While the gains over AT_Standard are marginal, they are consistent, suggesting that step-by-step reasoning helps the model capture finer linguistic nuances.

Method	**ClariQ**	**QULAC**
Vanilla	0.9019	0.8960
AT_Standard	0.9174	0.9080
AT_CoT	**0.9187**	**0.9093**

Table 4.6: Comparison of Prompting Strategies (Mean BERTScore F1)

4.2.5 Discussion

The results validate our hypothesis that ambiguity-type knowledge improves clarification quality. Crucially, pairwise statistical tests confirm that these performance gains are robust and not the result of random variation.

Key Findings and Interpretation

The fact that **AT_Standard consistently outperforms the Vanilla baseline** suggests that explicit guidance is essential for semantic alignment. By following a taxonomy, the model avoids generating generic questions and instead identifies specific missing information. Statistical significance confirms that this structural knowledge is a reliable driver of performance.

Furthermore, **AT_CoT provides additional benefits by enhancing faithfulness**. Although the numerical increase over AT_Standard is small, it is statistically validated. This indicates that the reasoning process effectively filters out irrelevant details. Consequently, the performance hierarchy (**Vanilla** $<$ **AT_Standard** $<$ **AT_CoT**) remains stable across different datasets. This consistency highlights the generalizability of these strategies, proving they are dependable for various types of linguistic ambiguity.

Chapter 5

Case Study: Mobility Inquiries from Older Adults

As ClariGen will be integrated into a RAG system that delivers fall-prevention information for older adults, this chapter presents a qualitative evaluation of real-world user queries and demonstrates how systematic clarification can improve query specificity and retrieval performance.

5.1 Dataset

Twelve real-world queries from older adults were used for qualitative evaluation, representing diverse concerns such as mobility limitations, chronic illness, and medication safety.

1. How can individuals with Parkinson's disease reduce their fall risk, and are there specific strategies that work best for them?
2. How can individuals with Arthritis (specifically Osteoarthritis) reduce their fall risk, and are

there specific strategies that work best for them?

3. How do common medications, such as those for high blood pressure or pain management, contribute to fall risk, and what precautions can seniors take?
4. What are the most effective exercises to improve balance and prevent falls, especially for seniors with limited mobility or strength? What about Tai Chi?
5. Is there a safe way to fall? How does one safely fall? (e.g., forwards or backwards, catch with hands or not)
6. Are there any vitamins recommended for strengthening bones? (e.g., vitamin D)
7. What are some simple and cost-effective home modifications to make living spaces safer for seniors at risk of falling?
8. What role do wearable devices, like fall-detection systems or smartwatches, play in preventing or managing falls for seniors? Are they effective tools?
9. After a fall and a serious injury like a broken hip, how can seniors regain mobility and prevent long-term inactivity that can further increase fall risk?
10. What can be done for fear of falling?
11. What/where are some local Ontario-based fall prevention programs?
12. How can caregivers or healthcare providers encourage older adults who are fearful of falling to start and stick with a regular exercise routine to improve balance and strength?

5.2 Ambiguity Detection

To identify potentially ambiguous queries, stability tests were conducted using the Llama-3.1-8B-Instruct model. Each query was processed through the detection pipeline 100 times to measure output consistency. Results for zero-shot and few-shot prompting are compared in Table 5.1.

ID	Zero-Shot			Few-Shot		
	Ambiguous %	**Clear %**	**Total**	**Ambiguous %**	**Clear %**	**Total**
1	0.0%	100.0%	100	100.0%	0.0%	100
2	0.0%	100.0%	100	100.0%	0.0%	100
3	0.0%	100.0%	100	100.0%	0.0%	100
4	100.0%	0.0%	100	100.0%	0.0%	100
5	100.0%	0.0%	100	100.0%	0.0%	100
6	13.0%	87.0%	100	100.0%	0.0%	100
7	0.0%	100.0%	100	100.0%	0.0%	100
8	0.0%	100.0%	100	100.0%	0.0%	100
9	0.0%	100.0%	100	100.0%	0.0%	100
10	100.0%	0.0%	100	100.0%	0.0%	100
11	8.0%	92.0%	100	100.0%	0.0%	100
12	0.0%	100.0%	100	100.0%	0.0%	100

Table 5.1: Comparison of zero-shot and few-shot stability results for Llama-3.1-8B-Instruct

The few-shot approach exhibited strong bias toward classifying all queries as ambiguous. In contrast, zero-shot prompting produced a more balanced distribution, consistently identifying IDs 4, 5, and 10 as ambiguous. Accordingly, zero-shot prompting was selected for subsequent stages.

5.3 Clarification and Queries Reformulation

This section examines the three queries consistently classified as ambiguous under zero-shot conditions (IDs 4, 5, 10). For each, we present the clarification interaction and assess its impact on query specificity and downstream retrieval quality in the RAG system.

5.3.1 Query ID 4: Exercises for Balance and Fall Prevention

Clarification Conversation

USER: What are the most effective exercises to improve balance and prevent falls, especially for seniors with limited mobility or strength? What about Tai Chi?

SYSTEM: Ambiguity Detected: LEXICAL, SEMANTIC

SYSTEM: Are you looking for exercises that can be done at home or in a supervised setting?

USER: I'm looking for exercises that can be done at home...

SYSTEM: Reformulated Query: What are the most effective at-home exercises to improve balance and prevent falls for seniors with limited mobility or strength, and can exercises like Tai Chi be safely and effectively practiced at home for this purpose?

Clarification Summary

For ID4, the original query about effective exercises for seniors is refined by a clarifying question on the setting, like home or supervised, which led to a reformulated query that includes at-home feasibility and how Tai Chi can apply, thereby improving practicality and relevance.

5.3.2 Query ID 5: Techniques for Safe Falling

USER: Is there a safe way to fall? How does one safely fall? (e.g., forwards or backwards, catch with hands or not)

SYSTEM: Ambiguity Detected: SEMANTIC, LEXICAL

SYSTEM: What kind of situation are you thinking of when you're considering falling?

USER: I am not sure.... probably general situation like walking...?...

SYSTEM: Reformulated Query: What are the safest techniques for falling while walking, such as the best direction to fall and whether to use your hands to break your fall, in order to minimize injury?

Clarification Summary

For ID5, the ambiguous query regarding safe falling techniques was clarified by asking about the specific situation. This resulted in a new, focused query about how to fall while walking in a way that minimizes injury.

5.3.3 Query ID 10: Strategies for Fear of Falling

USER: What can be done for fear of falling?

SYSTEM: Ambiguity Detected: SEMANTIC, LEXICAL

SYSTEM: Are you asking about treating a fear of falling in general, or is this related to a specific situation, like a medical condition or an activity?

USER: no specific situation, but just a fear of falling in general..

SYSTEM: Reformulated Query: What are some effective treatments or strategies for overcoming a general fear of falling?

Clarification Summary

For ID10, the general question about fear of falling was clarified by separating overall fear from fear in specific situations. This facilitated the precise reformulation of the query, targeting the right treatments and coping strategies. Overall, this clarification reduced confusion in the query and is expected to improve how well the RAG system retrieves relevant information.

5.4 Retrieval Result Comparison

This section compares the retrieval results of ambiguous and clarified queries for the three queries above. It explains how clearer queries improve the relevance, completeness, and usefulness of system responses, and shows why reducing ambiguity is important for better retrieval in RAG systems.

5.4.1 Query ID 4: Exercises for Balance and Fall Prevention

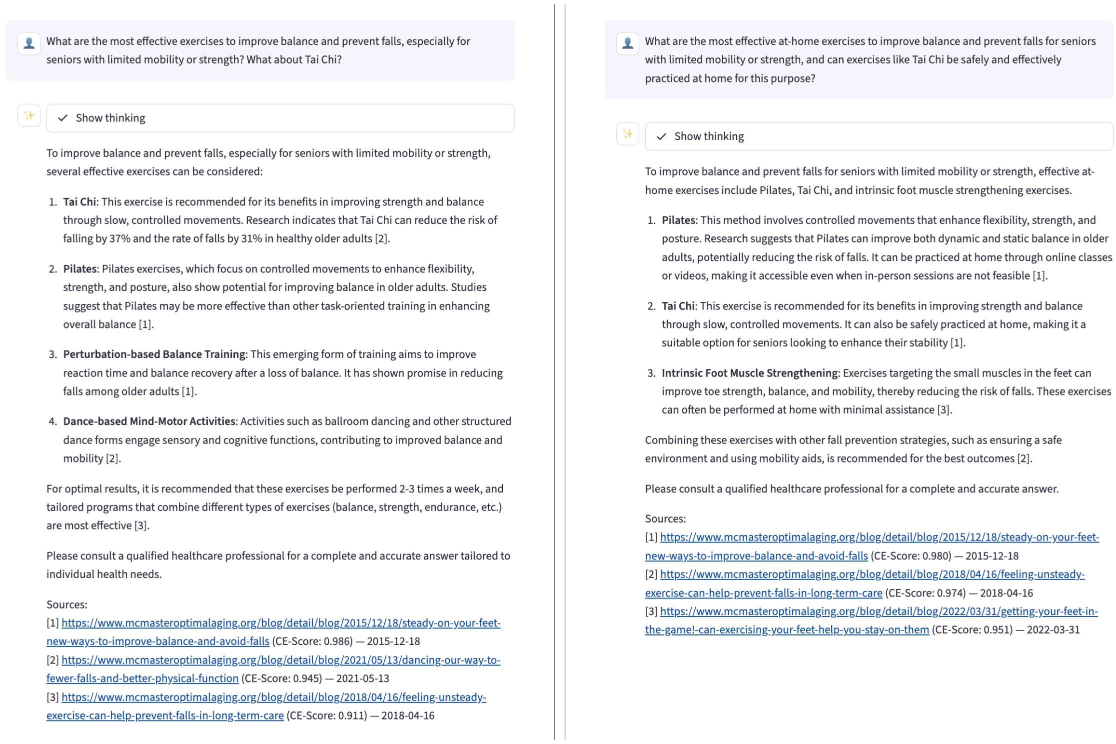

Figure 5.1: Grounding results for clear and unambiguous queries for ID4

The two responses to queries about exercises for improving balance and preventing falls in seniors show clear differences due to query ambiguity. The left one answers a broad question on the most effective exercises for seniors with limited mobility, including Tai Chi. It lists several options like Tai Chi, Pilates, perturbation-based training and dance activities. For example, it mentions Tai Chi can reduce fall risk by 37%, which is supported by research. However, some suggestions such as dance or perturbation training may not be suitable because they need equipment or group classes, making them less practical for people with limited strength.

In contrast, the right response is for a more clarified query that specifies at-home exercises which are safe and can be done easily at home. It focuses on Pilates, Tai

Chi and foot muscle strengthening, all of which require minimal assistance or can use online videos. This makes the answer more targeted and helpful for the user's needs.

Overall, the clarified query leads to better retrieval results with higher relevance and practicality. It avoids irrelevant options and better addresses constraints like limited mobility. This shows how reducing ambiguity can improve the quality of answers in QA systems.

5.4.2 Query ID 5: Techniques for Safe Falling

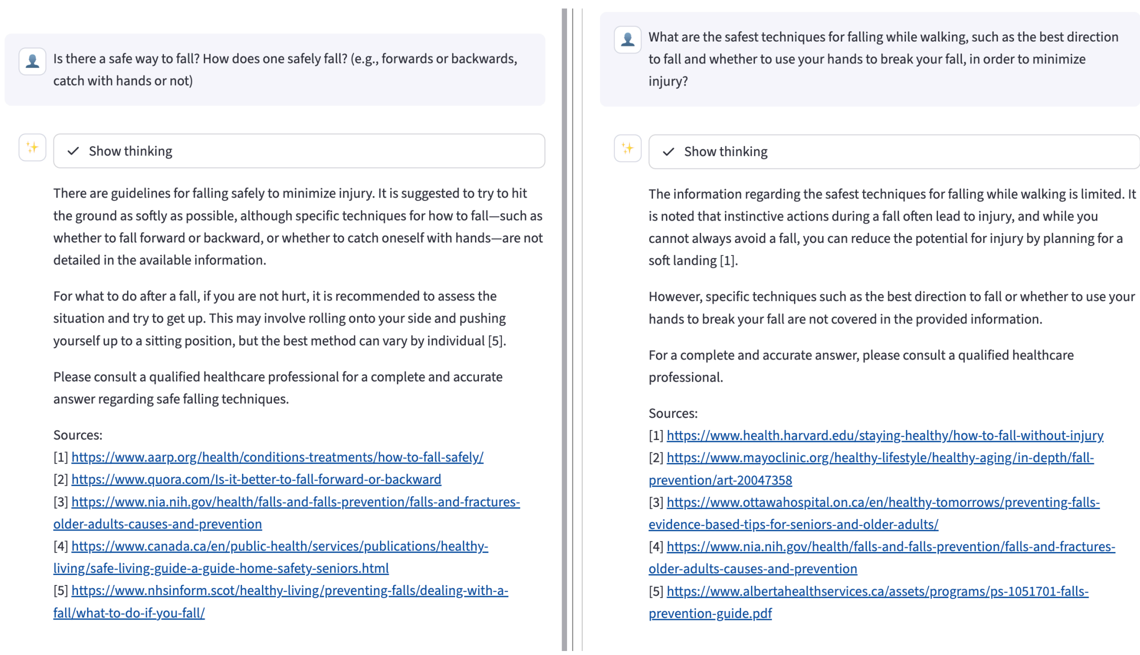

Figure 5.2: Grounding results for clear and unambiguous queries for ID5

The two answers from the RAG system deal with questions about safe ways to fall and reduce injury. The left one is for a broad and ambiguous query asking if there is a safe way to fall, like forward or backward, and whether to catch with hands. It says there are general guidelines to hit the ground softly, but no detailed techniques for direction or using hands are given. For example, it suggests rolling to the side to get up after falling, but advises consulting a professional for accurate safe falling advice.

In contrast, the right response answers a more clarified query that focuses on the safest techniques for falling while walking, such as best direction and using hands to break the fall. However, it still says information is limited and instinctive actions often cause injury. It mentions planning for soft landing but no specific details on direction or hands are provided.

Overall, both responses are cautious and avoid giving direct techniques, probably for safety reasons. The clarified query gets a slightly more targeted answer about walking falls, but neither provides practical advice like tucking chin or bending arms. This shows that clarification helps a bit, but health queries need careful handling to prevent wrong information.

5.4.3 Query ID 10: Strategies for Fear of Falling

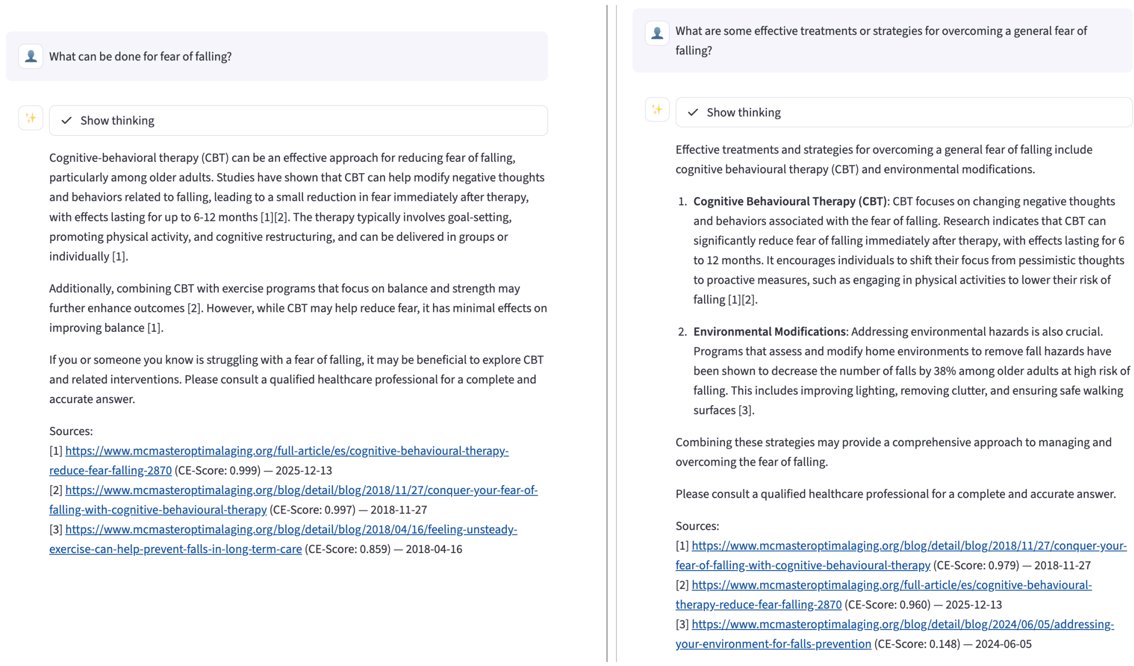

Figure 5.3: Grounding results for clear and unambiguous queries for ID10

The two responses discuss ways to handle fear of falling, but they come from different queries. The left one answers a simple and ambiguous question: ”What can be done for fear of falling?” It mainly focuses on Cognitive-Behavioural Therapy (CBT). It explains that CBT can change negative thoughts and behaviours, and studies show it reduces fear for up to 6-12 months. It also mentions combining CBT with balance and strength exercises can help more, though CBT alone has little effect on actual balance.

In contrast, the right response is for a clearer query asking for effective treatments or strategies to overcome a general fear of falling. It gives a more comprehensive answer by including two main ideas: CBT, which is explained in a similar way using the same evidence on reducing fear, and environmental modifications. For example, removing home hazards like clutter and improving lighting can lower incidence of falls by 38% in high-risk older adults.

Overall, the clarified query leads to a better and more balanced response that covers both psychological and practical strategies. This shows how adding details to the question helps the QA system provide more useful and complete information.

5.4.4 Summary

From the three query pairs, we can see that query ambiguity affects the quality of retrieval results in RAG systems. The ambiguous queries often lead to broad or incomplete answers, such as including impractical exercises or missing key strategies. In contrast, clarified queries produce more relevant, targeted and comprehensive responses that better match user needs, like focusing on at-home options or adding environmental modifications. Overall, these examples show that reducing ambiguity through clarification can improve relevance, practicality and completeness in QA systems, especially for health-related topics. This highlights the importance of developing an effective clarification module to enhance QA performance.

Chapter 6

Conclusion

In conclusion, this report has presented ClariGen, a multi-stage pipeline designed to detect and resolve ambiguous user queries in QA systems, particularly for RAG applications. The system uses a dual-model approach: a lightweight, instruction-tuned model for efficient binary ambiguity detection, and a larger model for deeper reasoning and targeted clarification question generation, guided by a modified CLAMBER taxonomy and ambiguity-aware CoT prompting. Benchmark evaluations on datasets such as ClariQ and QULAC showed strong performance, with zero-shot detection proving reliable and CoT-based clarification achieving high semantic similarity. Case studies on fall prevention queries for older adults further demonstrated practical benefits, including improved retrieval quality through interactive reformulation in ambiguity-sensitive domains like older adult mobility.

However, ClariGen has several limitations. The pipeline is heavily dependent on prompt engineering and zero-shot or few-shot techniques, which may reduce robustness across diverse query types. Evaluations were conducted on established datasets like ClariQ and QULAC, but real-world deployment could face challenges from multi-turn interactions, user non-cooperation, or domain-specific ambiguities not fully captured in benchmarks. Moreover, computational costs remain high for the larger model during clarification generation, and the system has not been tested extensively in live user settings, potentially overlooking issues like over-clarification or cultural variations in query expression.

Several promising extensions could further strengthen ClariGen. Adding multi-turn clarification would allow the system to resolve complex ambiguities through iterative follow-up questions, potentially trained with user feedback loops or simulated dialogues. Fine-tuning on domain-specific datasets (e.g., healthcare queries) could improve accuracy in high-stakes settings while reducing dependence on large models for better efficiency. Finally, tighter integration with benchmarks like CLAMBER would enable more granular ambiguity-type analysis and smarter clarification strategies. These improvements would make ClariGen more robust and practical, especially in domains where precise user intent is critical.

Overall, this project highlights the value of modular, prompt-based clarification modules in making QA systems more reliable and user-centred, paving the way for safer and accurate AI assistants in the field of older adults care.

Appendix A

Ambiguity Detection Evaluation Prompts

A.1 System Prompt for ambiguity detection evaluation, zero-shot

```
You are an expert query ambiguity classifier.


Your task is to read a user query and output ONLY:
- "yes" if the query is ambiguous.
- "no" if the query is unambiguous.
Do not output anything else.
```

A.2 System Prompt for ambiguity detection evaluation, few-shot

```
You are an expert query ambiguity classifier.


Your task is to analyze a user query and determine whether it is ambiguous.
You must respond with ONLY:
- "yes",  the query is ambiguous
- "no", the query is unambiguous
No explanations or extra text.


A query is ambiguous ("yes") if it matches ANY of the following types:


1. UNFAMILIAR - References an unknown, mixed, or unlikely entity.
   Example: "Find the price of Samsung Chromecast."


2. CONTRADICTION - Contains logically inconsistent information.
   Example: "The light bulb is on and off at the same time."


3. LEXICAL - A word has multiple meanings that change the interpretation.
   Example: "Can you book the book?"


4. SEMANTIC - Implausible, unclear, or contextually odd meaning.
   Example: "The boar is in the theatre."


5. WHO / WHEN / WHERE / WHAT - Missing referent, time, place, or context.
   Example: "When did he land on the moon?"
```

```
A query is unambiguous ("no") when it is clear, specific, and complete.

Unambiguous examples:

1. "What is the population of Brazil?"

2. "Convert 10 kilometers to miles."

3. "Define the term photosynthesis."

4. "Who invented the telephone?"

5. "Show me the weather forecast for Tokyo tomorrow."
```

A.3 User Prompt for ambiguity detection evaluation

```
"Is the following user query ambiguous?\n\n"

"Query: {query}\n\n"

"Answer with ONLY 'yes' or 'no':"
```

Appendix B

ClariGen Prompts

B.1 Binary Detection

B.1.1 System Prompt

```
    You are an expert at detecting ambiguity in user queries for an
    information-seeking system.

Determine whether the user's query is ambiguous.

Definition:
- A query is clear if a single, well-defined interpretation is the most
    reasonable, and an answer can be provided without making important
    assumptions.
```

```
- A query is ambiguous if there are two or more distinct, reasonable
    interpretations that would lead to materially different answers or
    retrieval results, and the query does not provide enough constraints to
     choose among them.
- Treat missing critical constraints (e.g., which entity, timeframe,
    location, version/edition, metric/units, or sense of a polysemous term)
     as ambiguity when the omission could change the answer.

Decision rule:
- If multiple interpretations are plausible, set "is_ambiguous" to true.
- If one interpretation is clearly dominant and alternatives are unlikely
    or would not change the answer, set it to false.

Output format:
Return ONLY a valid JSON object exactly in this format:
{
  "is_ambiguous": true or false
}
Do not include any text outside the JSON.
```

B.1.2 User Prompt, zero shot

```
Analyze the following query and determine if it is ambiguous or clear.
Query: "{query}"
Output:"""
```

B.1.3 User Prompt, few shot

```
Analyze the following query and determine if it is ambiguous or clear.


Examples:


Example 1:
Query: "What is the capital of France?"
Output: {{"is_ambiguous": false}}


Example 2:
Query: "Tell me about the source of Nile."
Output: {{"is_ambiguous": true}}
(Reason: "source" could mean geographical origin or informational source)


Example 3:
Query: "When did he land on the moon?"
Output: {{"is_ambiguous": true}}
(Reason: "he" is an ambiguous reference - which person?)


Example 4:
```

```
Query: "Find the price of Samsung Chromecast."
Output: {{"is_ambiguous": true}}
(Reason: Samsung doesn't make Chromecast - unfamiliar/incorrect entity)

Example 5:
Query: "What is the population of Tokyo in 2023?"
Output: {{"is_ambiguous": false}}

Example 6:
Query: "John told Mark he won the race."
Output: {{"is_ambiguous": true}}
(Reason: "he" could refer to John or Mark)

Now analyze this query:
Query: "{query}"
Output:"""
```

B.2 Clarification Generation Prompts

B.2.1 vanilla, system prompt

```
You are an expert at analyzing ambiguous user queries and generating
    clarifying questions for an information-seeking system.

Output ONLY a valid JSON object with the following structure:
```

```
{{
    "original_query": "the original query text",
    "clarifying_question": "your generated question"
}}

Do not include any text outside the JSON object.
```

B.2.2 AT-standard, system prompt

```
You are an expert at analyzing ambiguous user queries and generating
    clarifying questions for an information-seeking system.

Here are the possible ambiguity types:

{ambiguity_definitions}

Your task:
Generate ONE clear, simple clarifying question that you think is most
    appropriate to gain a better understanding of the user's intent.
    Consider the above ambiguity types when generating.

Important:
- Avoid compound questions (don't use "or", "and" to ask multiple things)
- Make it natural and conversational
- Focus on the most critical ambiguity if multiple types are present
- If you cannot identify any ambiguity, use "NONE" as the ambiguity type
```

```
Output ONLY a valid JSON object with the following structure:
{{
    "original_query": "the original query text",
    "ambiguity_types": ["LEXICAL", "SEMANTIC"],
    "reasoning": "brief explanation of your clarification approach",
    "clarifying_question": "your generated question"
}}

Do not include any text outside the JSON object."""
```

B.2.3 AT-CoT, system prompt

```
You are an expert at analyzing ambiguous user queries and generating
    clarifying questions for an information-seeking system.

Here are the possible ambiguity types:

{ambiguity_definitions}

Your task (think step by step):
1. Analyze the given query and identify which ambiguity type(s) apply
2. Explain your reasoning: describe which types of ambiguity apply to the
    given query
3. Based on these ambiguity types, describe how you plan to clarify the
    original query
```

```
4. Generate ONE clear, simple clarifying question that addresses the MOST
    IMPORTANT missing information

Important:
- Avoid compound questions (don't use "or", "and" to ask multiple things)
- Make it natural and conversational
- Focus on the most critical ambiguity if multiple types are present
- If you cannot identify any ambiguity, use "NONE" as the ambiguity type

Output ONLY a valid JSON object with the following structure:
{{
    "original_query": "the original query text",
    "ambiguity_types": ["LEXICAL", "SEMANTIC"],
    "reasoning": "your explanation of which types of ambiguity apply and
    how you plan to clarify the query",
    "clarifying_question": "your generated question"
}}

Do not include any text outside the JSON object.
```

B.2.4 Shared user prompt

```
Given a query in an information-seeking system, generate a clarifying
    question that you think is most appropriate to gain a better
    understanding of the user's intent. The ambiguity of a query can be
    multifaceted, and there are multiple possible ambiguity types.

Before generating the clarifying question, provide a textual explanation of
     your reasoning about which types of ambiguity apply to the given query
    . Based on these ambiguity types, describe how you plan to clarify the
    original query.

Query: "{query}"
Output:"""
```

B.3 Query Reformulation

B.3.1 System prompt

```
    You are an expert at reformulating ambiguous queries into clear,
    unambiguous versions.

Your task and think step by step:
1. Review the original ambiguous query
2. Review the ambiguity type and clarifying question
3. Review the user's clarification
```

```
4. Reformulate the query to be clear and unambiguous by incorporating the
    clarification

The reformulated query should:
- Incorporate the information from the user's clarification
- Be clear, specific, and unambiguous
- Maintain the original intent of the query
- Be a complete, standalone query (not requiring additional context)
- Be natural and well-formed

Respond with ONLY the reformulated query - no extra text, explanations, or
    formatting."""
```

B.3.2 User Prompt

```
Original Query: "{original_query}"

Ambiguity Type(s): {types_str}

Clarifying Question: "{clarifying_question}"

User's Clarification: "{user_clarification}"

Reformulate the original query to be clear and unambiguous by incorporating
    the user's clarification. Output ONLY the reformulated query."""
```

Appendix C

Data Schemas for ClariGen

C.1 Pydantic Schemas for Structured Model Outputs

```
"""Pydantic schemas for structured outputs from vLLM models."""


from pydantic import BaseModel, Field
from typing import List



class BinaryDetectionResponse(BaseModel):
    """Schema for binary ambiguity detection response."""

    is_ambiguous: bool = Field(
        description="Whether the query is ambiguous (True) or
          clear (False)"
```

```
    )



class ClarificationResponse(BaseModel):
    """Schema for clarification generation response."""

    original_query: str = Field(description="The original query
       that was ambiguous")
    ambiguity_types: List[str] = Field(
        description="List of identified ambiguity types (e.g., ['
           LEXICAL', 'SEMANTIC'])"
    )
    reasoning: str = Field(
        description="Explanation of how the clarifying question
           will resolve the ambiguity"
    )
    clarifying_question: str = Field(
        description="The generated clarifying question to resolve
            the ambiguity"
    )



class VanillaClarificationResponse(BaseModel):
    """Schema for vanilla clarification generation response."""
```

```
    original_query: str = Field(description="The original query
        that was ambiguous")
    clarifying_question: str = Field(
        description="The generated clarifying question to resolve
            the ambiguity"
    )


class AmbiguityClassificationResponse(BaseModel):
    """Schema for ambiguity classification response."""

    ambiguity_types: List[str] = Field(
        description="List of identified ambiguity types (e.g., ['
            LEXICAL', 'SEMANTIC'])"
    )
    reasoning: str = Field(
        description="Explanation of why these ambiguity types
            were identified"
    )
```

Listing C.1: Pydantic schemas used for structured outputs from vLLM models

Appendix D

Use of Generative Artificial Intelligence

This appendix discloses the use of generative artificial intelligence tools during the preparation of this report, in accordance with McMaster University's Provisional Guidelines on the Use of Generative AI in Research (January 2025). These guidelines stress transparency, responsible application, and human oversight in all research stages, including dissemination.

D.1 Overall Statement

Generative AI tools were used in supportive, non-substantive roles to assist with drafting, refinement, and visualization tasks. Specific uses included:

- Structuring report outlines and chapter suggestions.
- Grammar checking, style improvements, and rephrasing for clarity.

- Code generation, refactoring, and bug fixing in implementation scripts.
- Suggestions for diagram generation (e.g., system architecture and state machine illustrations).
- LaTeX syntax assistance and formatting code for sections, tables, and appendices.

No AI-generated text, code, or figures were incorporated verbatim without substantial review, editing, verification, and modification by the author. All core intellectual contributions including the design of the ClariGen pipeline, ambiguity taxonomy modifications, prompting strategies, experimental methodology, data analysis, evaluation conducted and case studies are the author's original work.

D.2 Tools Employed

The following tools were used in late 2025:

Tool	**Provider**	**Primary Uses**
Claude (Sonnet 4.5 series)	Anthropic	Code-related assistance (evaluation script generation, code refactoring, bug fixing)
Gemini (Pro 3 series and Flash variants)	Google	Diagram generation suggestions and LaTeX syntax assistance
ChatGPT (GPT series, accessed as equivalent to GPT-4 in 2025)	OpenAI	Report outline drafting and grammar checking
Grok (various versions, including Grok-4)	xAI	Structural suggestions (e.g., appendix design), prompt engineering brainstorming, and general refinement

Table D.1: Generative AI tools employed in report preparation.

D.3 Extent and Boundaries

- AI assistance was limited to auxiliary tasks that enhanced efficiency without contributing original project content.
- AI did not generate experimental results, analyses, benchmark evaluations, ClariGen prompts (Appendices A–C), or primary technical descriptions.
- All AI outputs underwent critical evaluation, significant editing, and validation by the author to ensure factual accuracy and alignment with project objectives.
- Diagrams and figures in the report (e.g., system overview in Chapter 3) were conceptualized and drawn by the author and AI provided iterative suggestions where applicable.

Bibliography

[1] Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. 2020. ConvAI3: Generating clarifying questions for open-domain dialogue systems (ClariQ). *arXiv preprint arXiv:2009.11352* (2020).

[2] Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. 2021. Building and evaluating open-domain dialogue corpora with clarifying questions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 4473–4484.

[3] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W Bruce Croft. 2019. Asking clarifying questions in open-domain information retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 475–484.

[4] Hiteshwar Kumar Azad and Akshay Deepak. 2019. Query expansion techniques for information retrieval: a survey. *Information Processing & Management* 56, 5 (2019), 1698–1735.

[5] Keping Bi, Qingyao Ai, and W Bruce Croft. 2021. Asking clarifying questions based on negative feedback in conversational search. In *Proceedings of the 2021*

ACM SIGIR International Conference on Theory of Information Retrieval. 157–166.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[7] Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. Rq-rag: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610* (2024).

[8] Maximillian Chen, Ruoxi Sun, Tomas Pfister, and Sercan Ö Arık. 2024. Learning to clarify: Multi-turn conversations with action-based contrastive self-training. *arXiv preprint arXiv:2406.00222* (2024).

[9] Yizhou Chi, Jessy Lin, Kevin Lin, and Dan Klein. 2024. Clarinet: Augmenting language models to ask clarification questions for retrieval. *arXiv preprint arXiv:2405.15784* (2024).

[10] Charles L Clarke, Nick Craswell, and Ian Soboroff. 2009. Overview of the trec 2009 web track. (2009).

[11] Steve Cronen-Townsend, Yun Zhou, and W Bruce Croft. 2002. Predicting query performance. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. 299–306.

[12] Lenart Dolinar, Erik Calcina, and Erik Novak. 2025. Evaluating Open-Source

Large Language Models for Synthetic Non-English Medical Data Generation Using Prompt-Based Techniques. *Informatica* 49, 27 (2025).

[13] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv e-prints* (2024), arXiv–2407.

[14] Kimiya Keyvan and Jimmy Xiangji Huang. 2022. How to approach ambiguous queries in conversational search: A survey of techniques, approaches, tools, and challenges. *Comput. Surveys* 55, 6 (2022), 1–40.

[15] Johannes Kiesel, Arefeh Bahrami, Benno Stein, Avishek Anand, and Matthias Hagen. 2018. Toward voice query clarification. In *The 41st international acm sigir conference on research & development in information retrieval*. 1257–1260.

[16] Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, and Jaewoo Kang. 2023. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 996–1009.

[17] Joo-Kyung Kim, Guoyin Wang, Sungjin Lee, and Young-Bum Kim. 2021. Deciding whether to ask clarifying questions in large-scale spoken language understanding. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 869–876.

[18] Antonios Minas Krasakis, Mohammad Aliannejadi, Nikos Voskarides, and Evangelos Kanoulas. 2020. Analysing the effect of clarifying questions on document

ranking in conversational search. In *Proceedings of the 2020 acm sigir on international conference on theory of information retrieval*. 129–132.

[19] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2022. Clam: Selective clarification for ambiguous questions with generative language models. *arXiv preprint arXiv:2212.07769* (2022).

[20] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. arXiv:2309.06180 [cs.LG] `https://arxiv.org/abs/2309.06180`

[21] Dongryeol Lee, Segwang Kim, Minwoo Lee, Hwanhee Lee, Joonsuk Park, Sang-Woo Lee, and Kyomin Jung. 2023. Asking clarification questions to handle ambiguity in open-domain QA. *arXiv preprint arXiv:2305.13808* (2023).

[22] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[23] Lili Lu, Chuan Meng, Federico Ravenda, Mohammad Aliannejadi, and Fabio Crestani. 2025. Zero-Shot and Efficient Clarification Need Prediction in Conversational Search. arXiv:2503.00179 [cs.IR] `https://arxiv.org/abs/2503.00179`

[24] Meta. 2025. meta-llama/Llama-3.1-8B-Instruct. `https://huggingface.co/`

meta-llama/Llama-3.1-8B-Instruct. Hugging Face model card. Accessed: 2025-12-15.

[25] Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart Oberman, Mohammad Shoeybi, Michael Siu, and Hao Wu. 2022. FP8 Formats for Deep Learning. arXiv:2209.05433 [cs.LG] https://arxiv.org/abs/2209.05433

[26] Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering Ambiguous Open-domain Questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 5783–5797.

[27] NVIDIA. 2025. Llama 3.3 70B Instruct FP8 (Hugging Face Model Card). https://huggingface.co/nvidia/Llama-3.3-70B-Instruct-FP8. Accessed: 2025-12-16.

[28] OpenAI. 2025. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/. Accessed: 2025-12-01.

[29] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[30] Bhawna Piryani, Abdelrahman Abdallah, Jamshid Mozafari, and Adam Jatowt. 2024. Detecting temporal ambiguity in questions. *arXiv preprint arXiv:2409.17046* (2024).

[31] David Schlangen. 2004. Causes and strategies for requesting clarification in dialogue. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue at HLT-NAACL 2004*. 136–143.

[32] Zhengyan Shi, Giuseppe Castellucci, Simone Filice, Saar Kuzi, Elad Kravi, Eugene Agichtein, Oleg Rokhlenko, and Shervin Malmasi. 2025. Ambiguity detection and uncertainty calibration for question answering with large language models. In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*. 41–55.

[33] Ruihua Song, Zhenxiao Luo, Ji-Rong Wen, Yong Yu, and Hsiao-Wuen Hon. 2007. Identifying ambiguous queries in web search. In *Proceedings of the 16th international conference on World Wide Web 2007*. 1169–1170. `https://www.microsoft.com/en-us/research/publication/identifying-ambiguous-queries-in-web-search/`

[34] Anfu Tang, Laure Soulier, and Vincent Guigue. 2025. Clarifying ambiguities: on the role of ambiguity types in prompting methods for clarification generation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 20–30.

[35] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane

Rivière, et al. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786* (2025).

[36] Antonia Wiebeler and Nikita Kozodoi. 2023. Improve LLM responses in RAG use cases by interacting with the user. https://aws.amazon.com/blogs/machine-learning/improve-llm-responses-in-rag-use-cases-by-interacting-with-the-user/. AWS Machine Learning Blog.

[37] Hamed Zamani, Susan Dumais, Nick Craswell, Paul Bennett, and Gord Lueck. 2020. Generating clarifying questions for information retrieval. In *Proceedings of the web conference 2020*. 418–428.

[38] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. arXiv:1904.09675 [cs.CL] https://arxiv.org/abs/1904.09675

[39] Tong Zhang, Peixin Qin, Yang Deng, Chen Huang, Wenqiang Lei, Junhong Liu, Dingnan Jin, Hongru Liang, and Tat-Seng Chua. 2024. Clamber: A benchmark of identifying and clarifying ambiguous information needs in large language models. *arXiv preprint arXiv:2405.12063* (2024).

[40] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-Following Evaluation for Large Language Models. arXiv:2311.07911 [cs.CL] https://arxiv.org/abs/2311.07911