Diversity is Not Ambiguity: Toward Accurate and Efficient Ambiguity Detection for Open-Domain QA

Anonymous ACL submission

Abstract

How can question answering systems determine whether a query is ambiguous? Ambiguity detection is essential in open-domain question answering, as misclassification leads to either confidently answering the wrong interpretation or triggering unnecessary clarification. However, existing methods conflate answer diversity with ambiguity, leading to inaccurate predictions. They also process all queries uniformly, resulting in wasteful computation. To address these limitations, we propose ARCHIVE (Ambiguity Recognition via Cascaded Hypothesis Inspection and Conflict VErification), an accurate and efficient framework that detects ambiguity via *logical conflict*: a query is ambiguous if and only if its valid answers cannot all be true under a single interpretation. ARCHIVE combines a lightweight early-exit encoder for surface-detectable cases with a conflict reasoning module that models logical relations among answers, reinforced by an invariance objective for robustness to noisy answer sets. We also present QUIREQA, a 4,703-query benchmark spanning factoid, non-factoid, and ill-formed queries. Experiments show ARCHIVE outperforms competitors, improving F1-amb by up to 10.4% and F1-unamb by up to 21.6%, while operating 16× faster than the best competitor.

1 Introduction

Given a user query q in an open-domain question answering (QA) system, how can we accurately and efficiently determine whether q is ambiguous? Ambiguity detection is critical because users often issue underspecified queries when they lack domain knowledge (Min et al., 2020). A QA system must either *answer directly* when the query is unambiguous or *ask a clarifying question* when multiple interpretations exist (Sun et al., 2023). Missed ambiguity leads systems to commit to a single interpretation, producing incorrect or misleading answers (Liao et al., 2023). Conversely, false positives trigger unnecessary clarification, degrading interaction quality (Kuhn et al., 2022).

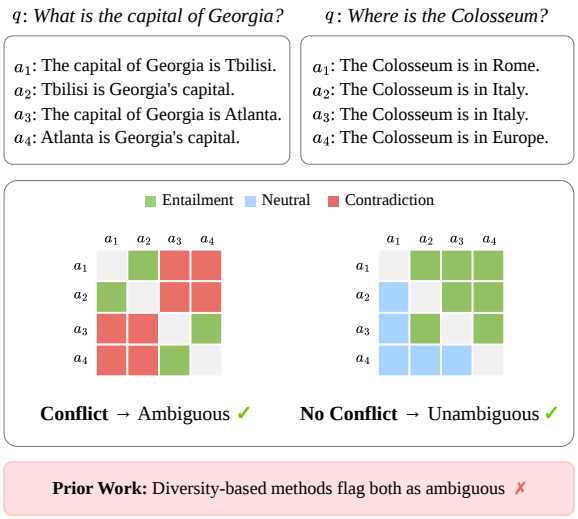


Figure 1: **Diversity is not ambiguity.** Both queries yield sets of diverse answers, but only the first contains genuinely conflicting interpretations. ARCHIVE distinguishes them by testing for logical conflict rather than mere answer diversity.

Although modern LLMs are capable of generating clarifying questions, the key challenge is deciding *when* to do so (Braslavski et al., 2017). Existing approaches detect ambiguity through sets of candidate answers, treating answer diversity as a signal for ambiguity (Zhang and Choi, 2025; Shi et al., 2025; Min et al., 2020; Lee et al., 2023; Sun et al., 2023). However, this view fundamentally cannot distinguish *incompatible answers that arise from different interpretations* from *compatible answers that describe the same fact*, as both manifest as answer diversity. For example, "What is the capital of Georgia?" yields incompatible answers (Tbilisi vs. Atlanta), whereas "Where is the Colosseum?" yields diverse but compatible responses (Rome, Italy, Europe). Prior methods label both as ambiguous due to answer diversity, even though only the first exhibits genuine conflict (Figure 1). As a result, they generate unnecessary clarifying questions for unambiguous queries (e.g., "Do you mean the city or the country?") and incur unnecessary computation by routing every query through costly LLM pipelines.

Table 1: **Coverage gaps and non-exclusive labels in prior taxonomies.** Each row shows how a taxonomy labels four canonical query types; only ARCHIVE provides a mutually exclusive assignment for every query.

Taxonomy \ Query	*"what is the"*	*"What is the capital of Georgia?"*	*"Where is the Colosseum?"*	*"How should I organize my desk?"*
Zhang et al. (2024)	*N/A*	Lexical	*N/A*	What, Whom
Tang et al. (2025)	*N/A*	Semantic, Specify	*N/A*	Specify
Tanjim et al. (2025)	Syntactic	*N/A*	*N/A*	*N/A*
ARCHIVE (proposed)	Surface-detectable ambiguous	Knowledge-dependent ambiguous	Closed unambiguous	Open unambiguous

No applicable class | Overlapping labels | Mutually exclusive classes in ARCHIVE

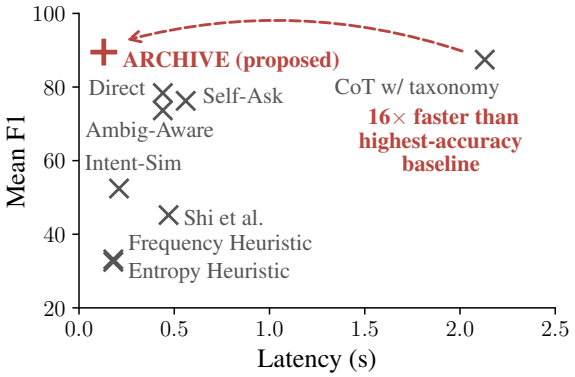


Figure 2: **ARCHIVE achieves the highest mean F1 at the lowest latency on Qwen2.5-14B.** Mean F1 averages F1-amb and F1-unamb.

To address these limitations, we propose ARCHIVE (Ambiguity Recognition via Cascaded Hypothesis Inspection and Conflict VErification), an accurate and efficient framework that redefines ambiguity detection through *logical conflict*: a query is ambiguous if and only if its valid answers cannot all be true under a single interpretation. Unlike diversity-based heuristics, this formulation directly captures the underlying cause of ambiguity and avoids conflating benign variation with genuine uncertainty. Based on this formulation, ARCHIVE adopts a cascaded architecture. A lightweight encoder first resolves surface-detectable cases via early exit, reducing unnecessary computation. For remaining queries, ARCHIVE generates candidate answers and models logical relations among them to detect conflict. To handle noisy or incomplete generations, we introduce an invariance objective that stabilizes predictions under perturbations of the candidate set. Our contributions are as follows:

- **Ambiguity via logical conflict.** We redefine ambiguity as logical conflict among candidate answers and present a mutually exclusive four-class taxonomy based on this formulation.
- **Accurate and efficient detection.** ARCHIVE combines a lightweight early-exit encoder with a conflict reasoning module, reinforced by an invariance objective for robustness to noisy LLM generations. As shown in Figure 2, it operates 16× faster than the highest-accuracy baseline.
- **Benchmark and experiments.** We construct QUIREQA, a 4,703-query benchmark with annotated context for factoid queries. ARCHIVE outperforms nine baselines, improving F1-amb by up to 10.4% and F1-unamb by up to 21.6%.

Code and data are available at: https://anonymous.4open.science/r/archive_.

2 Related Work

2.1 Prior Taxonomies of Ambiguity

Prior taxonomies enumerate categories of ambiguity without providing an operational criterion for distinguishing ambiguous from unambiguous queries. Earlier works categorized ambiguity types within specific QA datasets without targeting detection (Min et al., 2020; Guo et al., 2021; Amplayo et al., 2022). Three recent taxonomies address ambiguity detection: Zhang et al. (2024) define eight categories across linguistic, epistemic, and aleatoric dimensions; Tang et al. (2025) propose three action-oriented classes to guide LLM response strategies, explicitly aiming for a mutually exclusive partition; Tanjim et al. (2025) categorize linguistic defects in conversational QA logs. However, without a unifying criterion, these frameworks admit overlap: a single query may satisfy multiple categories (Table 1), leaving annotators with no principled way to choose among labels.

A second limitation is coverage. Zhang et al. (2024) and Tang et al. (2025) assume well-formed input and do not account for surface defects, which are common in real query logs (Craswell et al., 2020). Tanjim et al. (2025) address this gap but are scoped to enterprise data and do not cover open-domain knowledge ambiguity. These limitations motivate a different organizing principle: rather

than enumerating ambiguity categories, we define ambiguity via *logical conflict*—whether candidate answers can be jointly true under a single interpretation of the query—and separate surface-detectable cases as a distinct class, forming a mutually exclusive partition (Figure 3, Section 3.1).

2.2 Ambiguity Detection

Directly prompting an LLM to detect ambiguity is unreliable even for strong models (Zhang et al., 2024). Existing methods address this through indirect signals, but exhibit three main limitations. First, these methods route every query through the same generation pipeline, even when ambiguity is detectable from surface form alone. This incurs unnecessary cost on trivially resolvable cases.

Second, diversity is an unreliable proxy. A common approach estimates ambiguity from the diversity of generated outputs (Cole et al., 2023), for example via entropy over sampled answers (Shi et al., 2025; Kuhn et al., 2023) or LLM-simulated user responses (Zhang and Choi, 2025). However, this conflates *aleatoric uncertainty* (genuine input ambiguity) with *epistemic uncertainty* (gaps in model knowledge) (Hou et al., 2024). Hallucinations can inflate diversity and produce false positives, while insufficient knowledge can suppress it and cause false negatives. Diversity does not distinguish incompatible answers from benign variation. For instance, "What is the capital of Georgia?" yields incompatible answers (Tbilisi vs. Atlanta), whereas "How should I organize my desk?" yields diverse yet compatible suggestions. Yet both produce high-entropy answer sets. Shi et al. (2025) reduce the epistemic component using oracle context, but still rely on output diversity.

Third, some methods define ambiguity relative to a particular model. Kim et al. (2024) measure ambiguity through information gain between a query and its rewrite under a specific LLM, while Zhang et al. (2024) formalize epistemic misalignment with respect to the LLM's knowledge state. Although framed as a feature of model-specific ambiguity, such formulations couple detection to a particular model's capabilities, making predictions sensitive to model choice and future updates.

In contrast, our proposed ARCHIVE defines ambiguity via *logical conflict* among valid answers, rather than output diversity. It filters surface-detectable cases before generation, derives candidate answers from evidence context, and trains for robustness against fabricated distractors.

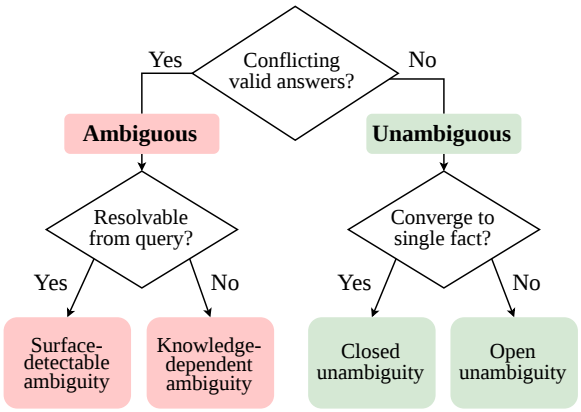


Figure 3: **Proposed taxonomy of query ambiguity.**

3 Problem Formulation

We formalize ambiguity detection in two parts: a taxonomy (Section 3.1) and a benchmark that applies this criterion to four query sources, including queries that prior datasets label inconsistently under logical conflict (Section 3.2).

3.1 Proposed Taxonomy

Building on the limitations in Section 2, we define ambiguity by *logical conflict* among valid answers.

Definition. *A query q is **ambiguous** if its valid answers include a_i, a_j that cannot be jointly satisfied under a single interpretation of q; it is **unambiguous** if all valid answers are compatible.*

Taxonomy. As shown in Figure 3, queries are first classified as *ambiguous* or *unambiguous* based on logical conflict, then divided into four classes:

1. **Surface-detectable ambiguity.** Queries whose plausible repairs or completions yield interpretations with conflicting answers (e.g., malformed syntax or unanchored references).
2. **Knowledge-dependent ambiguity.** Well-formed queries admitting multiple interpretations with conflicting answers (e.g., "capital of Georgia": Tbilisi vs. Atlanta).
3. **Closed unambiguity.** Answers converge to a single fact or bounded set (e.g., Rome, Italy).
4. **Open unambiguity.** A single interpretation admits multiple compatible answers (e.g., diverse advice for organizing a desk).

Surface-detectable ambiguity is prevalent in practice: we apply heuristics (Appendix B.3) to the 10.4M-query ORCAS corpus (Craswell et al., 2020) and identify 124K candidate queries, of which 1,511/6,336 reviewed cases are confirmed as surface-detectable ambiguous.

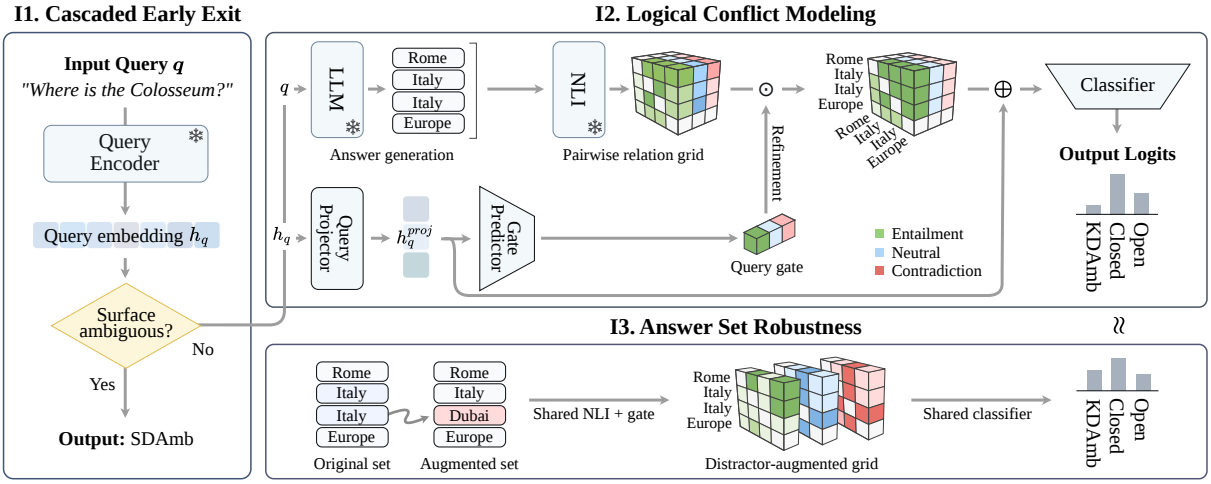


Figure 4: **Overview of ARCHIVE.** ARCHIVE early-exits on surface-detectable ambiguous (SDAmb) queries, reducing unnecessary LLM calls (I1). Remaining queries are classified by modeling pairwise logical relations among candidate answers (I2), and robustness is enforced through prediction invariance after distractor augmentation (I3).

3.2 Dataset

Existing benchmarks focus on factoid queries with diversity-based ambiguity labels that do not align with logical conflict. We therefore construct QUIREQA (Queries with Unstructured and Incomplete Real-world Expressions), a benchmark of factoid and non-factoid queries from four sources, re-annotated under our four-class taxonomy (Table 2). For example, AmbigNQ (Min et al., 2020) labels "Where is the oldest bristlecone pine tree located?" as ambiguous because its answers span multiple geographic levels (peak, state, country), yet these describe the same location at different granularities and are closed unambiguous under our criterion. Overall, 206 AmbigNQ queries flagged as ambiguous by previous methods are unambiguous under our criterion. To support future work on context-aware detection, we annotate and provide 3,299 oracle context passages across 680 queries. Details are in Appendix B.

Table 2: **QUIREQA statistics.** SDAmb/KDAmb: surface-detectable/knowledge-dependent ambiguous; Closed/Open: closed/open unambiguous.

	Ambiguous		Unambiguous		
Source	SDAmb	KDAmb	Closed	Open	Total
ORCAS (2020)	1,512	2	60	37	1,611
AmbigNQ (2020)	0	414	1,254	11	1,679
AmbER (2021)	0	317	83	0	400
Dolly-15K (2023)	0	4	61	948	1,013
Total	1,512	737	1,458	996	4,703

4 Proposed Method

We now present ARCHIVE, a framework for accurate and efficient ambiguity detection in open-domain QA. It addresses three key challenges:

C1. Computational cost. Existing methods apply generation and reasoning to every query, even when ambiguity is detectable from surface form alone. How can we avoid unnecessary LLM inference while preserving accuracy?

C2. Ambiguity vs. answer diversity. Prior work treats answer variability as a proxy for ambiguity. However, diverse answers may be mutually compatible (e.g., open-ended queries), while truly ambiguous queries exhibit conflicting interpretations. How can we distinguish true ambiguity from benign diversity?

C3. Unreliable LLM generations. LLMs may introduce hallucinated contradictions or omit valid alternative answers, causing detectors to misestimate ambiguity. How can predictions remain stable under such noise?

As illustrated in Figure 4, ARCHIVE tackles these challenges with the following main ideas:

I1. Cascaded early exit (Section 4.1). A lightweight encoder-based module detects surface-detectable ambiguity from the query, bypassing LLM inference when possible.

I2. Logical conflict modeling (Section 4.2). For remaining queries, we generate candidate answers and model pairwise relations to detect logical conflict among them.

I3. Answer set robustness (Section 4.3). Distractor augmentation with an invariance loss reduces sensitivity to hallucinated contradictions, while context-grounded generation helps recover omitted valid answers.

The early-exit and conflict modeling modules are trained independently. The former minimizes $\mathcal{L}_{\text{exit}}$, whereas the latter optimizes

$$\mathcal{L}_{\text{conflict}} = \mathcal{L}_{\text{CE}} + \lambda_{\text{inv}}\mathcal{L}_{\text{inv}},$$

where $\mathcal{L}_{\text{CE}}$ supervises classification, $\mathcal{L}_{\text{inv}}$ enforces robustness to distractor perturbations, and λ_{inv} controls the invariance strength.

4.1 Cascaded Early Exit

The first stage of ARCHIVE detects *surface-detectable ambiguity* (SDAmb) with a lightweight classifier on a frozen text encoder. This reduces cost and latency by reserving LLM calls for queries that require deeper reasoning. Given a query q, the encoder produces a mean-pooled query embedding $\mathbf{h}_q \in \mathbb{R}^d$. A trainable linear classifier head with sigmoid activation estimates $p = P(\text{SDAmb} \mid \mathbf{h}_q)$. It is trained with binary cross-entropy:

$$\mathcal{L}_{\text{exit}} = -\ell \log p - (1-\ell)\log(1-p),$$

where $\ell \in \{0, 1\}$ denotes the SDAmb label. At inference, queries with $p > \tau$ exit early and return the SDAmb label, where τ is a validation-tuned routing threshold. Otherwise, ARCHIVE forwards the query and its embedding to the conflict modeling module for deeper analysis.

4.2 Logical Conflict Modeling

For queries not handled by early exit, the conflict modeling module predicts one of three labels: knowledge-dependent ambiguity (KDAmb), closed unambiguity (Closed), or open unambiguity (Open). The key insight is that ambiguity manifests as *logical conflict* among valid answers, not answer diversity. The module thus generates candidate answers for a given query, employs natural language inference (NLI) to organize their pairwise relations into a grid that separates contradictions from compatible variations, and classifies the query from a query-conditioned refinement of that grid.

Answer generation. For each query q, we first generate a set $\{a_1, \ldots, a_k\}$ of k=10 candidate answers for conflict analysis. We prompt an LLM five times to produce multiple answers covering different interpretations of q, formatted as short declarative statements for reliable downstream NLI. The top-k most frequent unique answers constitute the final set. If fewer than k unique answers are produced, we pad the set by duplicating existing entries in proportion to their frequency, ensuring a fixed-size grid. Generation is conditioned on context when available, as described below. See Appendices D and F for generation details and prompts.

Pairwise relation grid. Rather than relying on indirect proxies, we model the logical structure among candidate answers as pairwise NLI relations. Given the answer set $\{a_1, \ldots, a_k\}$, we construct a relation grid $S \in \mathbb{R}^{k\times k\times 3}$. For each ordered pair (a_i, a_j) with $i \neq j$, a frozen NLI model (Laurer et al., 2022) predicts a distribution over entailment (E), neutral (N), and contradiction (C), indicating whether a_i entails, is neutral to, or contradicts a_j:

$$S_{i,j} = [p_E, p_N, p_C] \in [0,1]^3, \quad \sum_{r\in\{E,N,C\}} p_r = 1.$$

Thus, each channel of S represents one relation type over all k^2 pairs. Diagonal entries are zeroed because self-relations are uninformative.

Query-conditioned relation refinement. The grid S isolates answer-level conflict, the core signal of ambiguity under our definition. However, its interpretation depends on the query. Open unambiguous queries naturally produce neutral-dominated grids due to diverse compatible answers, while closed unambiguous queries may occasionally appear similar when NLI fails to recognize equivalent facts. To mitigate this noise, we apply a lightweight query gate that recalibrates the entailment, neutral, and contradiction channels of S. The query embedding $\mathbf{h}_q$ is first projected as $\mathbf{h}_q^{\text{proj}} = f_{\text{proj}}(\mathbf{h}_q)$ and mapped to per-channel scaling factors:

$$\mathbf{g} = 2\sigma(f_{\text{gate}}(\mathbf{h}_q^{\text{proj}})) \in \mathbb{R}^3,$$

where f_{proj} is a linear layer with ReLU activation, f_{gate} is a 2-layer MLP with ReLU activation, and σ denotes a sigmoid applied externally to the output of f_{gate}. The factor of 2 centers $\mathbf{g}$ at the identity ($\mathbf{g} = \mathbf{1}$ leaves S unchanged). The query gate $\mathbf{g} = [g_E, g_N, g_C]$ acts as a channel-wise temperature on the relation logits:

$$\tilde{S}_{i,j} = \text{softmax}_{r\in\{E,N,C\}}(\log(S_{i,j} + \epsilon) \odot \mathbf{g}),$$

where the softmax is applied for each pair (i, j) over the three NLI channels, $\odot$ is elementwise multiplication, and ϵ ensures numerical stability. $g_r > 1$ sharpens relation type r, whereas $g_r < 1$ suppresses it. A single shared $\mathbf{g}$ is applied to all k^2 pairs for efficient query-dependent refinement; pair-specific conflict remains encoded in the scores $S_{i,j}$. Figure 8 in Appendix J shows examples of $\tilde{S}$.

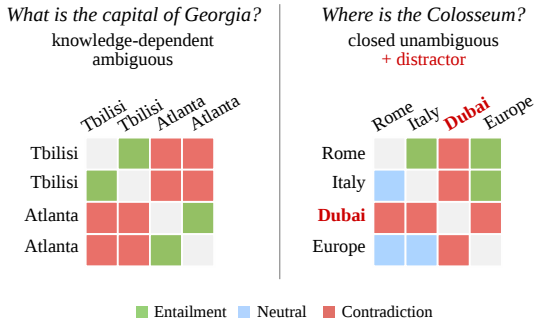


Figure 5: **Contradiction patterns distinguish genuine ambiguity from a distractor.** Genuine ambiguity yields distributed contradictions (Left), whereas a distractor yields isolated row/column contradictions (Right).

Classification. Given the flattened grid $\mathrm{vec}(\tilde{S}) \in \mathbb{R}^{3k^2}$ and projected query embedding $\mathbf{h}_q^{\text{proj}}$, we generate class logits

$$\hat{y} = f_{\text{cls}}\Big([\mathrm{vec}(\tilde{S});\, \mathbf{h}_q^{\text{proj}}]\Big) \in \mathbb{R}^3,$$

where f_{cls} is a 2-layer MLP with ReLU activation, and $\mathrm{vec}(\cdot)$ denotes vectorization. Concatenating $\mathbf{h}_q^{\text{proj}}$ grounds grid patterns in query intent, enabling finer distinctions (e.g., open vs. closed unambiguous) that channel-wise modulation alone cannot make. The classifier is trained with cross-entropy:

$$\mathcal{L}_{\text{CE}} = -\log \mathrm{softmax}(\hat{y})_{c^*},$$

where $c^* \in \{\text{KDAmb}, \text{Closed}, \text{Open}\}$ is the ground-truth class.

4.3 Answer Set Robustness

The conflict modeling module depends on the candidate answer set, but LLM generation may hallucinate unsupported answers or omit valid interpretations. We address hallucination with distractor invariance and omission with context grounding.

Invariance to distractors. Hallucinated answers should not change the predicted ambiguity label. We therefore train the classifier to be invariant to a synthetic distractor inserted into the candidate answer set. In QUIREQA, the LLM always produces fewer than k=10 unique answers. Thus, at least one duplicate answer is always present in $\{a_1, \ldots, a_k\}$, which we replace with an LLM-generated distractor (see Appendix F for the prompt). This distractor matches the form of valid answers but is factually invalid, contradicting the valid answers or available context. As Figure 5 shows, such distractors create *pointwise* rather than distributed contradictions. We enforce invariance by minimizing the KL divergence between classifier logits on clean ($\hat{y}$) and augmented ($\hat{y}^{\text{aug}}$) inputs:

$$\mathcal{L}_{\text{inv}} = \mathrm{KL}(\mathrm{softmax}(\hat{y}^{\text{aug}}) \parallel \mathrm{softmax}(\hat{y}))\,.$$

Algorithm 1 ARCHIVE Inference

Input: Query q, context c, and threshold τ
Output: $y \in \{\text{SDAmb}, \text{KDAmb}, \text{Closed}, \text{Open}\}$

$\mathbf{h}_q, p \leftarrow \mathrm{EarlyExit}(q)$ ▷ surface check
if $p > \tau$ **then**
  return SDAmb ▷ early exit
end if
$\{a_i\}_{i=1}^{k} \leftarrow \mathrm{LLM}(q, c)$
$S \leftarrow \mathrm{NLI}(a_1, \ldots, a_k)$ ▷ relation grid
$\mathbf{g} \leftarrow 2\sigma(f_{\text{gate}}(f_{\text{proj}}(\mathbf{h}_q)))$ ▷ query gate
$\tilde{S} \leftarrow \mathrm{Refine}(S, \mathbf{g})$ ▷ refinement
$\hat{y} \leftarrow f_{\text{cls}}([\mathrm{vec}(\tilde{S}); f_{\text{proj}}(\mathbf{h}_q)])$ ▷ classification
return $\arg\max \hat{y}$

Context grounding. Distractor invariance handles hallucinated answers, but cannot recover valid answers the LLM omits when generating without context. We thus provide an oracle context passage—a relevant document supplied with the query—during answer generation. This anchors candidate answers in external evidence and better covers the query's valid answer space.

4.4 Inference

Algorithm 1 shows the inference pipeline. The early-exit module first estimates the probability of surface-detectable ambiguity; if $p > \tau$, the pipeline exits without LLM calls. Otherwise, the conflict modeling module generates answers, builds and refines an NLI relation grid, and combines the grid with the query embedding to predict $\arg\max \hat{y}$.

5 Experiments

We evaluate ARCHIVE across five key questions:

Q1. Taxonomy utility (Section 5.2). Does the proposed taxonomy outperform existing ones in ambiguity detection and response generation?

Q2. Binary ambiguity detection (Section 5.3). Does ARCHIVE improve binary ambiguity detection accuracy over prior methods?

Q3. Fine-grained detection (Section 5.4). Does ARCHIVE accurately predict the four classes?

Q4. Efficiency (Section 5.5). Does the cascade reduce inference cost while preserving accuracy?

Q5. Ablation (Section 5.6). How much does each module contribute to the overall performance?

See Appendices E and I for additional analyses.

Table 3: **Defining ambiguity by logical conflict improves taxonomy utility.** Rows vary only by taxonomy. CQ/Ans BERTScore are computed on knowledge-dependent ambiguous and closed-unambiguous queries, respectively.

Backbone	Taxonomy	F1-amb	F1-unamb	CQ BERTScore	Ans BERTScore
LLaMA-2-7B	Zhang et al. (2024)	36.20	28.57	82.08	80.81
	Tang et al. (2025)	64.99	0.00	**84.37**	79.42
	Tanjim et al. (2025)	64.92	0.00	82.51	78.68
	ARCHIVE (proposed)	**69.83**	**64.21**	83.69	**83.97**
Qwen2.5-14B	Zhang et al. (2024)	69.58	36.70	81.34	82.04
	Tang et al. (2025)	65.82	17.06	86.18	82.57
	Tanjim et al. (2025)	67.86	28.42	83.22	83.01
	ARCHIVE (proposed)	**75.25**	**64.84**	**86.70**	**83.76**

5.1 Experimental Setup

Dataset. We train and evaluate on QUIREQA, using 8:1:1 train/validation/test split. For out-of-distribution evaluation, we additionally test on AmbigTriviaQA (Kim et al., 2024, Appendix C).

Backbones. The conflict modeling module is model-agnostic; we instantiate it with LLaMA-2-7B (Touvron and 67 others, 2023) and Qwen2.5-14B-Instruct (Qwen Team, 2025). The early-exit module uses DeBERTa-v3-base (He et al., 2023). Pairwise relations are computed with the released DeBERTa-v3 NLI checkpoint of Laurer et al. (2022). All backbone models are kept frozen.

Competitors. We compare with nine methods: (1) *prompting*: Direct, a simple QA prompt, Ambig-Aware (Kim et al., 2024) with ambiguity-specific instructions, Self-Ask (Amayuelas et al., 2024), which generates an answer before judging ambiguity, and CoT w/ taxonomy, adapted from Tang et al. (2025) using our taxonomy; (2) *diversity heuristics*: Frequency, Entropy, Intent-Sim (Zhang and Choi, 2025), which infer ambiguity from answer diversity; and (3) *trained models*: APA (Kim et al., 2024) and Shi et al. (2025). Details are in Appendix H.

Metrics. We report per-class F1 (F1-amb, F1-unamb) for binary detection; 4-way classification uses macro- and per-class F1. Generation quality is measured with BERTScore (Zhang et al., 2020) for answers and clarifying questions. Latency is the mean end-to-end wall-clock time per query. All results are averaged over 3 seeds.

5.2 Taxonomy Utility (Q1)

We fix the prompt and vary only the taxonomy schema, evaluating all queries with LLM generation conditioned on oracle context. CQ BERTScore and Ans BERTScore measure response quality on KDAmb and Closed queries, respectively—the classes with available references. We construct pseudo-reference clarifying questions by adapting the clarification prompt of Kim et al. (2024) to each query and its gold disambiguation (details are in Appendix G). Table 3 shows that our taxonomy yields the strongest detection and competitive response quality relative to symptom-based (Zhang et al., 2024), action-oriented (Tang et al., 2025), and linguistic (Tanjim et al., 2025) alternatives, supporting logical conflict as a useful criterion.

Table 4: **ARCHIVE achieves the highest F1 in binary ambiguity detection.** Factoid queries with no context are used, matching the setup of prior detection work.

Method	QuireQA		AmbigTriviaQA	
	F1-amb	F1-unamb	F1-amb	F1-unamb
Backbone: *LLaMA-2-7B*				
Direct	2.77	59.52	55.99	1.17
Ambig-Aware	49.70	0.00	56.76	0.80
Self-Ask	33.98	64.04	54.61	56.54
CoT w/ taxonomy	49.85	4.01	49.14	57.80
Frequency	49.70	0.00	56.67	0.00
Entropy	48.80	32.14	39.72	30.87
Intent-Sim	41.84	66.57	28.01	48.13
APA	52.24	20.94	25.53	60.06
ARCHIVE	**58.96**	**85.57**	**60.55**	**66.58**
Backbone: *Qwen2.5-14B*				
Direct	3.74	68.76	25.34	59.60
Ambig-Aware	45.32	69.18	65.75	61.98
Self-Ask	34.51	72.43	65.54	68.22
CoT w/ taxonomy	45.27	69.87	69.67	62.39
Frequency	49.70	0.00	66.67	0.00
Entropy	49.67	2.42	66.84	6.80
Intent-Sim	48.68	15.91	71.08	59.67
APA	50.88	64.05	53.95	67.22
ARCHIVE	**61.31**	**85.66**	**79.19**	**69.04**

5.3 Binary Ambiguity Detection (Q2)

Table 4 compares ARCHIVE against competitors on factoid queries in a context-free setting, following prior work. Shi et al. (2025) assume access to

Table 5: **ARCHIVE outperforms baselines in 4-way classification.** We evaluate on all queries with the LLM conditioned on oracle context when available.

Backbone	Method	Macro-F1	Per-Class F1			
			SDAmb	KDAmb	Closed	Open
LLaMA-2-7B	CoT w/ taxonomy	41.04	79.99	35.54	6.29	42.33
	ARCHIVE (proposed)	**84.63**	**99.68**	**61.86**	**80.68**	**92.29**
Qwen2.5-14B	CoT w/ taxonomy	76.99	96.17	59.33	74.70	77.76
	ARCHIVE (proposed)	**85.55**	**99.79**	**65.89**	**81.73**	**94.78**

Table 6: **ARCHIVE achieves the highest F1 at the lowest latency.** Results use Qwen2.5-14B on all queries, with oracle context when available.

Method	LLM Calls	Latency (s)	F1-amb	F1-unamb
Direct	**1**	0.44	68.55	78.76
Ambig-Aware	**1**	0.44	78.57	78.17
CoT w/ taxonomy	**1**	2.13	86.72	88.16
Self-Ask	2	0.56	75.35	77.14
Frequency	10	0.18	64.99	0.00
Entropy	10	0.18	64.95	1.40
Shi et al.	10	0.47	44.78	45.65
Intent-Sim	11	0.21	65.31	39.53
ARCHIVE	3.34	**0.13**	**89.11**	**89.87**

Table 7: **Each component of ARCHIVE contributes.** Results use Qwen2.5-14B on all queries, with oracle context when available.

Variant	Macro-F1	F1-amb	F1-unamb	Latency (s)
w/o routing	53.34	30.78	68.63	0.16
w/o $\mathcal{L}_{\text{inv}}$	83.46	87.99	86.12	0.13
w/o $\tilde{S}$ ($\mathbf{h}_q$ only)	80.67	87.55	85.45	0.14
w/o gate	82.22	86.89	87.94	0.15
ARCHIVE (full)	**85.55**	**89.11**	**89.87**	**0.13**

context and thus are omitted from this comparison. First, ARCHIVE achieves the highest F1-amb and F1-unamb across both datasets and backbones. This confirms that modeling logical conflict captures ambiguity more reliably than diversity proxies. Baselines struggle to balance both metrics: direct prompting often collapses to a single class, while the frequency heuristic yields zero F1-unamb by counting paraphrases as separate answers.

5.4 4-Way Classification (Q3)

Table 5 reports 4-way classification using our taxonomy across all queries, with the LLM given oracle context. ARCHIVE outperforms CoT w/ taxonomy on macro-F1, with near-perfect performance on SDAmb queries and large gains on unambiguous classes (Closed, Open). While KDAmb queries remain most challenging, pairwise relation modeling still delivers consistent improvements, highlighting its ability to capture logical conflict.

5.5 Efficiency Analysis (Q4)

Table 6 compares inference cost and accuracy on all queries. Filtering SDAmb queries through the early-exit module, ARCHIVE uses 3.34 LLM calls per query on average, versus 10 for heuristic and sampling-based baselines. It achieves the lowest latency (0.13s) and highest accuracy, showing that relation analysis outperforms brute-force sampling.

5.6 Ablation Study (Q5)

Table 7 analyzes the contribution of each ARCHIVE component. Removing the early-exit module forces all queries through conflict modeling, increasing latency because every query invokes the LLM and removing the only route for SDAmb queries. Macro-F1 drops sharply, confirming early exit is a required routing component, not merely a speed optimization. Removing $\mathcal{L}_{\text{inv}}$ reduces both F1 metrics. Using only the query embedding $\mathbf{h}_q$ causes the largest macro-F1 drop among non-routing components, showing pairwise relations are essential. Removing the gate degrades F1-amb the most, confirming the value of channel-wise recalibration.

6 Conclusion

We propose a four-class taxonomy based on *logical conflict* and construct QUIREQA, a benchmark for its evaluation. We present ARCHIVE, a framework for accurate and efficient ambiguity detection in open-domain QA. ARCHIVE combines an early-exit module for LLM-free detection of visibly ambiguous or malformed queries, and a conflict modeling module for query-conditioned NLI over candidate answers. Experiments show that ARCHIVE improves F1-amb by up to 10.4% and F1-unamb by up to 21.6% with 16× faster inference than the highest-accuracy baseline. These results illustrate that modeling ambiguity via structured logical conflict outperforms diversity-based proxies in both detection quality and computational cost.

Limitations

We acknowledge several limitations of this work. First, ARCHIVE is evaluated only on English-language queries; extending the taxonomy and pipeline to multilingual settings remains an open direction. Second, the conflict modeling module depends on a frozen LLM to generate candidate answers. Although query-gated refinement and distractor augmentation mitigate hallucination noise, a weak backbone can still limit performance. Third, although QUIREQA aggregates multiple QA benchmarks, it may not fully capture the distribution of queries in real-world deployments: we exclude queries that are underspecified due to external dependencies. Specifically, we filter out (1) temporally evolving queries whose answers change over time (e.g., "When will the next Olympics be held?") and (2) user-dependent queries (e.g., "What's my IP address?"). This filtering keeps ground-truth annotations stable and reproducible. Finally, distinguishing knowledge-dependent ambiguity from closed unambiguity depends on the NLI model's ability to recognize genuine contradictions among candidate answers. Standard NLI training emphasizes textual entailment between sentence pairs, but our task often requires entity-level knowledge to determine whether two candidate answers are genuinely incompatible. When the NLI model lacks this knowledge, it may assign weak contradiction scores. The query-conditioned gate has limited corrective capacity in these cases because it rescales existing signals rather than introducing new knowledge. Incorporating knowledge-aware or domain-adapted NLI models is a promising direction.

Ethical Considerations

All source datasets are publicly available, contain no personally identifiable information, and were vetted by their authors. During curation, we additionally sample-checked 100 queries per source for personally identifying information and offensive content; we found none. There is an inherent risk of propagating social biases in those models because ARCHIVE relies on frozen LLMs to generate candidate answers; we manually inspected generated outputs and found no harmful content, though subtle biases may have been overlooked. ARCHIVE serves as a classification module rather than a user-facing generation system, but ambiguity detectors could be misused to suppress legitimate user intent and should be deployed with safeguards.

References

Alfonso Amayuelas, Kyle Wong, Liangming Pan, Wenhu Chen, and William Wang. 2024. Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models. *Preprint*, arXiv:2305.13712.

Reinald Kim Amplayo, Kellie Webster, Michael Collins, Dipanjan Das, and Shashi Narayan. 2022. Query refinement prompts for closed-book long-form question answering. *Preprint*, arXiv:2210.17525.

Pavel Braslavski, Denis Savenkov, Eugene Agichtein, and Alina Dubatovka. 2017. What do you mean exactly?: Analyzing clarification questions in cqa. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, pages 345–348.

Anthony Chen, Pallavi Gudipati, Shayne Longpre, Xiao Ling, and Sameer Singh. 2021. Evaluating entity disambiguation and the role of popularity in retrieval-based NLP. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4472–4485, Online. Association for Computational Linguistics.

Jeremy R. Cole, Michael J.Q. Zhang, Daniel Gillick, Julian Martin Eisenschlos, Bhuwan Dhingra, and Jacob Eisenstein. 2023. Selectively answering ambiguous questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 530–543, Singapore. Association for Computational Linguistics.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Nick Craswell, Daniel Campos, Bhaskar Mitra, Emine Yilmaz, and Bodo Billerbeck. 2020. ORCAS: 18 million clicked query-document pairs for analyzing search. *CoRR*, abs/2006.05324.

Meiqi Guo, Mingda Zhang, Siva Reddy, and Malihe Alikhani. 2021. Abg-coQA: Clarifying ambiguity in conversational question answering. In *3rd Conference on Automated Knowledge Base Construction*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *International Conference on Learning Representations (ICLR)*.

Bairu Hou, Yujian Liu, Kaizhi Qian, Jacob Andreas, Shiyu Chang, and Yang Zhang. 2024. Decomposing uncertainty for large language models through input clarification ensembling. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *CoRR*, abs/1705.03551.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen,

and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *Preprint*, arXiv:2004.04906.

Hyuhng Joon Kim, Youna Kim, Cheonbok Park, Junyeob Kim, Choonghyun Park, Kang Min Yoo, Sanggoo Lee, and Taeuk Kim. 2024. Aligning language models to explicitly handle ambiguity. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1989–2007.

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

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*.

Tom Kwiatkowski and 17 others. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

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. *Preprint*, arXiv:2309.06180.

Moritz Laurer, Wouter van Atteveldt, Andreu Casas, and Kasper Welbers. 2022. Less annotating, more classifying: Addressing the data scarcity issue of supervised machine learning with deep transfer learning and BERT-NLI. *Preprint*.

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. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11526–11544, Singapore. Association for Computational Linguistics.

Lizi Liao, Grace Hui Yang, and Chirag Shah. 2023. Proactive conversational agents in the post-ChatGPT world. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3452–3455.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

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)*, pages 5783–5797.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial nli: A new benchmark for natural language understanding. *Preprint*, arXiv:1910.14599.

OpenAI. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Adam Paszke and 20 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Preprint*, arXiv:1912.01703.

Qwen Team. 2025. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

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)*, pages 41–55.

Weiwei Sun, Hengyi Cai, Hongshen Chen, Pengjie Ren, Zhumin Chen, Maarten de Rijke, and Zhaochun Ren. 2023. Answering ambiguous questions via iterative prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7669–7683, Toronto, Canada. Association for Computational Linguistics.

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*, pages 20–30.

Md Mehrab Tanjim, Xiang Chen, Victor S Bursztyn, Uttaran Bhattacharya, Tung Mai, Vaishnavi Muppala, Akash Maharaj, Saayan Mitra, Eunyee Koh, Yunyao Li, and Ken Russell. 2025. Detecting ambiguities to guide query rewrite for robust conversations in enterprise ai assistants. *arXiv preprint arXiv:2502.00537*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *Preprint*, arXiv:1803.05355.

Hugo Touvron and 67 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. *Preprint*, arXiv:1704.05426.

Thomas Wolf and 21 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Michael JQ Zhang and Eunsol Choi. 2025. Clarify when necessary: Resolving ambiguity through interaction with LMs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5541–5558.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *Preprint*, arXiv:1904.09675.

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. *Preprint*, arXiv:2405.12063.

Table 8: Summary of notation.

Symbol	Description
q	User query
c	Optional context passage
y	Ambiguity label $\in$ {SDAmb, KDAmb, Closed, Open}
$\hat{y}$	Predicted class logits
ℓ	Binary surface-detectable ambiguous label (1 if q is SDAmb, else 0)
τ	Early-exit routing threshold
d	Encoder embedding dimension
k	Size of the candidate answer set
$\mathbf{h}_q$	Query representation $\in \mathbb{R}^d$
$\mathbf{h}_q^{\text{proj}}$	Projected query representation
$\mathbf{g}$	Query-conditioned gate vector $\in \mathbb{R}^3$
S	Pairwise NLI relation grid $\in \mathbb{R}^{k\times k\times 3}$
$\tilde{S}$	Refined relation grid (query-gated)
$\mathcal{L}_{\text{exit}}$	Binary cross-entropy loss for the early-exit module
$\mathcal{L}_{\text{CE}}$	Cross-entropy classification loss
$\mathcal{L}_{\text{inv}}$	Invariance loss
λ_{inv}	Weight for invariance loss
ϵ	Numerical stability constant

A Notation

Table 8 summarizes the paper's notation.

B QuireQA Construction

We construct QUIREQA, a benchmark of factoid and non-factoid queries under our four-class taxonomy. QUIREQA draws from four sources: ORCAS (Craswell et al., 2020) for web-search queries, including surface defects; AmbigNQ (Min et al., 2020) for factoid queries derived from Natural Questions (Kwiatkowski and 17 others, 2019) with crowdsourced disambiguations; AmbER (Chen et al., 2021) for Wikipedia-grounded entity disambiguation queries; and Dolly-15K (Conover et al., 2023) for human-written instructions with open-ended and closed-form unambiguous queries. We annotate 3,299 oracle context passages from English Wikipedia for 680 queries. Table 9 shows representative examples from each source and class.

B.1 Data Format

Each sample in QUIREQA contains a query, a taxonomy label, and, where applicable, valid interpretations; each interpretation consists of a clarification, valid answers, and optional supporting context. Knowledge-dependent ambiguous (KDAmb) samples contain multiple incompatible interpretations, whereas unambiguous samples (Closed and Open) contain a single interpretation whose answers are mutually compatible. Closed queries have a finite valid-answer set; Open queries have an unbounded answer space, so we store one representative valid answer rather than attempting full enumeration. Representative answers come from source annotations when available and are otherwise generated during dataset construction.

Surface-detectable ambiguous (SDAmb) samples do not include answer annotations because their interpretations arise from an open-ended space of plausible repairs or completions. For instance, "`where is the`" could be completed as a question about the Colosseum or the Eiffel Tower, yielding incompatible answers such as Rome or Paris. Since these cases lack a bounded set of canonical repairs, we treat SDAmb queries as requiring clarification rather than exhaustively annotating all interpretations and answers.

B.2 Annotation Protocol and Reliability

All labels and annotations are produced by the authors, who are fluent in English. The authors apply the four-class taxonomy to every query. Each query from the four sources is manually inspected and verified. To assess label reliability, two authors independently annotate a shared set of 150 queries spanning all four classes (50 SDAmb, 35 KDAmb, 35 Closed, and 30 Open). They achieve 88% agreement (Cohen's $\kappa = 0.81$). Remaining disagreements are resolved through discussion or, when necessary, adjudication by a third author.

During annotation, we exclude KDAmb candidates whose interpretations yield identical answer sets. Although such queries may be entity-ambiguous, they are not ambiguous under our conflict-based definition when the competing interpretations do not induce incompatible answers. Since all plausible interpretations support the same answer, a system can answer directly without first asking the user to disambiguate.

B.3 ORCAS Dataset

ORCAS (Craswell et al., 2020) consists of real-world user search queries. We use it primarily to source surface-detectable ambiguity, while also retaining manually verified non-SDAmb cases that satisfy the taxonomy definitions. Because the full 10.4M-query corpus is large, we use lightweight, high-recall regex heuristics to retrieve candidate surface defects. These heuristics target three non-exclusive pattern families, described below.

Table 9: Representative examples in QUIREQA by source and taxonomy class.

Source	Taxonomy Class	Example Query & Details
ORCAS	Surface-detectable ambiguous	**Query:** iuuui **Query:** what is the **Query:** who sings this
	Closed unambiguous	**Query:** what kind of word is and **Answer:** a conjunction
	Open unambiguous	**Query:** what does asdfghjkl mean **Answer:** Often used to represent random typing or keysmashing . . .
AmbigNQ	Knowledge-dependent ambiguous	**Query:** When did the Simpsons first air on television? **Answers:** April 19, 1987 (animated short) vs. December 17, 1989 (prime time show)
	Closed unambiguous	**Query:** What is the greek word for city state? **Answer:** polis
	Open unambiguous	**Query:** What kind of meat is used for shabu shabu? **Answer:** top sirloin, ribeye steak, Wagyu, . . .
AmbER	Knowledge-dependent ambiguous	**Query:** What does Billy Preston play? **Answers:** basketball (the athlete) vs. piano (the musician)
	Closed unambiguous	**Query:** Who directed The Godfather? **Answer:** Francis Ford Coppola
Dolly-15K	Open unambiguous	**Query:** How can I sleep well at night? **Answer:** Here are some suggestions for a good night's rest . . .
	Closed unambiguous	**Query:** How many planets make up the Solar System? **Answer:** 8

Table 10: **Prompt for generating representative answers for open unambiguous ORCAS queries.** The prompt produces one valid answer rather than enumerating the full answer space.

Task. Given an open-ended unambiguous query, generate a valid representative answer.
Rules.
1. Provide one plausible answer that directly addresses the query.
2. If the query requires factual information, provide an answer that is broadly correct and verifiable; do not invent unsupported details.
3. Do not attempt to enumerate all possible valid answers.
4. Keep the answer concise: one sentence or a short list is preferred.
5. Return only valid JSON: {"answer": "..."}.

Query: `{query}`
Answer:

- **Meaningless noise.** Queries consisting entirely of non-alphanumeric symbols, keyboard sequences (e.g., `asdfghjkl;'#`), consonant clusters (e.g., `fffrff`), or vowel runs (e.g., `iuuui`) that render the string unintelligible.
- **Structural incompleteness.** Queries with truncated syntax or missing arguments, including stub questions (e.g., `where are`), dangling determiners or articles (e.g., `what is the`), incomplete comparisons (e.g., `steam vs`), and fragments ending in a preposition (e.g., `when to`).
- **Referential underspecification.** Syntactically complete queries containing pronouns or definite noun phrases whose referents cannot be resolved from the query alone (e.g., `who sings this`, `what is the movie`, `can i watch it`).

Manual verification. Heuristic matching on the full dataset yields ∼124K candidate queries. We randomly sample 6,336 candidates and manually label each as a true surface-detectable case or a false positive; 1,511 are confirmed as genuine. Recurring false positives (e.g., `what kind of word is and`, `hermes is the god of`) are added to rejection lists and excluded from later sampling. Candidates filtered out as non-SDAmb are manually relabeled under the full taxonomy rather than discarded. For KDAmb and Closed cases requiring grounding, we annotate 53 oracle context passages across 46 samples. For Open cases, whose answer spaces are unbounded, we generate one representative valid answer using the prompt in Table 10.

B.4 AmbigNQ Dataset

AmbigNQ (Min et al., 2020) provides factoid questions with crowdsourced disambiguations, as well as single-answer questions from Natural Questions (Kwiatkowski and 17 others, 2019). We use the former as candidates for knowledge-dependent ambiguity and the latter as candidates for closed unambiguity.

Conflict-based relabeling criterion. AmbigNQ considers a question ambiguous if it admits multiple valid interpretations, each resolved by a disambiguated rewrite. This definition is well-suited for generating disambiguated question–answer pairs, but does not match our notion of *logical conflict* among answers. For example, the query "Where is the oldest bristlecone pine tree located?" admits answers at different geographic granularities, such as a peak, state, and country. We treat these as one interpretation because they identify the same referent at different levels of specificity. To formalize this distinction, we enforce a conflict-based criterion: two candidate answers belong to the same interpretation if both can be included in a single response without making either incorrect, irrelevant, or misleading. If including one answer renders the other false or inapplicable, they constitute distinct, incompatible interpretations. Annotators were given the following guidelines:

- **List merge:** If answers are complementary elements of the same response (e.g., filming locations for "Where was Pirates of the Caribbean 2 filmed?"), they form a single interpretation.
- **Granularity merge:** If answers identify the same referent at different levels of specificity (e.g., the peak, state, and country for "Where is the oldest bristlecone pine tree located?"), they form a single interpretation.
- **Aspect merge:** If answers describe facets of the same referent or event (e.g., the band and vocalist for "Who sings A Crazy Little Thing Called Love?"), they form a single interpretation.

Curation summary. Table 11 summarizes our curation process. We randomly sample 2,000 AmbigNQ queries to enable exhaustive manual inspection under our conflict-based criterion while maintaining balance across taxonomy classes. We exclude 321 queries whose answers change over time, leaving 1,679 AmbigNQ-derived samples in QUIREQA. Because AmbigNQ's original labels do not always match our conflict-based taxonomy, we further revise the retained samples. These revisions fall into two categories: (1) *data-quality fixes:* we annotate 2,409 oracle context passages across 196 factoid samples, correct outdated or wrong answers, and supply missing answers; (2) *label corrections:* in addition to 206 ambiguous → unambiguous relabelings, we relabel 15 samples unambiguous → ambiguous, where the original single-answer annotation overlooks a competing interpretation.

Table 11: **Summary of AmbigNQ curation.** The top block tracks sample counts from the initial pool to the retained set; the lower block reports revision operations on the 1,679 retained samples, which may overlap.

Curation	# Samples
Initial AmbigNQ samples	2,000
Excluded temporally evolving queries	321
Retained samples	1,679
Added oracle context passages	196
Corrected wrong or outdated answers	73
Added missing answers	56
Relabeled ambiguous → unambiguous	206
Relabeled unambiguous → ambiguous	15
Flagged stable time-dependent queries	28
Rewrote completed future-tense queries	13

Temporal handling. The excluded queries are those whose ground truth changes over time. We nonetheless retain 28 technically time-dependent queries whose answers are expected to remain relatively stable for the foreseeable future, and flag them for future re-validation (e.g., "The most populous country in the world is?" → India). We also rewrite 13 future-tense queries about completed events into the past tense, making their answers fixed and reproducible.

B.5 AmbER Dataset

AmbER (Chen et al., 2021) contains groups of Wikipedia entities that share a surface name, such as "Billy Preston" for both the basketball player and the musician. We instantiate queries from these entity groups using GPT-4o (OpenAI, 2024), following the prompts in Table 12. Ambiguous queries mention only the shared underspecified name and remain plausible for multiple entities in the group; we label them knowledge-dependent ambiguous because resolving the intended referent requires entity-level disambiguation. Closed unambiguous queries add a minimal descriptor (e.g., "the musician" or "the basketball player") to identify the intended referent. This construction keeps

Table 12: **GPT-4o prompts used to instantiate AmbER queries.** Labels are assigned deterministically from the AmbER entity-group structure: queries that contain a shared surface name and omit disambiguating descriptors are labeled knowledge-dependent ambiguous. Queries with disambiguating descriptors are labeled closed unambiguous.

Label	Prompt
Knowledge-dependent ambiguous	**Task:** Given an ambiguous entity name and its Wikipedia interpretations, generate a query that mentions only the shared entity name and could reasonably refer to multiple interpretations. **Rules.** 1. The query must be genuinely ambiguous across the provided interpretations and must not include disambiguating descriptors. 2. Use the associated properties to choose a plausible query relation that can apply to multiple interpretations. 3. Each answer must be extractable from the corresponding context passage. 4. Return only valid JSON: {"query": "...", "interpretations": [{"qid": "...", "wikipedia_title": "...", "clarification": "...", "answers": ["..."]}]}. **Input format.** For each ambiguous entity group: Entity name: `{name}` Interpretations, one per entity sharing the surface name: Entity: `{title of Wikipedia article}` Context: `{first 3 sentences from Wikipedia article}` Candidate relations/properties: `{property name} - {top-5 values}`
Closed unambiguous	**Task:** Given a target Wikipedia interpretation from an ambiguous entity group, generate a query that identifies the intended referent. **Rules.** 1. Add only the minimal descriptor needed to identify the intended referent, such as an occupation, work type, nationality, or domain. 2. Use the associated properties to choose a plausible query relation for the target referent. 3. Each answer must be extractable from the corresponding context passage. 4. Return only valid JSON: {"query": "...", "label": "closed unambiguous", "answers": ["..."]}. **Input format.** Shared surface name: `{name}` Target referent: `{title of Wikipedia article}` Context: `{first 3 sentences from Wikipedia article}` Associated properties: `{property name} - {top-5 values}`

query forms similar across labels while making the distinction depend on entity-level disambiguation rather than template-level artifacts. We manually verify every query for label correctness and entity plausibility, annotating 786 oracle context passages across 400 samples by checking that each passage supports the intended entity.

B.6 Dolly-15K Dataset

Dolly-15K (Conover et al., 2023) contains human-generated instruction–response pairs across diverse task categories. We specifically filter for general QA and brainstorming tasks, which primarily yield open unambiguous queries with multiple compatible responses. Queries with a closed answer space, such as a fixed list or single factual answer, are labeled closed unambiguous. The authors manually verify this closed-versus-open assignment for every query. We additionally annotate 51 oracle context passages across 38 queries.

C AmbigTriviaQA Construction

We build AmbigTriviaQA as an out-of-distribution evaluation set by deriving ambiguous questions from TriviaQA (Joshi et al., 2017), adapting the procedure of Kim et al. (2024). Each original question is first rewritten into an ambiguous form by GPT-4o using the template in Table 13. GPT-4o then re-examines every rewrite (Table 14) as an automatic quality filter, and any rewrite it does

Table 13: GPT-4o ambiguation prompt for constructing the AmbigTriviaQA dataset.

Please make the following question ambiguous. Your task is to introduce ambiguity by altering the specificity of the noun phrase or omitting crucial details from the statement. Keep the rest of the sentence unchanged except for the modified sections. Generate only the revised statement. Question: `{question}` Ambiguation:

Table 14: GPT-4o verification prompt for constructing the AmbigTriviaQA dataset.

An ambiguous question has multiple valid answers. Is the following question ambiguous with multiple possible answers? Answer only in Yes or No. Question: `{ambiguous generation}` Yes or No:

Table 15: **Example ambiguation pairs in AmbigTriviaQA.** The ambiguated phrase is highlighted in bold.

Original Question	Ambiguated Question
What is **Uma Thurman's** middle name?	What is **the** middle name?
Who directed **Back to the Future**?	Who directed **that movie**?
In which year was **CNN** founded?	When was **it** founded?

not judge ambiguous is discarded. Example ambiguation pairs are shown in Table 15. To mitigate confirmation bias from using the same model for generation and filtering, the authors review the surviving questions and curate the final set. The resulting set contains 1,000 queries, evenly split into 500 ambiguous and 500 unambiguous.

D Implementation Details

We split QUIREQA into train/validation/test sets (3,762/470/471; 8:1:1). The conflict modeling module trains on the non-SDAmb subset (∼2,540 samples per seed). At evaluation, the test split is routed end-to-end through the early-exit and conflict modeling modules. Among the test queries, 218 have oracle context passages; we call these the factoid subset. The *factoid* setting uses this 218-query subset: Table 4 follows prior work with context-free candidate-answer generation, while Table 28 compares no context, DPR-retrieved context (Karpukhin et al., 2020), and oracle context. The *all-query* setting (Tables 5, 6, and 7) covers the full test split and grounds candidate-answer generation in context when available.

The early-exit module trains a linear head on a frozen DeBERTa-v3-base encoder for 8 epochs using AdamW (Loshchilov and Hutter, 2019) with learning rate 2×10^{-4}, weight decay 0.01, and gradient clipping 1.0. The conflict modeling module keeps the encoder frozen and trains only lightweight heads for 200 epochs using AdamW (learning rate 10^{-4}, weight decay 0.01). We use batch size 16 for both modules, retain k=10 answers after deduplication, and set λ_{inv}=1.0. Pairwise relations are computed using a frozen DeBERTa-v3-base NLI model fine-tuned on MNLI (Williams et al., 2018), FEVER (Thorne et al., 2018), and ANLI (Nie et al., 2020). Candidate-answer generation uses five LLM calls at temperature 0.7. When fewer than k unique answers are produced, we fill remaining slots by frequency-proportional duplication; if parsing fails, all slots are set to `Unknown` (< 0.5% of queries). Distractor generation replaces one duplicate slot with an LLM-generated distractor sampled at temperature 0.8. We implement models in PyTorch (Paszke and 20 others, 2019) 2.10 with Hugging Face Transformers (Wolf and 21 others, 2020) 4.57.6 and vLLM (Kwon et al., 2023) 0.19.1. We run experiments on one node with 2× NVIDIA B200 GPUs.

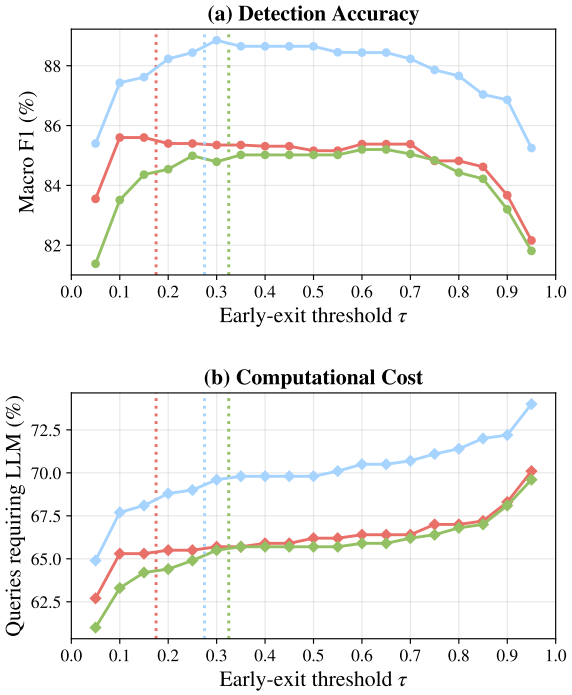


Figure 6: **ARCHIVE is robust to early-exit threshold selection.** Macro-F1 remains stable across a broad range of early-exit thresholds (a), while higher thresholds route more queries to the LLM (b). Colored curves show per-seed sweeps using Qwen2.5-14B-Instruct; dotted vertical lines indicate validation-selected thresholds; and cropped y-axes highlight threshold-dependent trends.

E Hyperparameter Sensitivity

E.1 Early-exit Threshold

Figure 6 reports per-seed test-set sweeps of the early-exit routing threshold τ over $[0.05, 0.95]$ using Qwen2.5-14B-Instruct checkpoints, with the full ARCHIVE pipeline evaluated end-to-end. The threshold trades off early-exit efficiency and detection accuracy; the operating value is selected on the

Table 16: **Prompts for candidate answer generation.** The context-grounded prompt is used when oracle context is available; otherwise, the context-free prompt is used. Three examples for each prompt are omitted for brevity.

Setting	Prompt
Context-grounded	**Task:** You will receive a question and grounding context. Produce a JSON list of candidate answers. **Rules.** 1. Identify all plausible candidate answers supported by the context. 2. If the context supports multiple dates, locations, entities, or facets, extract each as a separate candidate answer. 3. Each answer must be a short, independent, declarative sentence. 4. Return only valid JSON: {"answers": ["...", "...", ...]}. 5. The answers list must contain at least one string. 6. Do not include meta-commentary such as "The context does not provide..." or "I cannot determine...". Question: `{question}` Context: `{context}` Answers:
Context-free	**Task:** You will receive a question. Produce a JSON list of candidate answers. **Rules.** 1. Identify all plausible candidate answers using world knowledge. 2. For list-type queries, give each item as a separate candidate answer. 3. Each answer must be a short, independent, declarative sentence. 4. Return only valid JSON: {"answers": ["...", "...", ...]}. 5. The answers list must contain at least one string. 6. Do not include meta-commentary such as "This is subjective..." or "I cannot determine...". Question: `{question}` Answers:

validation split to maximize the early-exit module's binary SDAmb-detection F1. Panel (a) shows that 4-way macro-F1 remains stable across a broad middle range of τ and degrades at the extremes. At low τ, the early-exit module routes too aggressively and many non-SDAmb queries are incorrectly early-exited as SDAmb. At high τ, genuine SDAmb queries bypass early exit and reach the conflict modeling module, which has no SDAmb class and misclassifies them. All three seeds exhibit the same qualitative pattern, indicating that the plateau and extreme-case degradation reflect the routing mechanism rather than a seed-specific artifact. Panel (b) shows that the fraction of queries requiring LLM inference rises with τ in every seed. The validation-selected thresholds (mean $\tau \approx 0.26$, range 0.175–0.325) fall within each seed's accuracy plateau, achieving mean macro-F1 of $85.37 \pm 2.42\%$. This indicates that validation tuning selects thresholds aligned with downstream performance.

E.2 Strength of Invariance Loss

We sweep the weight of the invariance loss $\lambda_{\text{inv}} \in \{0, 0.5, 1.0, 1.5, 2.0\}$ on the QUIREQA test split using the same three Qwen2.5-14B-Instruct seeds as in the main experiments. Figure 7 reports per-seed 4-way macro-F1. We make two observations. First, the invariance loss improves performance: λ_{inv}=1.0 outperforms λ_{inv}=0 for all three seeds and yields the best cross-seed mean in the sweep.

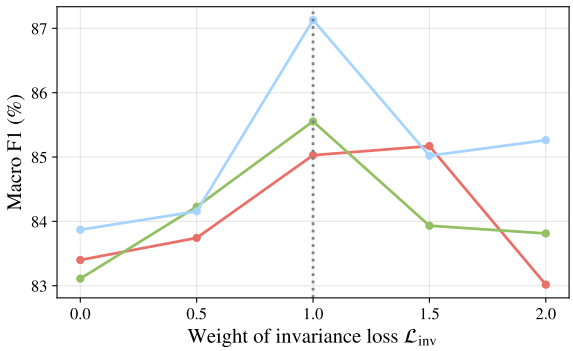


Figure 7: **ARCHIVE benefits from the invariance loss, with λ_{inv}=1.0 providing a stable high-performing choice.** Colored curves show per-seed 4-way macro-F1 on the QUIREQA test split using Qwen2.5-14B-Instruct; the dotted vertical line marks the selected value.

This trend is consistent with the "w/o $\mathcal{L}_{\text{inv}}$" row of Table 7. Second, ARCHIVE does not require fine-grained tuning of λ_{inv}: nearby nonzero values stay close to the best setting, although large weights degrade some seeds. We select λ_{inv}=1.0, which lies in a high-performing region, suggesting that the invariance objective provides robust regularization.

Table 17: **Prompt for generating distractors.** The `Context:` field is included only when oracle context is available.

Task. Generate three distractor answers for the given question. Each should be plausible, but verifiably invalid with respect to the provided valid answers and, when available, context.
Allowed distractor types.
1. Partial answer: Omit a required component from a multi-part valid answer.
2. Role swap or misattribution: Reassign entities from the context to incorrect roles or relationships.
3. Controlled perturbation: Modify a date, number, name, or attribute appearing in the context or valid answers.

Constraints.
- Do not introduce real-world entities, items, or facts absent from the provided information.
- Do not generate paraphrases, close synonyms, or valid subsets of the valid answers.
- Each distractor must be grounded in the provided information but altered so that it is invalid.
- Match the style, tone, and sentence structure of the valid answers.
- Use at least two distinct distractor types across the three outputs.
- Return only valid JSON: {"distractors": ["...", "...", "..."]}.

Question: `{question}`
Context: `{context, if available}`
Generated Valid Answers: `{valid answers}`
Distractors:

Table 18: **Example distractors.** One representative distractor per query; the bold span marks the perturbation that makes it invalid.

Query	Valid Answer(s)	Distractor
Knowledge-dependent ambiguous		
When did the Simpsons first air on television?	April 19, 1987 (animated short), December 17, 1989 (prime time show)	The Simpsons debuted as a half-hour show on December 17, **1990**.
Closed unambiguous		
Who directed the film New York, New York?	Martin Scorsese	**Earl Mac Rauch** directed the film New York, New York.
Open unambiguous		
How can I sleep well at night?	Maintain a consistent sleep schedule Avoid caffeine and large meals before bed	You can sleep well at night by going to bed and waking up **at different times every day.**

F Generation Prompts

Our pipeline uses three system-level LLM prompts: two answer-generation prompts (Table 16), selected by the availability of context, and one distractor-generation prompt (Table 17). Both answer-generation prompts include three examples. In the grounded setting, the user message supplies the query and passage; the prompt asks the LLM to extract all answers supported by that passage. In the context-free setting, the LLM is instructed to generate candidate answers from world knowledge.

We generate distractors using the same frozen LLM and the prompt in Table 17. Given the query, passage, and valid answers produced by the candidate-answer generation prompt, the LLM generates distractors that match the surface form, style, and format of valid answers but are factually invalid given the query and passage. For each training example, we randomly select one of the three generated distractors to replace a duplicate answer slot, yielding the augmented answer set. In QUIREQA, candidate generation always yields fewer than k=10 unique answers. Consequently, every answer set contains at least one duplicate slot. Distractors apply only to the three routed classes; surface-detectable ambiguous queries exit early. We show examples of distractors in Table 18.

G Taxonomy Utility Experiment

In Section 5.2, we evaluate whether the taxonomy improves ambiguity classification and response quality over existing taxonomies. We compare ARCHIVE against three alternatives: symptom-based (Zhang et al., 2024), action-oriented (Tang et al., 2025), and linguistic (Tanjim et al., 2025). We exclude Min et al. (2020), Guo et al. (2021), and Amplayo et al. (2022), whose taxonomies target dataset-specific subsets rather than general prompt-based ambiguity detection. We use per-taxonomy class descriptions from each paper (Table 19), embedded in a shared prompt scaffold (Table 20), varying only the `Taxonomy:` block for a

Table 19: **Taxonomy definitions used in the utility experiment.** The four `Taxonomy:` blocks are substituted into the scaffold of Table 20. The (A)/(U) tags shown here mark the ambiguous/unambiguous mapping used for binary F1; they are added for clarity and are not part of the prompt seen by the model.

Taxonomy	Class Definitions
ARCHIVE (proposed)	**(A) Surface-detectable ambiguity:** The query is ambiguous, incomplete, malformed, or underspecified in a way that is detectable from the query text alone, without external knowledge. **(A) Knowledge-dependent ambiguity:** The query is well-formed, but recognizing its multiple legitimate interpretations requires external or world knowledge. **(U) Closed unambiguity:** The query is clear and has a single definitive answer. No clarification is needed. **(U) Open unambiguity:** The query is clear and allows diverse but compatible valid answers, such as advice, examples, or lists. No clarification is needed.
Symptom-based (Zhang et al., 2024)	**(A) Unfamiliar:** The query contains unfamiliar entities or facts. **(A) Contradiction:** The query contains self-contradictions. **(A) Lexical:** The query contains terms with multiple meanings. **(A) Semantic:** The query lacks context, leading to multiple interpretations. **(A) Who:** The query is underspecified with respect to a person, entity, or agent needed to answer it. **(A) When:** The query is underspecified with respect to time or temporal scope. **(A) Where:** The query is underspecified with respect to place, location, or spatial scope. **(A) What:** The query is underspecified with respect to the requested information, object, or task. **(U) Unambiguous:** The query is clear, specific, and unambiguous.
Action-oriented (Tang et al., 2025)	**(A) Semantic:** The query is semantically ambiguous for several common reasons: it may include homonyms; a word in the query may refer to a specific entity while also functioning as a common word; or an entity mentioned in the query could refer to multiple distinct entities. **(A) Generalize:** The query focuses on specific information; however, a broader, closely related query might better capture the user's true information needs. **(A) Specify:** The query has a clear focus but may encompass too broad a research scope. It is possible to further narrow down this scope by providing more specific information related to the query. **(U) Unambiguous:** The query is clear, specific, and unambiguous.
Linguistic (Tanjim et al., 2025)	**(A) Pragmatic:** The meaning of the query depends on missing context, reference, or scope. **(A) Syntactic:** The structure of the query is incomplete or allows multiple interpretations. **(A) Lexical:** The meaning of a word or term in the query is unclear or has multiple interpretations. **(U) Unambiguous:** The query is clear, specific, and unambiguous.

controlled comparison. For Zhang et al. (2024)'s Who/When/Where/What classes, we rephrase the original template ("Query output contains confusion due to missing . . . elements") as "The query is underspecified with respect to . . ." for clearer LLM prompting. Because the baseline taxonomies do not define an unambiguous class, we use a common description for it across all schemas.

We evaluate each taxonomy on two outcomes: binary ambiguity detection and response quality. For each taxonomy, Qwen2.5-14B-Instruct is run once per query with oracle context, using greedy decoding and a 150-token cap. The model outputs a taxonomy class and response: a direct answer for unambiguous predictions or a clarifying question for ambiguous predictions. We map each predicted class to its binary label and compute F1. We evaluate response quality only for classes with well-defined references: closed unambiguous queries are scored with BERTScore against gold answers, and knowledge-dependent ambiguous queries with CQ BERTScore against pseudo-references.

Table 20: **Shared prompt scaffold for the taxonomy utility experiment.** The `Taxonomy:` placeholder is replaced with taxonomy class definitions from Table 19.

```
Classify the query using the provided taxonomy, then produce the corresponding response.
If the query is unambiguous, answer it directly.
If the query is ambiguous, ask a clarifying question that would help resolve the ambiguity.
You must format your output exactly as:
    Classification: <class>
    Response: <answer or clarifying question>
Taxonomy: {taxonomy-specific class definitions}
Query: {query}
Context: {context, if available}
Classification:
Response:
```

Pseudo-references for clarifying questions. Clarifying questions admit many valid phrasings, so we evaluate them with BERTScore against pseudo-references. We generate each pseudo-reference by applying the clarification prompt of Kim et al. (2024) in Table 21 to each query and its gold disambiguation. Because the prompt is conditioned on the annotated disambiguation, the pseudo-reference targets the intended ambi-

Table 21: **Prompt for generating pseudo-reference clarifying questions.** We adapt the prompt of Kim et al. (2024) to generate a pseudo-reference from each ambiguous query and its gold disambiguation for BERTScore evaluation.

```
Engage with the provided ambiguous question by extracting the key point of ambiguity, and interactively ask for clarification based on the disambiguated question.

Example 1:
Ambiguous Question: Who won?
Disambiguation: Who won the 2020 U.S. presidential election?
Clarification Request: Your question seems ambiguous. Could you specify which competition or event you are asking about?

Example 2:
Ambiguous Question: What's the weather like?
Disambiguation: What's the weather like in Miami today?
Clarification Request: Your question is ambiguous. Where are you interested in the weather report for?

Ambiguous Question: {query}
Disambiguation: {gold disambiguation}
Clarification Request:
```

Table 22: **Examples of pseudo-references for clarifying questions.** Knowledge-dependent ambiguous (KDAmb) references target concrete disambiguations, whereas surface-detectable ambiguous (SDAmb) references reduce to generic rephrasing requests.

Class	Query	Generated reference CQ
KDAmb	When did the Simpsons first air on television?	Could you please specify whether you mean its debut as an animated short or its first airing as a half-hour prime-time show?
	What does Billy Preston play?	Could you specify whether you mean the singer-songwriter or the professional basketball player?
SDAmb	iuuui	Could you provide more context or specify what you would like to know?
	what is the	Could you please specify what exactly you would like to know or learn about?

guity while BERTScore tolerates benign surface variation. CQ BERTScore is computed only for knowledge-dependent ambiguous queries: clarifications for surface-detectable queries collapse to generic "what do you mean?" rephrasings (see Table 22 for examples). Ans BERTScore is defined only for closed unambiguous queries because open unambiguous queries have an unbounded answer space that no finite gold set can enumerate.

Table 23: Direct baseline prompt.

```
Answer the following question.
Question: {question}
Context: {context, if available}
Answer:
```

Table 24: Ambig-Aware baseline prompt.

```
Answer the following question.
If the question is ambiguous, it is proper to answer with "The question is ambiguous".
Question: {question}
Context: {context, if available}
Answer:
```

Table 25: Self-Ask step 2 prompt.

```
Given the question and answer, is the question ambiguous or unambiguous?
Answer only ambiguous or unambiguous.
Question: {question}
Answer: {generated_answer}
Ambiguous or Unambiguous:
```

H Competitor Details

We provide detailed descriptions of each baseline. For all methods, oracle context is inserted after the question when available.

Direct. Following Kim et al. (2024), the LLM receives a plain QA prompt. A query is detected as ambiguous if the model's greedy generation contains any of the predefined ambiguity-indicating phrases—"ambiguous", "ambig", "unclear", "not clear", "not sure", "confused", "confusing", "vague", "uncertain", "doubtful", "doubt", "questionable", and "clarify"—and unambiguous otherwise. The prompt is shown in Table 23.

Ambig-Aware (Kim et al., 2024). The prompt, as shown in Table 24, explicitly describes how to handle ambiguity.

Self-Ask. A two-step approach detects ambiguity after answer generation. In step 1, the LLM generates a free-form answer to the query using the Direct baseline prompt (Table 23). In step 2, the LLM receives the query and its own answer and judges whether the query is ambiguous (Table 25). Following Kim et al. (2024), we modify the prompt from Amayuelas et al. (2024) so that the model focuses specifically on ambiguity.

CoT w/ taxonomy. The baseline prompts the LLM with ARCHIVE's four-class taxonomy and

Table 26: **CoT w/ taxonomy baseline prompt.** Examples are omitted for brevity.

Classify the following query using the ARCHIVE taxonomy. First reason step by step about whether the query is ambiguous and, if so, whether the ambiguity is surface-detectable or knowledge-dependent. Then output the final classification.

Taxonomy
1. Surface-detectable ambiguity: The query is ambiguous, incomplete, malformed, or underspecified in a way that is detectable from the query text alone, without external knowledge.
2. Knowledge-dependent ambiguity: The query is well-formed, but recognizing its multiple legitimate interpretations requires external or world knowledge.
3. Closed unambiguity: The query is clear and has a single definitive answer. No clarification is needed.
4. Open unambiguity: The query is clear and allows diverse but compatible valid answers, such as advice, examples, or lists. No clarification is needed.

Task Query: `{query}`
Context: `{context, if available}`
Rationale:
Classification:

Table 27: Frequency and entropy heuristics prompt.

Provide all plausible responses to the question.
Question: `{question}`
Context: `{context, if available}`
Answer(s):

four in-context examples, one per class. As Table 26 shows, the LLM outputs a step-by-step justification followed by a classification label.

Frequency heuristic. Following the frequency-based heuristic of Shi et al. (2025), the LLM generates 10 answers at temperature 0.7 using the prompt in Table 27. The most frequent answer is identified, and the query is deemed ambiguous if this answer accounts for less than a tuned threshold fraction of all responses.

Entropy heuristic. This baseline uses the same answer-oriented prompt and sampling procedure as the frequency heuristic (Table 27). It computes the Shannon entropy of the sampled answer distribution and predicts ambiguity when entropy exceeds a validation-tuned threshold. High entropy indicates many distinct answers with similar frequencies; low entropy indicates one dominant answer.

Intent-Sim (Zhang and Choi, 2025). For each query, the LLM first generates a clarifying question and then samples 10 hypothetical user responses at temperature 0.5. We cluster the responses by pairwise entailment using DeBERTa-large fine-tuned on MNLI. We classify queries as ambiguous when intent entropy exceeds a validation-tuned threshold.

APA (SFT) (Kim et al., 2024). This self-supervised fine-tuning baseline uses answer variability as a proxy for ambiguity. For each query, the LLM generates candidate answers and clarification questions, then fine-tunes on its own generated labels to predict ambiguity.

Shi et al. (RF) (Shi et al., 2025). This feature-engineering baseline trains a random forest on diversity features from sampled LLM responses. The LLM samples 10 responses across three temperatures: 3 at 0.3, 4 at 0.5, and 3 at 0.7. The random forest uses lexical overlap, semantic similarity, and frequency statistics to classify ambiguity.

I Context Quality Sensitivity

Table 28 compares ambiguity detection under no context, DPR retrieval (Karpukhin et al., 2020), and oracle context on factoid queries. ARCHIVE generally benefits from stronger grounding, with the strongest absolute scores under oracle context on both backbones. It achieves the best F1-amb in every setting and the best F1-unamb in five of six cells; CoT w/ taxonomy is the main competitor under DPR retrieval, surpassing ARCHIVE on F1-unamb for Qwen2.5-14B (83.02 vs. 80.51). F1-unamb dips below the no-context level under DPR for both backbones, indicating that imperfect retrieval introduces noise on unambiguous queries.

J Detailed NLI Relation Grids

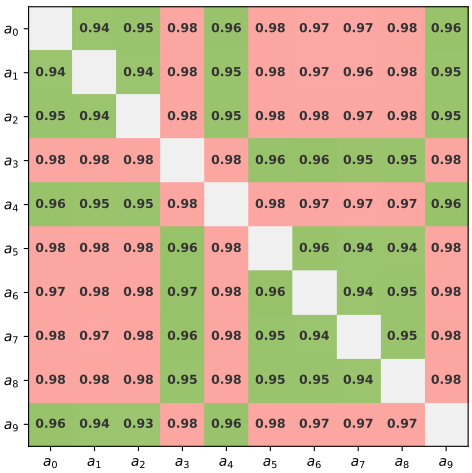
Figure 8 maps grid indices to generated answers for each of the three example queries. Each cell is colored by the highest-probability NLI class. The underlying $\tilde{S}$ retains the full distribution over $\{\text{entailment}, \text{neutral}, \text{contradiction}\}$ for each cell.

Table 28: **Context improves relation-based ambiguity detection.** Each cell reports F1-amb or F1-unamb on factoid queries. The no-context setting is omitted for Shi et al. (2025) as their method assumes context is available.

Method	No Context		DPR top-3		Oracle Context	
	F1-amb	F1-unamb	F1-amb	F1-unamb	F1-amb	F1-unamb
Backbone: *LLaMA-2-7B*						
Direct	2.77	59.52	2.66	58.39	6.23	79.99
Ambig-Aware	49.70	0.00	52.70	0.00	55.15	3.04
Self-Ask	33.98	64.04	33.90	61.75	53.68	68.89
CoT w/ taxonomy	49.85	4.01	54.44	12.94	64.62	24.49
Frequency	49.70	0.00	54.92	0.00	53.70	0.00
Entropy	48.80	32.14	49.34	1.66	55.05	34.19
Intent-Sim	41.84	66.57	0.29	68.32	45.35	79.99
Shi et al.	—	—	43.25	45.29	53.56	54.09
APA	52.24	20.94	56.14	63.48	59.45	60.52
ARCHIVE (proposed)	**58.96**	**85.57**	**62.33**	**77.48**	**68.31**	**87.62**
Backbone: *Qwen2.5-14B*						
Direct	3.74	68.76	27.49	56.80	53.82	82.23
Ambig-Aware	45.32	69.18	48.35	76.22	66.94	72.08
Self-Ask	34.51	72.43	35.39	72.59	74.31	75.75
CoT w/ taxonomy	45.27	69.87	62.32	**83.02**	60.55	83.79
Frequency	49.70	0.00	54.29	0.00	65.02	0.28
Entropy	49.67	2.42	49.96	17.39	64.92	7.49
Intent-Sim	48.68	15.91	49.60	39.86	66.22	43.83
Shi et al.	—	—	43.57	39.51	50.60	51.18
APA	50.88	64.05	44.41	70.15	64.05	72.87
ARCHIVE (proposed)	**61.31**	**85.66**	**76.83**	80.51	**76.84**	**88.35**

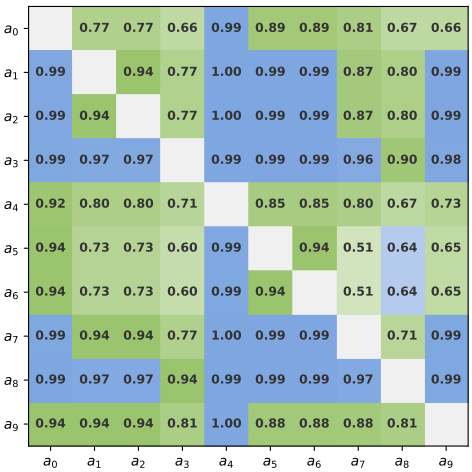
K Licensing

AmbigNQ (Min et al., 2020) does not list a separate license in its repository; it is built from Natural Questions, and we cite the original Natural Questions creators as requested by the AmbigNQ repository. AmbER (Chen et al., 2021) releases its data under CC0 1.0 and its code under the Apache License 2.0. ORCAS (Craswell et al., 2020) is released by Microsoft for non-commercial research use only, without extending any license or other intellectual-property rights. Dolly-15K (Conover et al., 2023) is licensed under CC BY-SA 3.0. TriviaQA (Joshi et al., 2017) code is released under the Apache License 2.0; the underlying questions and documents remain under the copyright of their original authors. LLaMA-2 (Touvron and 67 others, 2023) is released under the Llama 2 Community License Agreement. Qwen2.5 (Qwen Team, 2025) (14B-Instruct) is released under the Apache License 2.0. DeBERTa-v3 (He et al., 2023) is released under the MIT License. All assets are used in accordance with their respective licenses and terms. We release QUIREQA for non-commercial research use only, subject to the most restrictive applicable terms of its constituent sources. In particular, QUIREQA includes 1.6K relabeled ORCAS-derived queries with original query identifiers; these samples remain subject to Microsoft's ORCAS terms, and we do not extend any additional license or intellectual-property rights over the underlying ORCAS content.



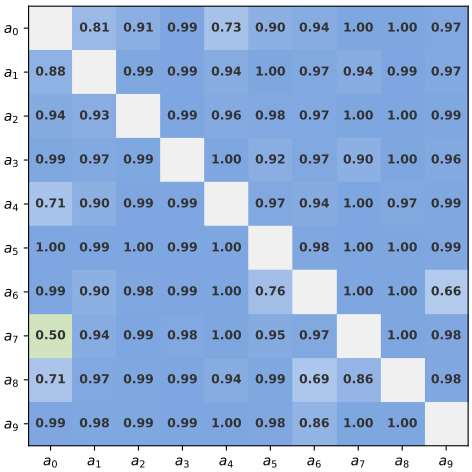
Knowledge-dependent ambiguous
Query: What is the capital of Georgia?

a_0: Atlanta is the capital of Georgia.
a_1: Atlanta serves as the capital of Georgia.
a_2: The capital city of Georgia is Atlanta.
a_3: Georgia's capital is Tbilisi.
a_4: Georgia's capital is Atlanta.
a_5: The capital of Georgia is Tbilisi.
a_6: Tbilisi is the capital of Georgia.
a_7: Tbilisi serves as the capital of Georgia.
a_8: The capital city of Georgia is Tbilisi.
a_9: The capital of Georgia is Atlanta.



Closed unambiguous
Query: Where is the Colosseum?

a_0: The Colosseum is in Rome.
a_1: The Colosseum is in Italy.
a_2: The Colosseum is in Italy.
a_3: The Colosseum is in Europe.
a_4: The Colosseum is in central Rome.
a_5: The Colosseum is in the city of Rome.
a_6: The Colosseum is in the city of Rome.
a_7: The Colosseum is in the country of Italy.
a_8: The Colosseum is in the continent of Europe.
a_9: The Colosseum is in Rome, Italy.



Open unambiguous
Query: How to organize a desk?

Each answer begins with "A desk can be organized by".
a_0: using storage containers.
a_1: regularly cleaning the surface.
a_2: using vertical space such as shelves or organizers.
a_3: aligning items neatly to maintain visual order.
a_4: keeping cables managed and untangled.
a_5: removing unnecessary items to reduce clutter.
a_6: grouping related items together.
a_7: storing rarely used items away from the main surface.
a_8: using a drawer organizer for small supplies.
a_9: dedicating specific zones for different tasks.

Figure 8: **Refined relation grids reveal whether candidate answers conflict or merely differ.** For knowledge-dependent ambiguity (top), competing candidate answers (Atlanta vs. Tbilisi) form a distributed contradiction pattern; for closed unambiguous queries (middle), compatible answers at varying granularity mostly entail or remain neutral; and for open unambiguous queries (bottom), diverse valid answers are largely neutral.