AOI :10.100.234513.0234
13th international conference on Microelectronics, Circuits and Systems, Micro2026

Detect-Then-Clarify: A Hybrid Approach for Identifying and Resolving Ambiguous User Queries

Shefali Kaushal, Tanvi Saroha, Saie Pawar, Shikha Bharadwaj and Vandana Niranjan

Department of Electronics and Communication Engineering
Indira Gandhi Delhi Technical University for Women
Kashmere Gate, Delhi 110006, India
Corresponding Author: shefali114bteceai22@igdtuw.ac.in

ABSTRACT

Ambiguity in natural language queries continues to be a significant barrier to accurate question answering and conversational systems, resulting in many incorrect or unreliable answers. The prevailing methods, ranging from AmbigQA style LLM pipelines to rule-based systems such as TaskLint, generally rely on large models that require powerful GPU capabilities or rigid taxonomy-based structures; therefore, usability and real-world practicality are inhibited by these limitations. Detect-Then-Clarify is an agile and lightweight system for detecting and clarifying ambiguities via the detection, classification, and clarification processes, reducing the reliance on large language models or expensive computational resources. Experimental evaluation demonstrates that this system effectively distinguishes between ambiguous requests, resolves ambiguities through detailed exchanges, and builds user trust efficiently while minimizing unnecessary communication between parties.

Keywords: Ambiguity Detection, Conversational AI, Clarification Dialogue, BERT-based Classification, Question Answering Systems, AmbigQA Dataset

I. INTRODUCTION

Due to continuous change in the domain of language processing and conversational AI, the ability to exactly interpret the user's intent is becoming a critical determinant of system effectiveness. Recent research points out that 30–40 percent of users ask queries in an open domain having some type of ambiguity [1] (Min et al., 2020), leading to diminished trust among the user due to misinterpretations and irrelevant responses. The different types of ambiguity, such as multiple meanings to words, generate lexical ambiguity, referential ambiguity occurs when there are unclear antecedents to references or pronouns, an unspecified time frame gives rise to temporal ambiguity, and structural ambiguity is a result of various interpretations to a question [8]. Both techniques are inadequate if we consider the area of customer service or healthcare information retrieval.

The recent development in transformer-based language models shows amazing abilities in comprehending contextual nuances and semantic relationships [2]. Although the literature in neural models dealing with uncertainty quantification shows that prediction confidence does not go hand in hand with interpretive accuracy arising due to inherently ambiguous queries, there arises the need to identify unclear queries through explicit ambiguity detection before the answer generation. One rule-based system working on identifying vagueness in the instructions through predefined linguistic patterns is Task Lint; however, it lacks the contextual understanding and adaptability for diverse conversational scenarios [3]. The ambiguity identification of neural approaches comes under a binary classification task, which lacks resolution, while the APA (Asking for Perspective) framework works mainly at the theoretical level without practical application [4]. The existing literature lacks a hybrid approach, which is an amalgam of rule-based clarification along with pattern recognition capabilities of deep learning. This study provides a solution to these gaps by presenting a novel "Detect Then-Clarify" framework, which combines transformer-based ambiguity detection and intelligent clarification question generation [1], [3]–[6].

The research is guided by three fundamental questions. First, whether a fine-tuned BERT model differentiates ambiguous and non-ambiguous user queries [2] using confidence-based and structural features. Second, whether classification performance can be achieved on real-world question-answering data if ambiguity arises due to synthetic construction and not naturally. Third, how the detected ambiguity can be translated into contextually appropriate clarification questions that facilitate resolution without overwhelming the users. The current approach will enable AI systems to engage in genuine clarification dialogues rather than forcing guesses or overwhelming users with multiple responses. The contribution to this field will be threefold. First, we develop a sturdy ambiguity labeling framework that leverages both structural indicators and confidence scores from the AmbiNQ dataset by creating a binary classification task [1]. Second, we demonstrate that a fine-tuned BERT-base model achieves competitive ambiguity detection performance under class-imbalanced conditions with F1 scores exceeding 0.64 on unseen data [2]. Third, we implement a rule-based clarification module that generates contextually relevant questions from detected ambiguity patterns, providing a solution from detection to resolution. Methodologically, this study adopts a hybrid approach that combines supervised learning on 10,036 annotated

ISBN: 978-81-985770-0-9

question-answer pairs from the AmbigQA training subset with heuristic clarification generation rules [1].

II. LITERATURE REVIEW

Earlier studies focused on vagueness and ambiguity in conversational instructions from both a task-oriented and question-answering point of view. TaskLint is a forecasting, governed system inspired by static code inspection to anticipate ambiguity in task directives, especially in crowdsourcing and human-in-the-loop settings. It uses established linguistic criteria to identify potentially ambiguous instructions. The experimental results demonstrate that updating tasks with TaskLint feedback enhances accuracy and consistency, emphasizing the importance of early ambiguity recognition. However, TaskLint is a static analytical tool and does not offer engagement, ambiguity resolution, or learning-based generalization [3].

Kim et al. indicated Alignment with Perceived Ambiguity (APA), which intends to explicitly recognize ambiguous queries via the inner uncertainty of large language models. APA uses supervised fine-tuning to encourage models to generate inquiries about clarification rather than straight replies by measuring the information gain between an input and its self-explanatory version. While APA improves ambiguity awareness and maintains efficacy on unambiguous inputs across various QA datasets, it focuses on model-side uncertainty calibration over end-to-end ambiguity resolution via interactive interaction between users [4]. Similarly, new research on ambiguity identification and uncertainty calibration in QA with LLMs utilizes answer distribution patterns and lightweight classifiers to identify ambiguous queries and enhance confidence estimation [6]. Although such techniques improve ambiguity detection accuracy, they mostly concentrate on concern recognition rather than structured resolution or multi-turn classification.

At the same time, AMBIGQA promotes ambiguity-aware open-domain question answering by requiring systems to enumerate plenty of probable interpretations and offer minimal clarified question rewrites, which is aided by the large-scale AMBIGNQ dataset [1]. Related clarification-based quality assurance systems include records like CAMBIGNQ and design workflows that explicitly ask clarification questions before producing final answers, proving the worth of interactive clarification in connecting responses to user intent [5], [12]. However, these approaches mainly deal with dataset construction, offline disambiguation, or separate clarifying methods rather than integrating detection, reasoning, and resolution into the same system. In contrast, our research combines transformer-based detection of ambiguity with contextually targeted clarification in a Detect-then-Clarify workflow. By combining neural ambiguity recognition, user-facing multi-turn interaction, and post-clarification performance evaluation, our method provides a comprehensive and practical model for real-time ambiguity resolution that is not addressed in the aforementioned experiments.

III. PROPOSED WORK

3.1 Problem Formulation

Let a user query be represented as a sequence of tokens:

$$x = \{w_1, w_2, \ldots, w_n\} \quad (1)$$

where w_i denotes the i^{th} word in the query and n is the total number of tokens.

The goal of the system is to determine if there is an ambiguity in the query x. If there is ambiguity, then the system creates a clarification question Q^c to gain further information from the user; otherwise, if there is no ambiguity in the query, it passes the query on to an answer generation module to create a response A(x). The final output of the system is given by (2):

$$y = \{ Q^c, \text{ if } x \text{ is ambiguous; } A(x), \text{ otherwise} \quad (2)$$

3.2 Dataset and Features

The AmbigQA dataset is used to evaluate the proposed framework, originally containing 14,042 ambiguous open-domain questions, from which a training subset was used, along with their disambiguated rewritten versions, sourced from the NQ-OPEN benchmark [1], [7]. The original queries are treated as ambiguous samples and the rewritten versions as clear samples, enabling supervised binary ambiguity detection consistent with the class distribution. The dataset includes many types of ambiguity in the real-world; these types are:

- Event ambiguity (39%) – Where events are unclear
- Property ambiguity (27%) – Where the entity's attributes are ambiguous
- Entity ambiguity (23%) – Where an entity could refer to more than one entity
- Answer-type ambiguity (16%) – Where the answer could be in many forms of the expected answer in an ambiguous manner
- Temporal ambiguity (13%) – When a time frame is unspecified
- Multi-part ambiguity (3%) – Where there are multiple sub-questions that are unresolved

The extraction of ambiguity detection and classification information from queries requires the use of five different features for each query:

- Query Length (|x|): consists of the number of tokens in the query.

ISBN: 978-81-985770-0-9

- Ambiguous Word Count (A(x)): the number of words that induce ambiguity in a query (e.g., recent, many, large, and near) indicating that there is an underspecified intent to the statement.
- Temporal Marker Count (T(x)): total number of vague expressions in the query that relate to time (e.g., last year, just recently).
- Numeric Vagueness Count (N(x)): number of words that are used in the query that indicate imprecision for any numerical term (e.g., a couple of, two or three).
- Named Entity Count (E(x)): the number of entities detected in the query, such as people, locations, or organizations.
- Pronoun Count (P(x)): Occurrence of referential pronouns (e.g., this, that, it).

These features are extracted using lightweight natural language processing techniques and are employed to calculate ambiguity confidence score and to classify the type of ambiguity present within each query [8].

3.3 Ambiguity Detection via Confidence Scoring

A fine-tuned BERT-based classifier with feature-assisted confidence scoring is used to detect ambiguity in user queries. This design is based on the fact that ambiguous question queries frequently have vague quantifiers, ambiguous antecedents, and imprecise references to time.

3.3.1 Ambiguity Cue Density

Four types of lexical indicators are defined: general ambiguity cues V^{amb}, temporal markers $V_{te}{}^{m}{}_{p}$, numeric vagueness terms $V_{n}{}^{um}$, and referential pronouns $V_{re}{}^{f}$. The general density of the ambiguity cue can then be computed using (3):

$$A(x) = (1/n)\, \Sigma\, I(w_i \in V^{amb} \cup V_{te}{}^{m}{}_{p} \cup V_{n}{}^{um} \cup V_{re}{}^{f}) \quad (3)$$

where I(·) denotes the indicator function.

3.3.2 Component Scores

To differentiate distinct sources of ambiguity, three normalized sub-scores are computed using (4)–(6):

$$T(x) = (1/n)\, \Sigma\, I(w_i \in V_{te}{}^{m}{}_{p}) \quad (4)$$

$$N(x) = (1/n)\, \Sigma\, I(w_i \in V_{n}{}^{um}) \quad (5)$$

$$R(x) = (1/n)\, \Sigma\, I(w_i \in V_{re}{}^{f}) \quad (6)$$

capturing temporal, numeric, and referential under specification, respectively.

3.3.3 Confidence Aggregation and Decision Rule

The final ambiguity confidence is computed using (7):

$$C(x) = \alpha A(x) + \beta T(x) + \gamma N(x) + \delta R(x) \quad (7)$$

where α, β, γ, δ are empirically tuned weights. The final ambiguity prediction combines BERT classifier output with confidence score thresholding as:

$$\hat{y} = \{\ 1,\ C(x) \geq \tau\ ;\ \ 0,\ C(x) < \tau \quad (8)$$

with threshold τ [9].

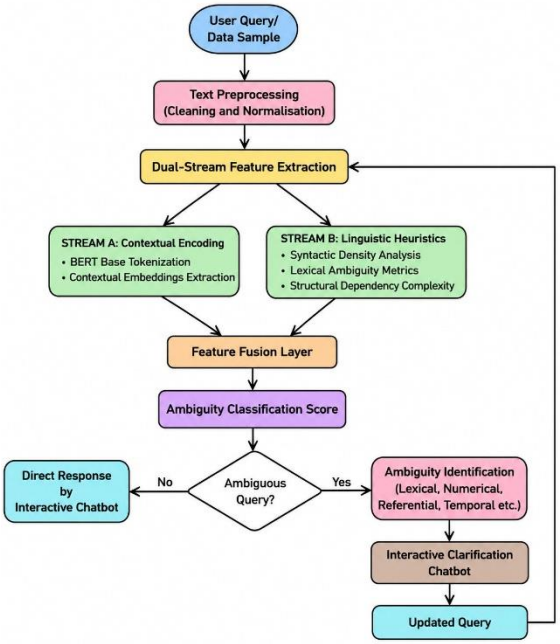

Fig. 1. The end-to-end architecture of the proposed Detect-then-Clarify framework includes preprocessors, dual stream feature extractors, BERT-based hybrid ambiguity classifier and type identification, and interactive clarification with refinement of queries before generating a final response.

This approach strictly follows the detect-then-clarify policy by intercepting underspecified queries before answer generation.

3.4 Classification of Ambiguity Types

Under-specified queries are further analyzed to find the exact cause of under specification, thus allowing the clarification module to produce follow-up queries based on that. The system employs a compact taxonomy K = {Lexical, Referential, Temporal, Numeric, Scope, Pragmatic}, which captures the ambiguity patterns seen in the AmbigQA corpus and implemented in the notebook.

Each query x is characterized by the feature vector f(x) extracted in the previous step. A rule-based scoring function $S_k(f(x))$ is specified for each class $k \in K$, and the predicted type is computed using (9):

$$\hat{k} = argmax_{k \in K}\, S_k(f(x)) \quad (9)$$

Linguistically based heuristics are used to score potential classification based on the following ten scoring criteria:

- Temporal: Selected when the count of temporal markers T(x) is predominant.
- Numeric: Picked when the number of numeric vagueness N(x) is significant.
- Referential: Selected when the frequency of pronouns P(x) is high and the number of named entities E(x) is low.
- Lexical: Activated by polysemous terms depending on context.
- Scope: Picked when there are no explicit constraints or ranges.

ISBN: 978-81-985770-0-9

- Pragmatic: Assigned when there is a need for additional discourse context.

After determining which type of vagueness exists from the scoring, the predicted class $\hat{k}$ will proceed to the clarification-generation stage to aid in type-specific interactions.

3.5 Clarification Question Generation

For each predicted ambiguity type $\hat{k}$, the system generates a specific clarification query based on that class using predefined templates, consistent with prior clarification-based QA approaches designed to resolve underspecified queries through interactive questioning [5], [12].

Let $T_{\hat{k}}(\cdot)$ denote the template associated with the ambiguity class $\hat{k}$. The generated clarification question is:

$$Q^c = T_{\hat{k}}(x) \quad (10)$$

The templates are meticulously curated and organized in conjunction with the ambiguity taxonomy present in the system:

- Temporal: "What time period are you referring to?"
- Referential: "What specific person or place are you referencing?"
- Numeric: "Can you specify the exact quantity?"
- Lexical: "Which meaning of this term do you intend?"
- Scope: "What exactly is the range or limit?"
- Pragmatic: "Please provide me with additional background regarding the inquiry?"

3.6 Interactive Response Control

This system's interaction policy is based on a Detect-then-Clarify method by which the system will detect whether or not an ambiguous query has been made before responding. Let C(x) represent the ambiguity confidence score as defined in Section 3.3, and let τ denote the decision threshold. The final system response y is determined by (11):

$$y = \{ Q^c, \ C(x) \geq \tau \ ; \ A(x), \ C(x) < \tau \quad (11)$$

where Q^c is the clarification question generated in Section 3.5, and A(x) denotes the output of the answer generation for unambiguous queries. The purpose of this policy is to prevent the model from giving incorrect answers and to ensure there is a clear process for resolving incomplete information before downstream reasoning occurs.

3.7 Conversational Agent Implementation

The Detect–then–Clarify framework is implemented as a lightweight conversational agent using a Python-based pipeline. Each query passes through three phases: (1) preprocessing, (2) BERT-based ambiguity detection, and (3) classification and generation of clarification. A clarification question is produced if ambiguity is found, and the user response is integrated so that the query can be re-evaluated repeatedly until the ambiguity confidence level falls below τ. Once this threshold is met, the resolved query is sent to the answer generation service. Intermediate outputs are logged to support evaluation and process analysis.

IV. RESULTS AND DISCUSSION

The hybrid classification model for ambiguous input detection was evaluated using Precision, Recall, F1-score, and Overall Accuracy [10]. The model achieved a Precision of 0.696, Recall of 0.599, F1-score of 0.644, and Accuracy of 0.566. These moderate values reflect the strong class imbalance in the dataset, where the majority of input samples are ambiguous, which can be seen in Fig. 2. However, the precision-recall trade-off demonstrates that the algorithm can discriminate ambiguous inputs, even in cases that are very difficult to ascertain. The confusion matrix from Fig. 3 confirms this finding, indicating a high proportion of true positives compared to false positives for ambiguous inputs.

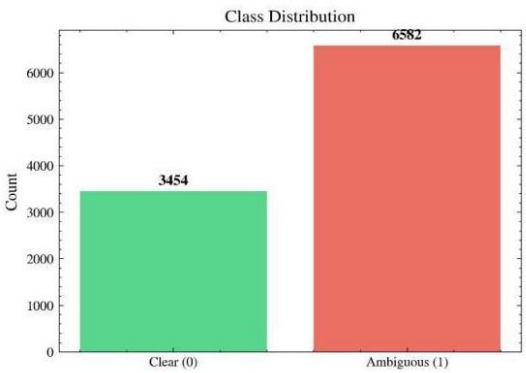

Fig. 2. The distribution of ambiguous and clear queries in the AmbigQA dataset shows a significant class imbalance in the task of detecting an ambiguity.

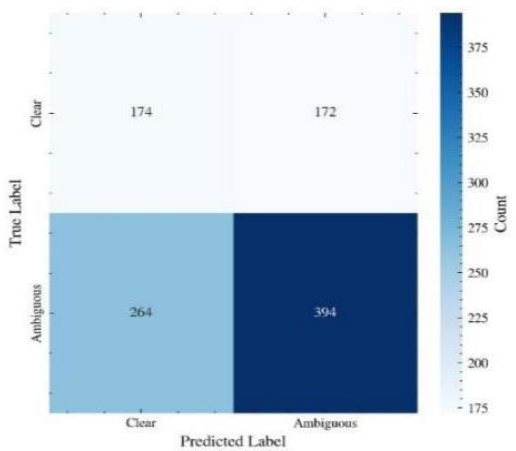

Fig. 3. Confusion matrix illustrating ambiguity classification performance. It shows both the number of correct and incorrect predictions.

Training dynamics show stable convergence across all epochs, with constant decreases in training loss but only slight increases in validation loss being observed, suggesting a case of mild overfitting as opposed to unstable learning. As illustrated in Fig. 5, validation accuracy varies between 0.54 and 0.61 and stabilizes near 0.60. The final test accuracy of 0.566 is slightly lower than validation accuracy, indicating a mild generalization gap but stable model performance on unseen data.

ISBN: 978-81-985770-0-9

Error analysis reveals many misclassified samples fall into a generalized "Other" classification, rather than clearly factual or entity-specific queries, reinforcing the notion that linguistic ambiguity is a subjective and context-dependent construct.

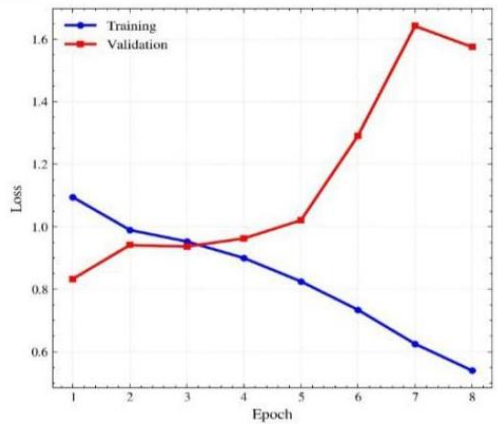

Fig. 4. Training and validation losses throughout various epochs.

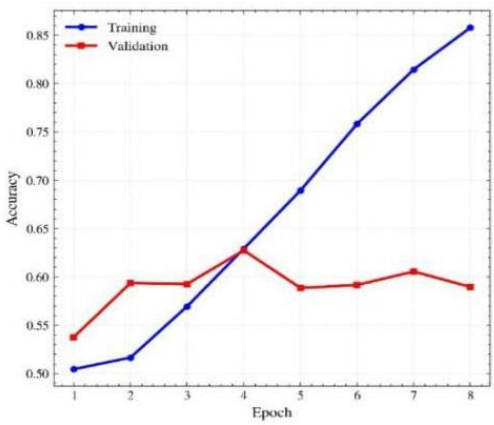

Fig. 5. Training and validation accuracy over the various epochs.

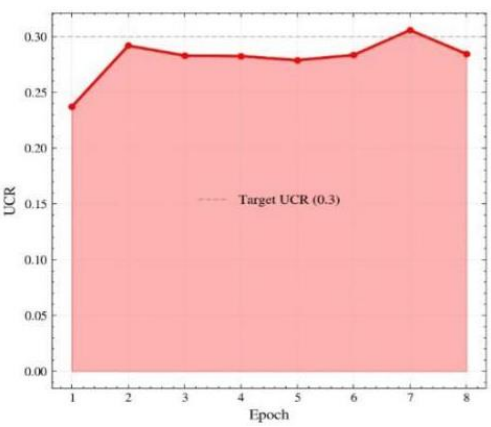

Fig. 6. Progression of the Unnecessary Clarification Rate (UCR) on the validation set across training epochs, with the dashed line representing the intended operational threshold.

The Unnecessary Clarification Rate (UCR) of 0.304, which stays slightly below the operational threshold, signifies that the model mostly avoids unnecessary subsequent questions and conservatively resolves any ambiguities. The Ambiguity Resolution Efficiency (ARE) of 0.793 shows that most ambiguous inputs are resolved by two interactions on average, with an average of 1.62 clarification turns required for each query. The Clarity Improvement Score (CIS) of 0.389 reflects overall clarity improvement, while the mean confidence gain of 0.264 indicates quantitative improvement in ambiguity confidence. All of these interactions show that interactive clarification increases confidence and conversational efficiency.

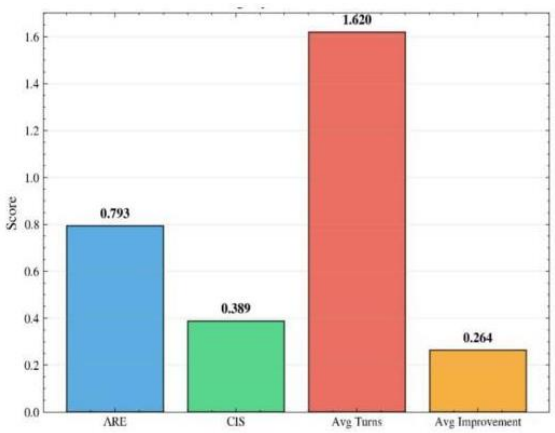

Fig. 7. Evaluation of ambiguity resolution effectiveness through Ambiguity Resolution Efficiency (ARE), Clarity Improvement Score (CIS), average number of clarification turns, and mean confidence improvement.

Comparative evaluation of baseline and optimized performance (Fig. 8) shows significant improvements: Precision increased from baseline to 0.696 (48.1%), ARE improved to 0.793 (98.3%), CIS increased substantially, and UCR decreased to 0.304 (42.7%). These improvements demonstrate the effectiveness of combining contextual embeddings with rule-based clarification in the Detect-then-Clarify framework.

A quantitative user study with 45 participants was conducted to evaluate real-world effectiveness, and 73.3% had a history of regular interaction with AI chatbots. Out of those that participated, more than 95% had interacted with AI chatbots at least once, and thus, they were able to provide informed feedback about their experience. The variety of academic backgrounds represented by the participants supported the generalization of results. 75.6% of participants indicated that the system provided clarification questions about queries that were unclear. The mean Likert score for the relevance of the clarification was 3.6 [11], with 60% rated as highly relevant, using a rating of either 4 or 5. The mean effectiveness rating in assisting participants to determine clarification of intent was 3.82, with 77.7% rated as highly effective, and no reports of complete dissatisfaction were received. Overall, the findings strongly indicate that the Detect–then–Clarify model produces relevant clarifications and greatly increases users' ability to interpret their true intent when interacting with real-world situations.

ISBN: 978-81-985770-0-9

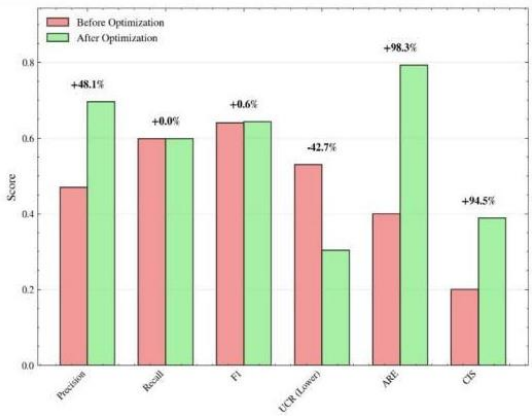

Fig. 8. A comparative analysis of key performance metrics before and after optimization reveals enhancements in the accuracy of ambiguity detection, the efficiency of clarification processes, and the overall quality of resolution.

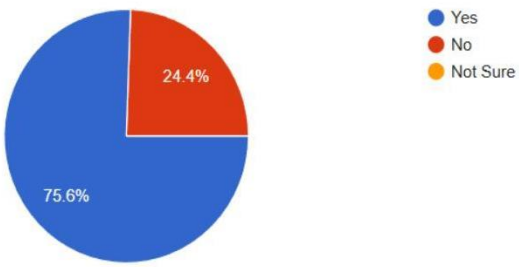

Fig. 9. User study results showing participant responses on whether the Detect–Then–Clarify chatbot asked clarification questions for unclear queries, with 75.6% confirming correct clarification behavior.

V. CONCLUSION

This work displays a hybrid Detect-then-Clarify framework which resolves ambiguous user queries while preserving the interaction efficiency in conversational AI. The system uses confidence-based scoring to learn ambiguity patterns by diverse linguistic cues over query feature and use a fine tuned BERT base uncased model along with the rule guided clarification generation. The model achieves a precision of 0.696 and an F1 score of 0.644 despite the significant class imbalance.

This study provides a meaningful improvement to the real-world dialogue behavior by introducing Ambiguity Resolution Efficiency (ARE), Unnecessary Clarification Rate (UCR), and the Clarity Improvement Score (CIS). These help in minimizing redundant clarification on the one hand and resolving genuine ambiguity on the other hand. The model achieved a UCR score of 0.304, an ARE of 0.793, and a CIS of 0.389, and there is also a mean confidence gain of 0.264. A large number of ambiguities are solved in two turns, giving an average of 1.62 clarification interactions with each query. To balance accuracy and efficiency, the framework employed context-specific follow-up questions and confidence thresholds, which is an extension of the traditional static classification evaluation system. Substantial gains over baseline were achieved, contributing a 98.3% improvement in ARE, a 48.1% increase in precision, and a reduction of nearly 42.7% in UCR, supporting the shift to a clarification-driven interaction from assumption-based responses. Practical effectiveness was confirmed with the validation of 45 participants, where 75.6% report apt clarification that triggers the relevance score of 3.6 out of 5 and 77.7% rating intent understanding improvement on a point scale of 4 or higher. To recommend future deployment, the study suggests several modifications to improve the accuracy and performance of future ambiguity/pragmatic AI systems for real-world use cases: 1) switch from rule-based confidence to learning an estimated level of uncertainty; 2) extend the estimated level of uncertainty to identify multi-label ambiguity for multi-part queries; 3) include dialogue history in the model (for multi-turn); and 4) add personalization to adjust clarification strategies according to user preferences and domain. Collectively, these enhancements will position the framework as a scalable, production-ready solution for ambiguity-aware conversational AI in real-world applications.

REFERENCES

[1] S. Min, J. Michael, H. Hajishirzi, et al., “AmbigQA: Answering Ambiguous Open-Domain Questions,” Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 5783–5797.

[2] J. Devlin, M.-W. Chang, K. Lee, et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Proc. NAACL-HLT, 2019, pp. 4171–4186.

[3] V. K. C. Manam, J. D. Thomas, A. J. Quinn, “TaskLint: Automated Detection of Ambiguities in Task Instructions,” Proc. AAAI Conf. Human Computation and Crowdsourcing (HCOMP), 2022.

[4] J. Kim, H. Seo, J. Park, et al., “Aligning Language Models to Explicitly Handle Ambiguity,” Proc. ACL, 2023.

[5] D. Lee, J. Kim, S. Min, et al., “Asking Clarification Questions to Handle Ambiguity in Open-Domain QA,” arXiv preprint arXiv:2305.13808, 2023.

[6] A. Rao, S. Gupta, R. Patel, et al., “Ambiguity Detection and Uncertainty Calibration for QA with LLMs,” Proc. EMNLP TrustNLP Workshop, 2023.

[7] T. Kwiatkowski, J. Palomaki, M. Redfield, et al., “Natural Questions: A Benchmark for Question Answering Research,” Transactions of the Association for Computational Linguistics, vol. 7, pp. 453–466, 2019.

[8] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. draft. Stanford University, 2023.

[9] A. Niculescu-Mizil and R. Caruana, “Predicting Good Probabilities with Supervised Learning,” Proc. Int. Conf. Machine Learning (ICML), 2005, pp. 625–632.

[10] M. Hossin and M. N. Sulaiman, “A Review on Evaluation Metrics for Data Classification,” International Journal of Data Mining & Knowledge Management Process, vol. 5, no. 2, pp. 1–11, 2015.

ISBN: 978-81-985770-0-9

[11] R. Likert, "A Technique for the Measurement of Attitudes," Archives of Psychology, vol. 22, no. 140, pp. 1–55, 1932.

[12] S. Vakulenko, N. Tu, S. Longpre, et al., "Question Clarification in Open-Domain Question Answering," Proc. ACM SIGIR, 2021.

Cite this article as:

Shefali Kaushal, Vandana Niranjan and et. al. " Detect-Then-Clarify: A Hybrid Approach for Identifying and Resolving Ambiguous User Queries", Proceedings of 13th international conference on Microelectronics, Circuits and Systems, Micro2026.

Displayed as online on 04st July 2026.

Link: http://actsoft.org/science/micro2026-pro/196-micro2026.pdf

ISBN: 978-81-985770-0-9