Multi-Modal Requirements Data-based Acceptance Criteria Generation using LLMs

Fanyu Wang*, Chetan Arora*, Yonghui Liu* Kaicheng Huang*,
Chakkrit Tantithamthavorn*, Aldeida Aleti* Dishan Sambathkumar†, David Lo‡
*Faculty of Information Technology, Monash University, Clayton, VIC, Australia
{fanyu.wang, chetan.arora, yonghui.liu, khua0042, chakkrit, aldeida.aleti}@monash.edu
†eSolutions, Monash University, Clayton, VIC, Australia
dishan.sambathkumar@monash.edu
‡School of Computing and Information Systems, Singapore Management University, Singapore
davidlo@smu.edu.sg

***Abstract*—Acceptance criteria (ACs) play a critical role in software development by clearly defining the conditions under which a software feature satisfies stakeholder expectations. However, manually creating accurate, comprehensive, and unambiguous acceptance criteria is challenging, particularly in user interface-intensive applications, due to the reliance on domain-specific knowledge and visual context that is not always captured by textual requirements alone. To address these challenges, we propose *RAGcceptance_M2RE*, a novel approach that leverages Retrieval-Augmented Generation (RAG) to generate acceptance criteria from multi-modal requirements data, including both textual documentation and visual UI information. We systematically evaluated our approach in an industrial case study involving an education-focused software system used by approximately 100,000 users. The results indicate that integrating multi-modal information significantly enhances the relevance, correctness, and comprehensibility of the generated ACs. Moreover, practitioner evaluations confirm that our approach effectively reduces manual effort, captures nuanced stakeholder intent, and provides valuable criteria that domain experts may overlook, demonstrating practical utility and significant potential for industry adoption. This research underscores the potential of multi-modal RAG techniques in streamlining software validation processes and improving development efficiency. We also make our implementation and a dataset available.**

***Index Terms*—Requirements-Driven Testing, Multi-Modal Large Language Models, Retrieval-Augmented Generation, Requirements Engineering, Acceptance Criteria, Industry Study**

I. Introduction

Acceptance testing is a crucial validation activity within the software development lifecycle, ensuring that a software system not only meets functional requirements but also aligns with user expectations [1], [2]. Distinguished from other testing approaches, acceptance testing focuses explicitly on compliance with the requirements specification [3], establishing a strong connection with the requirements engineering (RE) stage. At the core of *acceptance testing* are acceptance criteria, which serve as the principal test specifications for acceptance testing. Acceptance criteria (ACs)[1] are defined as "the conditions that a system must meet to fulfil a user story and ultimately be accepted by the user" [4]. This definition highlights two key requirements: (1) acceptance criteria must be well-aligned with the requirements, and (2) they must effectively capture the user's intent. Distinguishing from other test artifacts (e.g., unit or integration tests), which typically verify concrete implementation in primary [5], ACs are not only designed to reflect high-level user or business objectives, domain-specific nuances clearly, but also often abstract concepts conveying user interactions and system behaviors.

In professional software development environments, ACs are predominantly documented in structural natural language (NL). The Gherkin format has been well adopted as the typical format of ACs, following the "GIVEN", "WHEN", and "THEN" pattern [6], [7]. The manual specification of ACs requires a comprehensive understanding of both the requirements and the context, as well as a significant time investment to ensure accuracy and clarity. This process becomes even more challenging due to ambiguities, inadequate details in user stories, and the evolving nature of software specifications.

Automating quality assurance artefact generation, like ACs, is an important task in the industry [8]. However, the inherent challenges in manual creation persist in automated approaches, necessitating solutions that ensure accurate intent understanding and requirements alignment. Large Language Models (LLMs) have demonstrated remarkable capabilities in natural language processing (NLP), and automating software engineering (SE) tasks [9], [10]. Despite their advanced capabilities, concerns remain regarding their ability to acquire domain-specific knowledge and fully grasp nuanced user intent [11]–[13]. In the context of ACs, an unguided LLM might inadvertently 'hallucinate' and invent conditions not specified by stakeholders or omit essential constraints that were specified. For example, given a vaguely worded requirement, an LLM might fill in gaps with its own assumptions, producing criteria that appear reasonable but do not actually reflect the true intent of the requirement. This undermines trust in automation: stakeholders and developers cannot rely on generated criteria if there is a risk of false conditions leading the project astray. Ensuring that every generated AC is faithful to the source requirements (and nothing more) is therefore paramount when introducing automation.

[1]We abbreviate multiple criteria as ACs and a single criterion as AC.

To address these challenges, we propose *RAGcceptance_M2RE*, which stands for Retrieval Augmented Generation based acceptance criteria generation from Multi-Modal REquirements information (reads RAGcceptance motor). Our approach combines the generative power of LLMs with a retrieval mechanism to automate ACs generation in a more reliable way. By integrating Retrieval-Augmented Generation (RAG) [14] with multi-modal requirements representations, this method grounds the LLM's output in authentic project data. Rather than relying solely on the LLM, *RAGcceptance_M2RE* automatically retrieves pertinent context from existing domain artifacts such as project background information (the interaction logics, stakeholders' description, or system objectives) and system UI, providing context to the LLM as additional input. This additional context ensures that each generated criterion is based on authentic project data rather than assumptions. In our paper, we leverage multi-modal RAG, i.e., in addition to using RAG for textual artifacts (which have been previously used in RE research [15], [16]), we leverage visual RAG models for incorporating additional information captured from the system's UI. We posit that integrating visual elements with the textual domain information enhances the accuracy and completeness of the generated ACs, resulting in outputs closely aligned with stakeholder expectations.

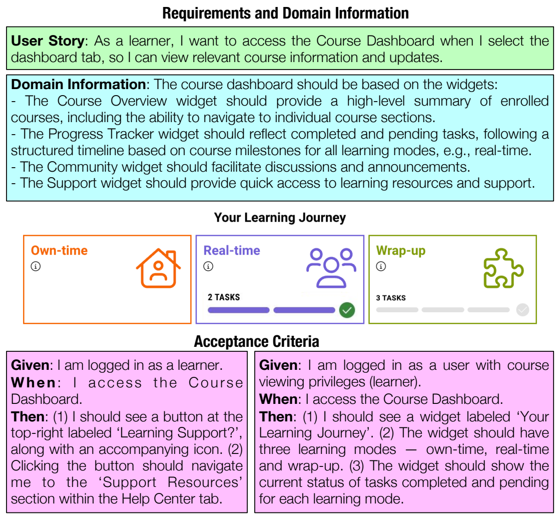
A. *Industry Context*

We evaluate *RAGcceptance_M2RE* in an industrial study conducted with *eSolutions*, which is the central IT partner for Monash, and as an individual entity comprises over 500 ongoing staff and an additional 200 contractors, casuals, and affiliates. It delivers end-to-end IT services across the university—from installing software on student devices to developing and maintaining the world's largest online exam platform and the infrastructure that underpins it. Their projects encompass course administration, IT service provision, building software products, and meeting all infrastructure, software, and cybersecurity needs. We conducted our evaluation on *UniLearn* product data (some details are anonymized due to proprietary nature), which is a representative product in their series of customised products. *UniLearn* is a university learning management system with ∼100,000 current users.

B. *Working Example and Motivation*

Fig. 1 shows an example user story, multi-modal domain information, and ACs[2]. This example has been slightly adapted from the original dataset from *UniLearn*. The user story in the example is related to the customisation for learner groups. The ACs in the example are also closely related to the original ACs specified by analysts. Having said that, the figure shows ACs that are composite, i.e., each criterion contains multiple sub-criteria. However, our approach generates atomic criteria, based on the best practices [17] and recommendations by the *eSolutions* quality assurance (QA) team.

[2]This example is taken *verbatim*—with all their minor formatting inconsistencies and non-standard labels to show the truly "noisy" industry inputs.


Requirements and Domain Information

User Story: As a learner, I want to access the Course Dashboard when I select the dashboard tab, so I can view relevant course information and updates.

Domain Information: The course dashboard should be based on the widgets:
- The Course Overview widget should provide a high-level summary of enrolled courses, including the ability to navigate to individual course sections.
- The Progress Tracker widget should reflect completed and pending tasks, following a structured timeline based on course milestones for all learning modes, e.g., real-time.
- The Community widget should facilitate discussions and announcements.
- The Support widget should provide quick access to learning resources and support.

Acceptance Criteria

Given: I am logged in as a learner.
When: I access the Course Dashboard.
Then: (1) I should see a button at the top-right labeled 'Learning Support?', along with an accompanying icon. (2) Clicking the button should navigate me to the 'Support Resources' section within the Help Center tab.

Given: I am logged in as a user with course viewing privileges (learner).
When: I access the Course Dashboard.
Then: (1) I should see a widget labeled 'Your Learning Journey'. (2) The widget should have three learning modes — own-time, real-time and wrap-up. (3) The widget should show the current status of tasks completed and pending for each learning mode.

Fig. 1. Example User Stories, Domain Information, System UI, and Acceptance Criteria from *UniLearn* project at *eSolutions* organisation.

The user story in the example only mentions "relevant course information and updates" – which is insufficient for generating accurate ACs. Analysts usually rely on their domain knowledge to write complete and accurate ACs, which is not captured in user stories. The first specified AC mentions "learning support" which is captured from the textual domain information from the project background document, which usually has much more information and hence contributes to writing ACs. The second AC relies on both the textual and visual information (see a real UI screenshot from *UniLearn*) of learning modes of own-time, real-time and wrap-up, and the corresponding tasks assigned to learners in a given course. As evident in Figure 1, background information is required to supplement the information captured in user stories. Specifying ACs manually requires deep domain expertise, which is often limited to a subset of project analysts. An approach like *RAGcceptance_M2RE* is targeted at supporting analysts who might be new to projects or save effort on specifying ACs even for experienced analysts.

C. *Contributions*

This paper introduces *RAGcceptance_M2RE*, an innovative approach for automating the generation of ACs from multi-modal requirements information. We release our implementation and testset [18].[3] Our key contributions are:

• **Multi-Modal Retrieval-Augmented Generation (RAG):** We propose an integrated framework that combines textual and visual RAG techniques. This ensures comprehensive and context-aware AC generation by retrieving and utilizing relevant textual and visual project artefacts. To the best of our knowledge, our study is among the first to explore and validate multi-modal RAG for requirements-driven automated quality assurance in software engineering, establishing a robust foundation for future research and practical applications.

[3]The testset is a small anonymized subset from *UniLearn*. We remove any confidential/proprietary information from the subset.

• **Reward-based Refinement Process**: We incorporate two distinct reward models, commonly used in the AI domain, to iteratively refine generated ACs, significantly improving their correctness, clarity, and alignment with user stories.

• **Empirical Evaluation in an Industrial Context**: Our method is rigorously evaluated in an industrial context involving data from *UniLearn* product from *eSolutions*. Industrial evaluations are rather scarce in the RE research domain, and our evaluation demonstrates substantial performance improvements and validates practical applicability.

• **Expert Validation**: We gathered detailed feedback from experienced practitioners, providing practical insights into the quality, utility, and effectiveness of our generated ACs. Experts confirmed the significant potential for time savings and highlighted instances where our method captured essential criteria not explicitly stated in user stories.

Paper Structure. Section II provides background on relevant concepts and related work. Section III outlines our technical approach. Section IV details our industry evaluation. Section V discusses threats to validity, and Section VI concludes.

II. Background

This section provides two key concepts: retrieval augmented generation (RAG) and reward models used in our approach. We further position our work against the related work.

A. Retrieval-Augmented Generation (RAG)

RAG is a recent significant advancement in the NLP domain. It is the process of integrating the external knowledge sources in the generation process with the help of pre-trained LLMs [14], [19], [20]. Three key stages of a RAG pipeline are indexing, retrieval, and generation [21]. Given a query, the RAG module first searches through the *indexed* structure of external knowledge to *retrieve* the most relevant target. The retrieved knowledge is then used in the *generation* stage, where it is fed along with a prompt to an LLM to perform the generation task, e.g., generating ACs in our case. RAG models, compared to LLMs for generation alone, help alleviate hallucination issues, particularly when project-specific context is unavailable as pre-training for LLMs [15], [19]. RAG models can have multi-modalities [22], [23]. In our work, we use textual and visual RAG models.

Textual RAG (T-RAG) is the most common scenario in practice, serving as the foundational approach for RAG models. Traditional information retrieval techniques, such as TF-IDF [24] and BM25 [25], are often utilized, particularly in recommendation systems, to identify relevant documents from large textual datasets efficiently. Classic T-RAG models typically involve indexing documents through embeddings, such as SentBERT [26], which transforms textual data into dense vector representations that capture semantic meaning. These embeddings enable effective retrieval through semantic similarity metrics. For example, SentBERT embeddings are commonly used due to their effectiveness in capturing sentence-level semantics, making them highly suitable for T-RAG. Another variation is *In-context textual RAG models*, which further refine the retrieval-generation interaction by explicitly embedding retrieved textual examples directly into the prompt provided to the LM. By including retrieved-context as explicit examples in the prompt, the model can generate outputs that closely match the exemplified style, structure, and content. In our work, we experiment with two different T-RAG alternatives – SentBERT (as discussed above) and ICRALM (In-Context Retrieval-Augmented Language Models) [27].

Visual RAG (V-RAG) extends the traditional T-RAG approaches by incorporating visual information into the retrieval-generation pipeline. V-RAG models index and retrieve relevant visual artefacts, such as images, screenshots, or interface mock-ups, leveraging visual embedding techniques from advanced vision models, e.g., MuRAG [28]. Given a textual query, V-RAG models retrieve visually similar artefacts by matching textual embeddings with visual embeddings in a shared semantic space. Such visually informed generation is especially valuable in tasks requiring visual comprehension, such as generating UI-related acceptance criteria. V-RAG models are also based on multi-modal LLMs, e.g., RA-CM3 [29], which integrates a base multi-modal LLM in the framework to generate mixtures of text and images.

B. Reward Model

A reward model helps evaluate how good or appropriate an output is from an AI model. Reward models originally gained prominence with systems like AlphaGo [30], where a reward was used to evaluate the outcomes of various actions. In the context of LLMs, a reward model is a specialized mechanism designed to assess the quality and appropriateness of generated outputs [31]. These models use a classifier architecture that assigns scores or labels to outputs, reflecting their relevance, accuracy, or preference alignment [32]. Common approaches include probability-based reward functions [33], [34], explicit class-based scoring with clearly interpretable scales (e.g., ratings 0–4) [35]–[37], and using LLMs themselves as judges, comparing candidate outputs directly to select the most appropriate one based on provided context [38], [39]. Such clearly defined scoring systems make the reward model outputs interpretable, enabling practical assessment and ongoing model improvement. We use different reward models in our approach to iteratively improve the quality of the generated ACs. In our work, we have two levels of the rewarding process, where Prometheus [36] serves as the higher level of reward model for overall scoring for the generated ACs based on five predefined quality levels: generative verifier [40] and UR3 [41] are two baselines for the lower level of reward process. The details of our reward processes are introduced in Section III-C.

C. Related Work

Requirements-driven automated quality assurance artifact generation remains underexplored in SE [42]. A recent survey by Wang et al. [8] identifies 156 related studies published between 1990 and 2024. Notably, the survey highlights that NL requirements, due to their inherent ambiguity and incompleteness [43], [44], receive relatively little attention, accounting for only 30% of all studies [45]–[47]. Acceptance criteria play a crucial role in acceptance testing by providing a clear and objective framework to evaluate whether a product or system meets the specified requirements and standards [48]–[50]. Compared to other test artifacts, ACs exhibit a more structured format, typically following the “GIVEN”, “WHEN” and “THEN” pattern [6], [7]. This structure demands a deeper understanding of stakeholder expectations, more concrete test implementation, and greater clarity [4]. The officially reported studies on this topic are very few, where Wang et al. [8] identify only one study that explicitly addresses the generation of acceptance criteria [51]. Aside from acceptance criteria, there are few studies on the generation of automated acceptance testing. Fischbach et al. [1] proposed CiRA, a conditional statements-based approach using NLP tools for acceptance test case creation. Wang et al. [52] introduced an automated acceptance testing generation approach based on use case specifications, utilizing a traditional NLP pipeline. Straszak et al. [53] proposed an automated acceptance testing method with some RSL (Requirements Specification Language) parsing tool. AGUTER is a platform to design UATs (User Acceptance Testing) from scenarios using the task/method model.

Our framework introduces advanced NLP techniques, specifically multi-modal retrieval-based generation, to enable diverse domain documents that support the generation of acceptance criteria. The framework design addresses the ambiguity of NL requirements by utilizing advanced LLMs with reward processes and implements concrete acceptance criteria using a multi-modal RAG. However, to the best of our knowledge, it is the first work to enable multi-modal, requirements-driven automated software testing.

III. Approach

Fig. 2 shows an overview of *RAGcceptance_M2RE* for generating ACs from NL requirements, specified as user stories. The inputs to our approach are user stories, with (optional) additional background information in the form of textual domain documents and UI in the form of screenshots. The output of our approach are acceptance criteria generated for the user story. Our approach consists of three stages: (1) Domain Information Retrieval, (2) Acceptance Criteria Generation, and (3) Reward-based Post-Processing.

A. Domain Information Retrieval

Given a user story, the first step in our approach is to retrieve relevant contextual domain information. User stories provide high-level requirements information, often rendering themselves insufficient for generating detailed and concrete acceptance criteria. To address this limitation, the information retrieval stage is introduced to leverage user stories as input to query relevant domain knowledge. Our approach supports textual information retrieval and enhances the retrieval process with image-based knowledge sources by integrating both textual and visual RAG methods, greatly enhancing the contextual information and the user story.

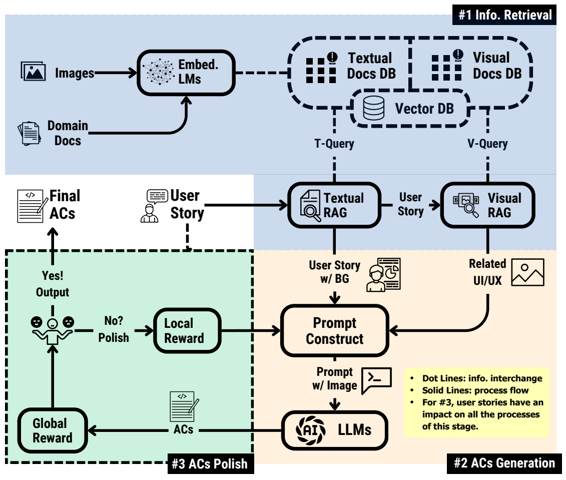
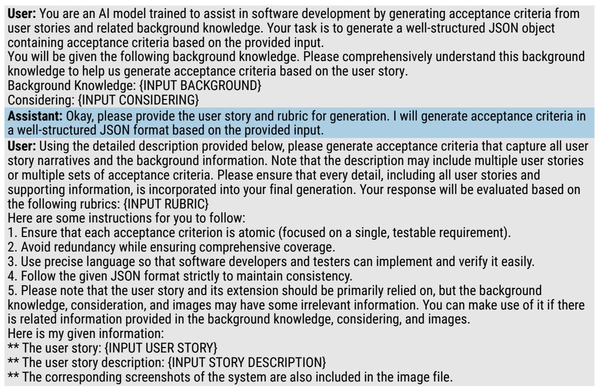


Fig. 2. Overview of *RAGcceptance_M2RE*

Textual Information Retrieval. Given a user story in textual format, the textual RAG (T-RAG) model predicts a relevance score for each domain document based on its relatedness to the user story. After ranking the passages, the top k-ranked ones are selected for retrieval. One can use several measures for determining the similarity between the query (in our case, the user story) and the underlying domain documents for RAG models. We experiment with two alternative techniques, namely *SentBERT* [26] with cosine similarity calculation (we name it as SentBERT in the following) and *ICRALM* [27], covered in Section IV. To maintain the integrity of the description of the domain knowledge, we use paragraph-level chunk size in the retrieval process, where each text chunk describes one function or feature (similar to the example in Fig. 1). Our approach is flexible for the other similarity metrics in RAG.

Visual Information Retrieval. As discussed in Section I, the system’s UI details (if available) can be valuable for generating more accurate ACs (also evaluated in Section IV-E). To retrieve the most closely related visual information (captured as UI screenshots in our case), we use the user story as the input for visual RAG (V-RAG) models. The primary distinction between T- and V-RAG lies in the processing method: instead of textual features, visual information retrieval involves extracting visual features from images to assess their relevance to the query text, which requires a more comprehensive process that incorporates both textual and visual modalities.

V-RAG offers several possibilities, of which we have implemented three. In the first alternative (query#1), all the visual content is converted into HTML using an image-to-HTML converter, and the best-matching HTML pages are retrieved. The second alternative (query#2) extends query#1, wherein the redundant information in HTML is pruned to retain only the key information, e.g., pruning CSS details. This reduces noise and enhances the specificity and relevance of the retrieved data. In the third alternative (query#3), visual embeddings from images are directly indexed without HTML conversion. The textual query is matched against the visual embeddings, allowing direct retrieval of visually similar artefacts.



User: You are an AI model trained to assist in software development by generating acceptance criteria from user stories and related background knowledge. Your task is to generate a well-structured JSON object containing acceptance criteria based on the provided input.
You will be given the following background knowledge. Please comprehensively understand this background knowledge to help us generate acceptance criteria based on the user story.
Background Knowledge: {INPUT BACKGROUND}
Considering: {INPUT CONSIDERING}

Assistant: Okay, please provide the user story and rubric for generation. I will generate acceptance criteria in a well-structured JSON format based on the provided input.

User: Using the detailed description provided below, please generate acceptance criteria that capture all user story narratives and the background information. Note that the description may include multiple user stories or multiple sets of acceptance criteria. Please ensure that every detail, including all user stories and supporting information, is incorporated into your final generation. Your response will be evaluated based on the following rubrics: {INPUT RUBRIC}
Here are some instructions for you to follow:
1. Ensure that each acceptance criterion is atomic (focused on a single, testable requirement).
2. Avoid redundancy while ensuring comprehensive coverage.
3. Use precise language so that software developers and testers can implement and verify it easily.
4. Follow the given JSON format strictly to maintain consistency.
5. Please note that the user story and its extension should be primarily relied on, but the background knowledge, consideration, and images may have some irrelevant information. You can make use of it if there is related information provided in the background knowledge, considering, and images.
Here is my given information:
** The user story: {INPUT USER STORY}
** The user story description: {INPUT STORY DESCRIPTION}
** The corresponding screenshots of the system are also included in the image file.

Fig. 3. APEER Prompt Template

B. Acceptance Criteria Generation

Once the multi-modal background information is available from the previous step, the next step is to generate ACs originating from the user story. Here, we have two sub-steps, namely prompt construction and generation. Prompt engineering can make a significant difference in the generated outcomes [54]–[56]. Chain of Thought (CoT) enables the logical thinking ability of LLMs by providing specific execution steps. We experiment with two different CoT prompt construction methods – (1) *Urial* [57], which focuses on prompt construction with in-context learning by restyling the given information into structured instruction with in-context alignment, and (2) *APEER* [58] which focuses on the ground knowledge in prompt construction but without any examples. The example prompt template of APEER is detailed in Fig. 3. All other prompt templates are also made available [18]. In our framework, we have two types of input, including textual and visual information. As for the multi-modal LLMs, we adopted the image parsing method by directly reading the base64 encoded information of the image.

C. Reward-based Post Processing

While LLMs are employed as generators for AC, their opaque processing process [59], [60] hinders the guarantee of consistently high-quality outputs. Yet RAG enhances the generation process by providing additional knowledge, introducing noise will also lead to irrelevant or unexpected results, which is named "retrieval noise" or "noise propagation " [61], [62]. To ensure the quality of the generated AC, we incorporate a post-processing/polishing step using reward models, which refines any low-quality AC and ensures alignment with the given user stories. This thereby improves the overall reliability of the generated AC. The notion of quality in AC (based on best practices and guidance from our industry partner) are:

- Avoid vague AC, like "The system should work properly.", as providing minimal guidance to the QA team.
- AC should be concise and include measurable outcomes for testability, unambiguity, and consistency.
- Generated AC should be comprehensive, i.e., cover edge cases, and positive and negative outcomes.
- Follow recommended syntax and atomicity, i.e., "GIVEN, WHEN, THEN" and provide atomic outcomes for "Then" part.

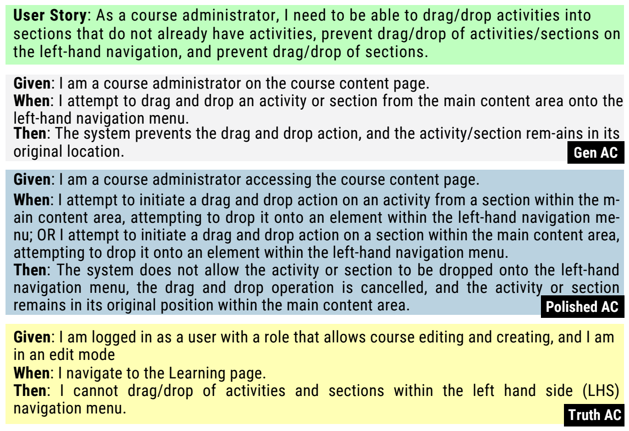
Score-based Polishing. Algorithm 1 outlines the AC polishing process. Our approach leverages reward models that assign scores to the generated candidates based on the user story. Instead of only considering the individual quality of each AC, we introduce an overall quality check for all ACs. We use a global reward model that classifies generated AC into predefined quality levels as the overall assessment. Prometheus [36] is an open-source reward model that supports absolute rewarding. We define five levels (scored 1-5) using a six-dimensional rubric to determine the overall quality of the generated AC. We designed the rubric based on the dimensions in our expert evaluation (see Sec. IV-C), but with a more atomic description, facilitating the understanding of LLMs. Given space constraints here, all material is available in our repository [18].

Algorithm 1 Polishing ACs using Reward Models

```
Require: Set of generated acceptance criteria $\mathcal{AC} = \{ac_1, ac_2, \ldots, ac_n\}$,
    user story $S$, global threshold $T$
Ensure: Refined set of $\mathcal{AC}'$
 1: $score_{\text{global}} \leftarrow \text{GlobalRewardModel}(S, \mathcal{AC})$
 2: if $score_{\text{global}} \geq T$ then
 3:     return $\mathcal{AC}$
 4: else
 5:     for each $ac_i \in \mathcal{AC}$ do
 6:         $score_i \leftarrow \text{LocalRewardModel}(S, ac_i)$
 7:     end for
 8:     $ac_{\text{worst}} \leftarrow \arg\min_{ac_i \in \mathcal{AC}} score_i$
 9:     $ac_{\text{polished}} \leftarrow \text{LLM\_Polish}(ac_{\text{worst}}, \mathcal{AC} \setminus \{ac_{\text{worst}}\}, S)$
10:     Replace $ac_{\text{worst}}$ in $\mathcal{AC}$ with $ac_{\text{polished}}$
11:     return $\mathcal{AC}$
12: end if
```

We also employ a local reward model to assign a numeric score to each AC. The numeric score is more friendly for us to select the low-quality AC among all the candidates. We experiment with two local reward models, namely Generative Verifier [40] and UR3 [41]. Given a predefined prompt with the reward objective, generative verifier [40] will predict the probability of "yes" and "no" tokens based on this prompt, which is similar to the idea of binary classification. The scale of the probability is from 0-1, where a higher probability of "yes" indicates that the prediction aligns better with the stated objectives. UR3 [41], compared with the generative verifier, prefers a comprehensive estimation of language perplexity for all the input tokens, including the prompt. It quantifies the score estimation process by measuring the divergence using the Kullback–Leibler divergence, which refers to the LLM's approximation and the actual objective-specific distribution. *Based on the output of the reward model, if ACs do not meet the quality checks, we re-generate the lowest-scored acceptance criterion unless all generated criteria are above a certain level.* The polishing is not for all generations. We set the threshold for polishing to 5, which means all the ACs with scores less than five will be polished. Ultimately, we have all the acceptance criteria that meet the quality checks in this post-processing step, as the final output of *RAGcceptance_M2RE*.

Example Implementation of RAGcceptance_M2RE. As we exemplify in Fig. 4, given a user story for dragging and



User Story: As a course administrator, I need to be able to drag/drop activities into sections that do not already have activities, prevent drag/drop of activities/sections on the left-hand navigation, and prevent drag/drop of sections.

Given: I am a course administrator on the course content page.
When: I attempt to drag and drop an activity or section from the main content area onto the left-hand navigation menu.
Then: The system prevents the drag and drop action, and the activity/section rem-ains in its original location.
Gen AC

Given: I am a course administrator accessing the course content page.
When: I attempt to initiate a drag and drop action on an activity from a section within the m-ain content area, attempting to drop it onto an element within the left-hand navigation me-nu; OR I attempt to initiate a drag and drop action on a section within the main content area, attempting to drop it onto an element within the left-hand navigation menu.
Then: The system does not allow the activity or section to be dropped onto the left-hand navigation menu, the drag and drop operation is cancelled, and the activity or section remains in its original position within the main content area.
Polished AC

Given: I am logged in as a user with a role that allows course editing and creating, and I am in an edit mode
When: I navigate to the Learning page.
Then: I cannot drag/drop of activities and sections within the left hand side (LHS) navigation menu.
Truth AC

Fig. 4. Examples of Case Study

dropping activities and sections, *RAGcceptance_M2RE* first generates a set of AC (one Gen AC shown in the figure). Then our polishing component assigns it a 3/5 score before producing the Polished AC. The original Gen AC ambiguously bundles activity and section drags into a single clause, merely states that "the system prevents" without clarifying whether the operation is formally cancelled or how the UI should behave, and refers only to the "original location" without specifying the main content area. The Polished AC, in contrast, distinctly separates activity-vs-section scenarios, explicitly cancels the drag-and-drop operation, and reiterates that the item remains in its original position within the main content area. In comparison to the expert-written Truth AC, it adds precision in the GIVEN clause by specifying "accessing..." ensuring tests cannot pass in view-only contexts.

IV. Empirical Evaluation

A. Research Questions

RQ1. Which RAGcceptance_M2RE configuration leads to the most accurate acceptance criteria generation results? As presented in Section III, different parts of *RAGcceptance_M2RE* can be instantiated using various alternatives. RQ1 aims to identify the V-RAG and T-RAG alternative techniques and the prompting strategies that yield the most accurate ACs generation results. We further perform an ablation study to assess the advantages (if any) of providing background information for ACs generation.

RQ2. Which combination of reward models and LLMs yields the best alignment between generated acceptance criteria and the underlying user story? Reward models are applied to improve the generated AC's structure and alignment with the underlying user story. RQ2 analysis demonstrates how the reward model improves alignment. We also determine the best LLM alternative for generating ACs.

RQ3. Do practitioners find the generated acceptance criteria useful? In RQ3, we report on an interview survey with practitioners to gain insights into the usefulness of the generated AC from their perspective, based on the best alternatives from RQ1 and RQ2 and previously unseen data.

B. Implementation Details

Platform. We conducted our experiments on an NVIDIA A100 GPU using full-parameter within the PyTorch framework. All models were initialized in bfloat16 and 4-bit mode.

Data Preprocessing. Our dataset consists of domain documents in paragraph or sentence style, where we restructure domain documents into JSON files.

Information Retrieval. For ICRALM, we use Llama 3B [63] with $\text{HTML}_{\text{Prune}}$ and $\text{HTML}_{\text{Full}}$. We use an open-source IMG2HTML tool [64] and a pruning tool in Tan et al. [65]. Additionally, we employ DSE and SentBERT from Hugging Face [66], [67]. The default prompts for these methods are used, based on their respective repositories.

Prompts and LLMs Configuration. We adopt Urial and APEER prompt templates, as detailed in their appendices. For image parsing, we directly process images using base64 encoding. We utilize the latest versions of each model available before March 2025, with default hyperparameter settings, including temperature and top_p, for initialization, but maintain the maximum input length to ensure flexibility in processing.

Reward Models and Polishing. Prometheus [36] provides fine-tuned checkpoints on Hugging Face. The Generative Verifier is adaptable to any open-source LLM. We initialize them using Llama 3.1 8B [68]. For polishing, we maintain the same LLM settings as in the previous step. The polishing prompt is appended to the AC generation prompt, creating a multi-cycle dialogue to maintain contextual understanding.

C. Data Collection Procedure.

The data collection procedure in our industry study consists of two stages: collecting industry data and gathering expert feedback. In the first stage, we gathered user stories, ACs, and the relevant multi-modal documents. In the second stage, we return to experts from *eSolutions* to obtain their evaluation of ACs generated by *RAGcceptance_M2RE*.

Industry Data Collection. We collected the data from Jira [69] using official REST API, where each case of the data consists of *User Story and Extension*, *Background Description*, *Consideration* (system requirements or non-functional requirements), *Screenshots*, and *Acceptance Criteria*. The user story contains the necessary information as input. The background description, consideration, and screenshots are the background knowledge, which are structured into textual and visual databases for querying (see Figure 2). Table I provides a brief description of the two samples, referred to as Sample A and Sample B. Sample A has been used in our approach to select different RAG, LLM, and reward model alternatives in RQs 1 and 2. Sample B data has been used to evaluate the best combination of *RAGcceptance_M2RE* for expert evaluation in RQ3. We intentionally collected expert feedback on previously unseen data (Sample B) to prevent evaluation bias, since the determined combination on Sample A should not be used with the same data for expert evaluation. Therefore, evaluating on separate data ensures an objective assessment of the effectiveness.

TABLE I
DATASET DESCRIPTION

Sample	User Stories	ACs	Atomic ACs	Description
A	42	169	224	*UniLearn* course structuring for learners.
B	32	109	179	*UniLearn* overall system management for all users.

Expert Feedback Collection. In this stage, we collected data from three experts from *eSolutions*, in an interview survey to assess their perception of the quality of the generated ACs in RQ3. The three experts include a senior business analyst (Expert 1), a functional integration business analyst (Expert 2), and a senior quality assurance and solutions architect (Expert 3). Expert 1 has three years of experience in *UniLearn* and 10 years of total industry experience, Expert 2 has 2.5 years of *UniLearn* experience and 7 years in the software industry, and Expert 3 has worked at *eSolutions* for 4 years and 15 years in total in the SE industry. We only recruit the experts from *eSolutions* rather than widely finding the experts from open domains due to their professional knowledge and understanding about *UniLearn*.

The interview survey session lasted ≈2 hours. At the beginning of the session, we explained the evaluation procedure to the experts and provided a quick recollection of the project requirements. Given the limited availability of experts, we were able to cover detailed feedback on 17 user stories and 81 corresponding ACs generated using *RAGcceptance_M2RE* for Sample B, in a random order. The experts reviewed each AC at a time and had access to the corresponding user story. For each AC and a user story, we asked experts to rate them on three quality aspects (noted below). For all ACs generated for a given user story, they also rated their *coverage*. The experts rated each of these four aspects on a five-point Likert scale [70]. Rated 1 (Strongly Disagree) means the AC significantly fails to meet the quality aspect. Rated 2 (Disagree) means that the AC does not adequately meet the quality aspect. Rated 3 (Neutral) means that the AC somewhat meets the quality aspect, but improvements are necessary. Rated 4 (Agree) means that the AC meets the quality aspect satisfactorily, with only minor issues or omissions. Rated 5 (Strongly Agree) means that the AC excellently meets the quality aspect. The four quality aspects, relevance, correctness and understandability for each AC and coverage for all ACs of a given user story are described below:

• *Relevance* assesses how well the generated AC aligns with the given user story.

• *Correctness* evaluates the accuracy of AC, format and the information captured in the AC. In short, the AC should be factually accurate and logically structured.

• *Understandability* assesses whether the generated AC is clear and comprehensible to the experts without any redundancies or inconsistent terminology introduced by an LLM.

• *Coverage* evaluates the 'completeness' of the generated ACs, i.e., which ACs encompass all relevant aspects of the user story, e.g., positive & negative flows and edge cases.

We asked experts to share their rationale for a given rating on each AC. The experts were free to agree or disagree with, or even change their rating, in case another expert had a different rationale that they might have missed. Ultimately, a consensus rating was achieved for each AC. We noted down the ACs where they did not agree on a rating. We also noted down each expert's brief sharing from their overall perspective on the results at the end of the session.

TABLE II
CHOICES TABLE

Independent Variable	Num_ Choices	Choices
T-RAG	2	*SentBERT* [26] and *ICRALM* [27]
V-RAG	3	$HTML_{\mathrm{Prune}}$, $HTML_{\mathrm{Full}}$ and *Document Screenshot Embedding (DSE)* [66]
Retr. Top-k	4	1, 5, 10 and 20
Prompt Construction	2	*Urial* [57] and *APEER* [58]
LLMs	3	*Claude* [78], *Gemini* [79] and *GPT4o* [80]
Local Reward Model	2	*Generative Verifier* [40] and *UR3* [41]
Background Information	3	With VRAG and TRAG, No VRAG or TRAG (w/o RAG), and No VRAG only TRAG (w/o VRAG)

D. Evaluation Procedure and Metrics

There are eight main independent considerations in our approach, namely, (i) T-RAG model, (ii) V-RAG model, (iii) top-k value for T-RAG and V-RAG (we set k values by referring the existed studies with similar chunk size [14], [71]), including the number of visual documents retrieved by V-RAG and the number of textual passages retrieved by T-RAG, (v) prompt construction techniques, (vi) selected LLMs, (vii) local reward models (note that we maintain global reward model as an integral part of our approach); and (viii) choice of background information. Table II shows all the choices for these. While an exhaustive evaluation of all possible combinations would be ideal, practical constraints necessitate a selective approach. Therefore, the alternative models chosen for each case are motivated either by their widespread adoption in NLP literature [72]–[74] or by the availability of implementations. Given the ground truth from *UniLearn* data, we conducted a stepwise evaluation in RQ1 and RQ2 to determine the best configuration for *RAGcceptance_M2RE* in three steps, including 1) information retrieval, 2) AC generation, and 3) AC polishing. while the generation ability is strongly determined by the generative methods, such as LLMs, Pretrained LMs, etc, they separately evaluate the information retrieval process for the RAG procedure [75]–[77].

1) Evaluation Metrics - Information Retrieval: The general results of information retrieval will return a list of the ranked candidates. Following the previous RAG and recommendation work in the NLP domain, which all related to the ranking task, we adopted similar evaluation metrics for the ranking results as **Precision@K**, **Recall@K**, **F1-Score@K**, **NDCG@K**, **HitRate@K** and **MAP** [75], [76]. These metrics are commonly used in evaluating ranking and recommendation systems. Precision@K measures the proportion of the relevant items in the top-k recommended items, focusing on the accuracy of the top suggestions. Recall@K assesses the fraction of all relevant items that appear in the top k recommendations, indicating how well the system retrieves relevant items overall. F1@K is the harmonic mean of precision@k and recall@k, providing a single metric that balances both precision and recall in the top-k results. MAP (Mean Average Precision) averages the precision scores obtained at the ranks where relevant items occur across multiple queries, reflecting the overall ranking quality. NDCG@K (Normalized Discounted Cumulative Gain) evaluates the ranking quality by considering the position of relevant items, giving higher scores when relevant items are ranked higher, and normalizing the score to allow comparisons. Finally, HitRate@K measures the percentage of users or queries for which at least one relevant item appears in the top-k results, offering a simple indicator of whether the system makes any correct recommendations at all.

Algorithm 2 Evaluation of ACs Using LLMs as Judges

```
Require: Generated AC, ground truth ACs split into test objectives
    $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$, user stories, and LLM judges $\mathcal{J} =$
    {Claude 3.5 Sonnet, Gemini 2 Flash, GPT4o}
Ensure: Accuracy metrics: Acc@Hit and Acc@Cor.
 1: Initialize hit counter $hit \leftarrow 0$ and correct counter $correct \leftarrow 0$
 2: for each test objective $T_i \in \mathcal{T}$ do
 3:     Retrieve the corresponding generated ACs segment for $T_i$
 4:     for each LLM judge $J \in \mathcal{J}$ do
 5:         Query $J$ with the ACs segment and $T_i$
 6:         Receive judgment $j \in$ {Full Cov., Partial Cov., Not Cov.}
 7:     end for
 8:     if all judges return at least Partial Coverage then
 9:         $hit \leftarrow hit + 1$ {Acc@Hit: test objectives are partially covered}
10:     end if
11:     if all judges return Full Coverage then
12:         $cor. \leftarrow cor. + 1$ {Acc@Cor.: test objectives are fully covered}
13:     end if
14: end for
15: Compute accuracy metrics:
16: Acc@Hit $\leftarrow \frac{hit}{n}$
17: Acc@Cor. $\leftarrow \frac{correct}{n}$
18: return Acc@Hit, Acc@Cor.
```

2) Evaluation Metrics - AC Generation: We adopt two types of evaluation metrics for AC generation. **Statistical Evaluation Metrics.** Inspired by previous work [15], we adopted similar statistical measures for acceptance criteria generation. These include *Semantic Similarity* using Sent-BERT [26], *ROUGE-1*, *ROUGE-2*, *ROUGE-L*, *BLEU*, and *Levenshtein distance* [81]–[83]. *Semantic Similarity* embeds each sentence and scores them with cosine similarity [26]. *ROUGE-1* / *ROUGE-2* / *ROUGE-L* match single words, word pairs, and the longest shared word sequence, respectively. *BLEU* counts n-gram matches and adds a brevity penalty to discourage very short outputs. *Levenshtein distance* counts the fewest single-character edits needed to turn one string into the other. **LLMs-as–judges from SE Perspective.** LLM-as-a-judge is commonly adopted in the evaluation of generative tasks, including RAG [84], [85]. However, they mostly execute on the semantic level and lack adaptation in specific domains. We introduce the LLMs-as-judges metric by manually segmenting the ACs to atomic objectives to ensure the AC-level evaluation (original single AC will cover several objectives, which will yield inconsistent evaluation results) and clearly defining the instructions written by software testing experts (e.g., evaluation rubric, AC definitions, etc). We manually segment the ACs into distinct points for each test objective in the ground truth, i.e., in Fig. 1's example, there are three objectives in the second AC, including "(1) I should see..", "The widget should have...", and "The widget should show...", we manually segment this AC into three ACs. The LLMs then assess whether these objectives are covered in the generated criteria on a point-by-point basis. We define three-level coverage for the atomic testing objectives, where Full Coverage. refers to all the segmented testing objectives that are covered in the generated ACs, Partial Cov. refers that the test objectives are mentioned but not fully covered, Not Covered. means the test objectives are completely not covered. Our evaluation framework defines two accuracy metrics: **Acc@Hit**, which measures partial coverage of test objectives, and **Acc@Cor.**, which indicates full coverage. Since ACs are not isolated from user stories, we further introduce two evaluation types to analyze the results from different perspectives. **Case acc. (C)** is the average accuracy calculated per user story, reflecting the model's generalization ability across different user stories. A higher case accuracy suggests stronger adaptability to varying contexts. **Point acc. (P)** measures accuracy for individual test objectives within the ground truth ACs, representing the absolute performance of the generation process. Variations between case and point accuracy reveal different tendencies in method performance. For instance, if a method achieves high point accuracy but relatively low case accuracy, it suggests that while the method can generate precise criteria for some user stories, it fails to generalize effectively across others. To mitigate potential bias in evaluation, we employ three LLMs and set temperature and *top_p* to 0 and 0.1 to maximally reduce the randomness. The details are specified in Algorithm 2, where the positive judgment is recorded only when all models *unanimously* agree. Considering the final judgment decisions are made by three LLMs, we compute Fleiss's κ [86] to illustrate the consistency of our settings for Table IV (0.62) and V (0.70), where the κ values are in acceptable range [87].

3) Evaluation Metrics - AC Polishing: We primarily adopt the evaluation metrics outlined in the AC generation process. Since polishing is not applied to all user stories, we introduce an additional LLM-annotated evaluation to compare the original and polished ACs. In this evaluation, we present the user story, original ACs, and polished ACs to three LLMs and ask them to determine which version is superior based on a predefined rubric. Compare accuracy is measured by only counting where all three LLMs *unanimously* judge the polished version is comparable better, similarly to the judgment methodology described in Section IV-D2.

E. Results and Discussion

RQ1. Table III presents the results of experiments with T-RAG and V-RAG alternatives (Table II). For T-RAG alternatives, ICRALM consistently outperforms SentBERT across all k values, i.e., in terms of retrieving the relevant textual documents, ICRALM performs the best. As for V-RAG models, three alternatives achieve similar results, where DSE is slightly better than $\text{HTML}_{\text{Full}}$, followed by $\text{HTML}_{\text{Prune}}$. During our retrieval process, $\text{HTML}_{\text{Full}}$, due to the length of the HTML code, has a longer processing time compared with the other two methods. Thus, we select DSE as the best alternative for V-RAG. Besides, we need to determine the number k in top-k settings, while we cannot pass all the top information to

TABLE III
RESULTS OF INFORMATION RETRIEVAL (RQ1)

Methods	MAP	@K=1					@K=5					@K=10					@K=20				
		P	**R**	**F1**	**nDCG**	**HR**	**P**	**R**	**F1**	**nDCG**	**HR**	**P**	**R**	**F1**	**nDCG**	**HR**	**P**	**R**	**F1**	**nDCG**	**HR**
SentBERT	10.35	5.56	5.56	5.56	5.56	5.56	2.22	11.11	3.70	9.06	11.11	2.22	22.22	4.04	12.47	22.22	1.67	33.33	3.17	15.21	33.33
ICRALM	**39.65**	**33.33**	**33.33**	**33.33**	**33.33**	**33.33**	**8.33**	**41.67**	**13.89**	**38.23**	**41.67**	**5.56**	**55.56**	**10.10**	**42.63**	**55.56**	**3.47**	**69.44**	**6.61**	**46.00**	**69.44**
DSE	**15.71**	**11.11**	**5.56**	**6.94**	**11.11**	**11.11**	**7.22**	19.21	9.88	15.04	**27.78**	**6.11**	29.86	**9.77**	**19.29**	36.11	**4.58**	38.89	**8.03**	**22.56**	38.89
$HTML_{Prune}$	11.97	5.56	3.70	4.17	5.56	5.56	4.44	15.28	6.58	10.56	19.44	4.17	25.69	6.92	14.54	30.56	3.89	**41.67**	6.95	19.62	**41.67**
$HTML_{Full}$	12.82	5.56	2.31	3.24	5.56	5.56	**7.22**	**19.68**	**9.92**	13.04	**27.78**	5.56	**30.56**	9.03	17.00	**41.67**	4.44	41.67	7.85	20.95	**41.67**

TABLE IV
RESULTS OF ACCEPTANCE GENERATION USING LLMS (RQ1)

Methods	LLMs-Annotated				Statistical					
	Hit(C)	**Cor(C)**	**Hit(P)**	**Cor(P)**	**Sim.**	**Leven.**	**BLEU**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
Claude $_{\text{Urial}}$	*61.81*	*13.67*	*57.02*	*12.40*	***72.86***	*1525*	0.1985	*0.3637*	*0.1448*	*0.2956*
Claude $_{\text{APEER}}$	61.80	14.91	56.20	14.88	71.23	1520	**0.2271**	0.3651	0.1433	0.2832
Claude $_{\text{Urial w/o VRAG}}$	57.48	12.97	53.93	10.20	71.94	*1525*	*0.2042*	0.3573	0.1417	0.2927
Claude $_{\text{w/o RAG}}$	48.72	5.62	37.60	3.72	70.64	1576	0.1830	0.3009	0.1004	0.2333
Gemini $_{\text{Urial}}$	***71.19***	*15.95*	***68.60***	*19.01*	*72.18*	1503	0.2161	***0.3940***	***0.1571***	***0.3682***
Gemini $_{\text{APEER}}$	69.93	**29.19**	64.05	**24.38**	71.24	1575	0.2209	0.3591	0.1401	0.3169
Gemini $_{\text{Urial w/o VRAG}}$	64.76	13.69	57.85	14.53	71.97	*1496*	*0.2206*	0.3883	0.1560	0.3626
Gemini $_{\text{w/o RAG}}$	46.37	10.32	34.71	7.44	71.03	1742	0.1653	0.3102	0.1041	0.2818
GPT4o $_{\text{Urial}}$	62.22	*13.31*	56.20	*13.64*	71.75	**1491**	0.1744	0.3796	0.1479	0.3540
GPT4o $_{\text{APEER}}$	60.79	12.11	56.20	15.70	71.02	1505	0.2037	0.3355	0.1158	0.2803
GPT4o $_{\text{Urial w/o VRAG}}$	*63.31*	9.93	*57.85*	8.64	*71.93*	1498	*0.2109*	*0.3847*	*0.1561*	*0.3561*
GPT4o $_{\text{w/o RAG}}$	48.06	8.31	35.54	5.79	71.16	1550	0.1644	0.2954	0.0795	0.2464

the generation step. Considering the input size for LLMs, we set top-5 for the textual and visual information retrieval. For ICRALM, top-5 results enable a balanced trade-off, which allows ICRALM to have acceptable recall (41.67) compared with the conditions of $@k = 10$ and $@k = 20$ while maintaining acceptable precision (8.33) compared with the condition of $@k = 1$. For V-RAG, we observed that with the increase in $@k$, the precision does not decrease significantly, but the recall incrementally increases. DSE approaches yield their highest F1 scores at $@k = 5$, indicating the best compromise between retrieving enough relevant documents (boosted recall) and avoiding the dilution of precision that occurs with higher $@k$ values. Thus, *we use ICRALM for T-RAG with top-*5 *results and DSE for V-RAG with top-*5 *results.*

Table IV presents the results of ACs generation using ICRALM and DSE, where italicized values indicate improvements on the ablation settings. We tested two prompt templates across three LLMs and included ablation results. In the w/o RAG setting, we retained the Urial prompt template but removed the image input. The findings in Table IV indicate several key observations: (1) The statistical results across all baselines are very similar, suggesting that statistical evaluations alone are insufficient to distinguish performance differences among the baselines; (2) Among the three LLM configurations, the versions incorporating RAG significantly outperform those without RAG, highlighting the importance of RAG integration in the ACs generation process; (3) Comparing the Urial w/o RAG version with the Urial version, we observe that Urial consistently outperforms Urial w/o RAG across most evaluation measures, particularly in correct accuracy. A potential explanation for this is that the textual background description provides sufficient contextual information to guide the generation process, but the inclusion of image input plays a crucial role in ensuring greater accuracy in ACs generation; (4) Of the three LLMs tested, Gemini 2 Flash achieves the best results across all prompt settings, including Urial, APEER, and w/o RAG. GPT-4o and Claude 3.5 Sonnet demonstrate comparable performance, following Gemini 2 Flash; (5) Between the two prompt templates, Urial performs better in terms of Hit@Accuracy, while APEER excels in Correct@Accuracy. This suggests that Urial leads to better relevance in generated ACs, whereas APEER enhances correctness; (6) Case-level accuracy is generally higher than point-level accuracy. Since different user stories correspond to varying numbers of ACs, this suggests that LLMs perform better on smaller cases rather than on complete, more complex ones; (7) Within each LLM, most statistical metrics, such as ROUGE-1, -2, and -L, and BLEU, exhibit trends similar to those observed in LLM-annotated evaluations.

Answer to RQ1: *With k=5, ICRALM (T-RAG) and DSE (V-RAG) give the best retrieval: +25% textual F1, +15% visual F1, and ∼50% less runtime than $HTML_{Full}$. Passing these top-5 hits to the generators, RAG boosts case-level accuracy 42.8→66.0% (+23.2 pp) and point-level accuracy 36.4 → 57.9% (+21.5 pp). Gemini 2 Flash tops the LLMs; Urial is most relevant (Hit@Acc 74.1%), while APEER is most correct (Correct@Acc 61.3%). Overall, the ICRALM+DSE pipeline lifts AC-generation F1 by 27% and accuracy by >20 pp vs the strongest non-RAG baseline.*

RQ2. Table V presents the polishing results of three LLMs using the Urial template. Our method incorporates two alternative reward models, UR3 and the Generative Verifier. We continue to experiment with all three LLMs in RQ2. Since the global reward model determines whether polishing is necessary, only a subset of user stories (≈20%) undergoes refinement, representing a small portion of the overall dataset. The results indicate that both UR3 and the Generative Verifier consistently improve the ACs in LLM-annotated evaluations. However, statistical results show inconsistent improvements. Specifically, (1) at the case level, improvements in Hit accuracy are comparable between UR3 and the Generative Verifier across all three LLMs, but UR3 demonstrates relatively greater enhancement in Correct accuracy; (2) At the point level, a similar trend is observed: UR3 yields higher Correct accuracy, while the Generative Verifier enhances the overall relatedness of the generated ACs; (3) Pairwise comparisons between the original and polished versions suggest that, from the LLMs' perspective, polishing generally improves the overall quality of the ACs; (4) Statistical results remain inconsistent, mirroring the trend observed in Table IV.

TABLE V
RESULTS OF ACCEPTANCE CRITERIA POLISHING (RQ2)

Methods	LLMs-Annotated					Statistical						
	Hit(C)	Cor(C)	Hit(P)	Cor(P)	Compare	Sim.	Leven.	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
Claude $_{\texttt{Urial}}$	61.81	13.67	57.02	12.40	NA	72.86	1525	0.1985	0.3637	0.1448	0.2956
Claude $_{\texttt{Urial w/ UR3}}$	69.08	16.68	62.05	14.73	**100.00**	72.28	1540	0.1993	0.3597	0.1430	0.2907
Claude $_{\texttt{Urial w/ Gen}}$	68.68	15.19	61.61	13.39	83.33	**72.95**	1525	0.2023	0.3628	0.1456	0.2946
Gemini $_{\texttt{Urial}}$	71.19	15.95	68.60	19.01	NA	72.18	1503	0.2161	**0.3940**	**0.1571**	**0.3682**
Gemini $_{\texttt{Urial w/ UR3}}$	**79.10**	**17.72**	**74.11**	**20.54**	**100.00**	71.79	1517	0.2180	0.3892	0.1529	0.3660
Gemini $_{\texttt{Urial w/ Gen}}$	**79.10**	**17.72**	**74.11**	**20.54**	**100.00**	71.80	1511	**0.2181**	0.3865	0.1545	0.3612
GPT4o $_{\texttt{Urial}}$	62.22	13.31	56.20	13.64	NA	71.75	1491	0.1744	0.3796	0.1479	0.3540
GPT4o $_{\texttt{Urial w/ UR3}}$	69.13	15.19	60.71	15.18	88.89	70.97	**1483**	0.1775	0.3807	0.1455	0.3501
GPT4o $_{\texttt{Urial w/ Gen}}$	71.91	14.79	64.73	14.73	**100.00**	71.39	1501	0.1749	0.3699	0.1396	0.3483

Answers to RQ2: *Applying polishing to the stories increases case-level accuracy by ∼7 pp and point-level accuracy by ∼5 pp. UR3 is the best choice when correctness is paramount, while the Generative Verifier is slightly better for relevance. With either reward model, Gemini 2 Flash remains the top LLM.*

RQ3. Expert assessment of 81 ACs across 17 user stories measured four dimensions: relevance, correctness, understandability (per AC), and coverage (per user story). Below we show response frequencies on a 5-point scale.

- **Relevance** [0, 3, 21, 31, 26] (average = 3.99)
- **Correctness** [3, 7, 14, 25, 32] (average = 3.94)
- **Understandability** [0, 0, 20, 24, 37] (average = 4.21)
- **Coverage** [0, 3, 4, 6, 4] (average = 3.89)

Experts rated all three criteria near 4 ("Agreed"), indicating that *RAGcceptance_M2RE*'s ACs are generally relevant, correct, and understandable. (1) This shows that the experts agree to a large extent that, on average, the AC generated by our approach is relevant to the underlying user story, correct, and understandable. (2) The correctness score is slightly low, which was due to redundancy in some generated ACs, i.e., LLMs will generate redundant ACs, essentially paraphrased versions of each other. This affects the correctness rating of all such ACs. In the three cases at correctness level 1, terminological issues were found in the generated criteria, and two were redundant in relation to a given user story. (3) Other issues related to ACs being low-rated on relevance, correctness, and understandability included not being particular and capturing the ACs at a very generic level. (4) In terms of coverage, in 7/17 user stories, the experts felt that the generated ACs failed to cover all cases that they would have wanted. In most cases, it was related to edge cases and negative inputs.

Qualitatively, experts were often impressed: Expert 1 noted "it's amazing how your system is able to capture bento box," a concept of UI container absent from the user story but surfaced by *RAGcceptance_M2RE*with RAG; Expert 2 said some ACs clarified the domain and "It's impressive and would be really useful for a new BA (business analyst)." They also expected time and effort savings. Expert 3 remarked, "This also gives some ACs which even as an expert I wouldn't have thought of." All expressed interest in deploying the approach, underscoring strong enthusiasm and practical utility. The negative comments from the expert primarily focus on the coverage, especially mentioning that "Sometimes the generated ACs are too abstract. We cannot get something useful from it."

Answers to RQ3: *Experts evaluated the quality of generated ACs across relevance, correctness, understandability, and coverage, assigning ratings mostly near 4 out of 5 (indicating agreement with quality). Minor correctness issues were primarily due to redundancy in LLM-generated outputs. Overall, experts showed strong enthusiasm for the generated criteria, highlighting instances where the approach captured valuable, non-obvious aspects. They expressed a clear interest in practical adoption.*

V. THREATS TO VALIDITY

Internal validity. Results could have been impacted by social-desirability bias [88]. Any such bias was addressed by ensuring our experts were blind to the system's internals during the interview and by instructing them to justify every decision in writing, a known remedy for this bias.

Construct validity. Two complementary families of metrics were used: (a) human-style judgements and (b) surface statistics (e.g., BLEU, ROUGE). Either family alone is imperfect, e.g., n-gram scores often miss semantic improvements, while LLM judgements can drift. Thus, we collaboratively reported where they diverged. This mitigates the risk that any single metric may misrepresent quality.

External validity. We acknowledge the potential limitation of a single industry study. Our study spans only two feature sets of *UniLearn*, so wider replications within *eSolutions* and beyond are needed. As all data here is proprietary, we have anonymized a small subset of the data and published every other technical artifact publicly available to aid reproducibility.

VI. CONCLUSION AND FUTURE WORK

In this paper, we introduce *RAGcceptance_M2RE*, an automated method for generating acceptance criteria derived from user stories, and demonstrate its efficiency through an industrial project from *eSolutions*. *RAGcceptance_M2RE* leverages multi-modal requirements processing by integrating advanced Large Language Models (LLMs) with both textual and visual retrieval-augmented generation (RAG) support. We evaluate *RAGcceptance_M2RE* on two samples from *eSolutions*, focusing on *UniLearn* course structuring for learners and overall system management for all users. Based on an interview survey with three experts from *eSolutions*, we assess the usefulness of *RAGcceptance_M2RE* in terms of correctness, relevance, understandability, and coverage. The experts agree with all four dimensions, indicating that *RAGcceptance_M2RE* achieves high human satisfaction from real-world application perspectives, by reducing manual effort and expanding analytical thinking.

Future work: We plan to (1) extend the multi-modal RAG pipeline to support additional test artifact generation, as our pipeline can be generalized; (2) validate *RAGcceptance_M2RE* across a broader set of industrial cases to further demonstrate its efficiency and generalization capabilities.

References

[1] J. Fischbach, J. Frattini, A. Vogelsang, D. Mendez, M. Unterkalmsteiner, A. Wehrle, P. R. Henao, P. Yousefi, T. Juricic, J. Radduenz *et al.*, "Automatic creation of acceptance tests by extracting conditionals from requirements: Nlp approach and case study," *Journal of Systems and Software*, p. 111549, 2023.

[2] A. G. Clark, N. Walkinshaw, and R. M. Hierons, "Test case generation for agent-based models: A systematic literature review," *Information and Software Technology*, p. 106567, 2021.

[3] S. Perala and A. Roy, "A review on test automation for test cases generation using nlp techniques," *Turkish Journal of Computer and Mathematics Education*, no. 6, pp. 1488–1491, 2021.

[4] J. Fischbach, H. Femmer, D. Mendez, D. Fucci, and A. Vogelsang, "What makes agile test artifacts useful? an activity-based quality model from a practitioners' perspective," in *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2020, pp. 1–10.

[5] R. Rwemalika, S. Habchi, M. Papadakis, Y. Le Traon, and M.-C. Brasseur, "Smells in system user interactive tests," *Empirical Software Engineering*, no. 1, p. 20, 2023.

[6] E. C. dos Santos and P. Vilain, "Automated acceptance tests as software requirements: An experiment to compare the applicability of fit tables and gherkin language," in *International conference on agile software development*. Springer, 2018, pp. 104–119.

[7] S. Karpurapu, S. Myneni, U. Nettur, L. S. Gajja, D. Burke, T. Stiehm, and J. Payne, "Comprehensive evaluation and insights into the use of large language models in the automation of behavior-driven development acceptance test formulation," *IEEE Access*, 2024.

[8] F. Wang, C. Arora, C. Tantithamthavorn, K. Huang, and A. Aleti, "Requirements-driven automated software testing: A systematic review," *ACM Transactions on Software Engineering and Methodology*, 2025.

[9] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, "Large language models for software engineering: Survey and open problems," in *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. IEEE, 2023, pp. 31–53.

[10] A. Nguyen-Duc, B. Cabrero-Daniel, A. Przybylek, C. Arora, D. Khanna, T. Herda, U. Rafiq, J. Melegati, E. Guerra, K.-K. Kemell *et al.*, "Generative artificial intelligence for software engineering–a research agenda," *arXiv preprint arXiv:2310.18648*, 2023.

[11] Z. Xue, L. Li, S. Tian, X. Chen, P. Li, L. Chen, T. Jiang, and M. Zhang, "Domain knowledge is all you need: A field deployment of llm-powered test case generation in fintech domain," in *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, 2024, pp. 314–315.

[12] X. Gu, M. Chen, Y. Lin, Y. Hu, H. Zhang, C. Wan, Z. Wei, Y. Xu, and J. Wang, "On the effectiveness of large language models in domain-specific code generation," *ACM Transactions on Software Engineering and Methodology*, no. 3, pp. 1–22, 2025.

[13] L. Welz and C. Lanquillon, "Enhancing large language models through external domain knowledge," in *International Conference on Human-Computer Interaction*. Springer, 2024, pp. 135–146.

[14] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, pp. 9459–9474, 2020.

[15] C. Arora, T. Herda, and V. Homm, "Generating test scenarios from nl requirements using retrieval-augmented llms: An industrial study," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*. IEEE, 2024, pp. 240–251.

[16] C. Arora, F. Wang, C. Tantithamthavorn, A. Aleti, and S. Kenyon, "From domain documents to requirements: Retrieval-augmented generation in the space industry," *arXiv preprint arXiv:2507.07689*, 2025.

[17] M. Cohn, *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[18] Artefact Repository of RAGcceptance M2RE. Online; accessed 15 Mar 2025. [Online]. Available: https://github.com/fanyuuwang/Multi_Modal_Requirements_Data_based_Acceptance_Criteria_Generation_using_LLMs

[19] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu, and Z. Liu, "Evaluation of retrieval-augmented generation: A survey," in *CCF Conference on Big Data*. Springer, 2024, pp. 102–120.

[20] S. Gupta, R. Ranjan, and S. N. Singh, "A comprehensive survey of retrieval-augmented generation (rag): Evolution, current landscape and future directions," *arXiv preprint arXiv:2410.12837*, 2024.

[21] X. Du, G. Zheng, K. Wang, J. Feng, W. Deng, M. Liu, B. Chen, X. Peng, T. Ma, and Y. Lou, "Vul-rag: Enhancing llm-based vulnerability detection via knowledge-level rag," *arXiv preprint arXiv:2406.11147*, 2024.

[22] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, and Q. Li, "A survey on rag meeting llms: Towards retrieval-augmented large language models," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 6491–6501.

[23] Z. Hu, A. Iscen, C. Sun, Z. Wang, K.-W. Chang, Y. Sun, C. Schmid, D. A. Ross, and A. Fathi, "Reveal: Retrieval-augmented visual-language pre-training with multi-source multimodal knowledge memory," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 23 369–23 379.

[24] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, no. 1. Citeseer, 2003, pp. 29–48.

[25] A. Trotman, A. Puurula, and B. Burgess, "Improvements to bm25 and language models examined," in *Proceedings of the 19th Australasian Document Computing Symposium*, 2014, pp. 58–65.

[26] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[27] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlgay, A. Shashua, K. Leyton-Brown, and Y. Shoham, "In-context retrieval-augmented language models," *Transactions of the Association for Computational Linguistics*, pp. 1316–1331, 2023.

[28] W. Chen, H. Hu, X. Chen, P. Verga, and W. W. Cohen, "Murag: Multimodal retrieval-augmented generator for open question answering over images and text," *arXiv preprint arXiv:2210.02928*, 2022.

[29] M. Yasunaga, A. Aghajanyan, W. Shi, R. James, J. Leskovec, P. Liang, M. Lewis, L. Zettlemoyer, and W.-t. Yih, "Retrieval-augmented multimodal language modeling," *arXiv preprint arXiv:2211.12561*, 2022.

[30] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, no. 7676, pp. 354–359, 2017.

[31] D. Reber, S. Richardson, T. Nief, C. Garbacea, and V. Veitch, "Rate: Score reward models with imperfect rewrites of rewrites," *arXiv preprint arXiv:2410.11348*, 2024.

[32] Y. Liu, Z. Yao, R. Min, Y. Cao, L. Hou, and J. Li, "Rm-bench: Benchmarking reward models of language models with subtlety and style," *arXiv preprint arXiv:2410.16184*, 2024.

[33] G.-H. Lee and Y.-N. Chen, "Muse: Modularizing unsupervised sense embeddings," *arXiv preprint arXiv:1704.04601*, 2017.

[34] A. Hüyük, W. R. Zame, and M. van der Schaar, "Inferring lexicographically-ordered rewards from preferences," in *Proceedings of the AAAI Conference on Artificial Intelligence*, no. 5, 2022, pp. 5737–5745.

[35] L. Eshuijs, S. Wang, and A. Fokkens, "Balancing the scales: Reinforcement learning for fair classification," *arXiv preprint arXiv:2407.10629*, 2024.

[36] S. Kim, J. Suk, S. Longpre, B. Y. Lin, J. Shin, S. Welleck, G. Neubig, M. Lee, K. Lee, and M. Seo, "Prometheus 2: An open source language model specialized in evaluating other language models," *arXiv preprint arXiv:2405.01535*, 2024.

[37] Z. Wang, A. Bukharin, O. Delalleau, D. Egert, G. Shen, J. Zeng, O. Kuchaiev, and Y. Dong, "Helpsteer2-preference: Complementing ratings with preferences," *arXiv preprint arXiv:2410.01257*, 2024.

[38] A. S. Thakur, K. Choudhary, V. S. Ramayapally, S. Vaidyanathan, and D. Hupkes, "Judging the judges: Evaluating alignment and vulnerabilities in llms-as-judges," *arXiv preprint arXiv:2406.12624*, 2024.

[39] H. Li, Q. Dong, J. Chen, H. Su, Y. Zhou, Q. Ai, Z. Ye, and Y. Liu, "Llms-as-judges: a comprehensive survey on llm-based evaluation methods," *arXiv preprint arXiv:2412.05579*, 2024.

[40] L. Zhang, A. Hosseini, H. Bansal, M. Kazemi, A. Kumar, and R. Agarwal, "Generative verifiers: Reward modeling as next-token prediction," *arXiv preprint arXiv:2408.15240*, 2024.

[41] X. Yuan, Z. Yang, Y. Wang, J. Zhao, and K. Liu, "Improving zero-shot LLM re-ranker with risk minimization," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 17 967–17 983.

[42] A. Mustafa, W. M. Wan-Kadir, N. Ibrahim, M. A. Shah, M. Younas, A. Khan, M. Zareei, and F. Alanazi, "Automated test case generation from requirements: A systematic literature review," *Computers, Materials and Continua*, no. 2, pp. 1819–1833, 2021.

[43] H. S. Dar, S. Imtiaz, and M. I. Lali, "Reducing requirements ambiguity via gamification: comparison with traditional techniques," *Computational Intelligence and Neuroscience*, no. 1, p. 3183411, 2022.

[44] H. Dar, R. Aziz, J. A. Khan, M. I. Lali, and N. A. Almujally, "Gamify4lexamb: a gamification-based approach to address lexical ambiguity in natural language requirements," *PeerJ Computer Science*, p. e2229, 2024.

[45] P. K. Chittimalli and M. J. Harrold, "Regression test selection on system requirements," in *Proceedings of the 1st India software engineering conference*, 2008, pp. 87–96.

[46] A. Almohammad, J. F. Ferreira, A. Mendes, and P. White, "Reqcap: Hierarchical requirements modeling and test generation for industrial control systems," in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2017, pp. 351–358.

[47] R. P. Verma and M. R. Beg, "Representation of knowledge from software requirements expressed in natural language," in *2013 6th International Conference on Emerging Trends in Engineering and Technology*. IEEE, 2013, pp. 154–158.

[48] M. G. Dos Santos, S. Hallé, F. Petrillo, and Y.-G. Guéhéneuc, "Aat4irs: automated acceptance testing for industrial robotic systems," *Frontiers in Robotics and AI*, p. 1346580, 2024.

[49] E. Bjarnason, M. Unterkalmsteiner, M. Borg, and E. Engström, "A multi-case study of agile requirements engineering and the use of test cases as requirements," *Information and Software Technology*, pp. 61–79, 2016.

[50] A. Brockenbrough, H. Feild, and D. Salinas, "Exploring llms impact on student-created user stories and acceptance testing in software development," in *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2*, 2025, pp. 1401–1402.

[51] B. Güldali, H. Funke, S. Sauer, and G. Engels, "Torc: test plan optimization by requirements clustering," *Software Quality Journal*, pp. 771–799, 2011.

[52] C. Wang, F. Pastore, A. Goknil, and L. C. Briand, "Automatic generation of acceptance test cases from use case specifications: an nlp-based approach," *IEEE Transactions on Software Engineering*, no. 2, pp. 585–616, 2020.

[53] T. Straszak and M. Śmialek, "Automating acceptance testing with tool support," in *2014 Federated Conference on Computer Science and Information Systems*. IEEE, 2014, pp. 1569–1574.

[54] C. Arora, J. Grundy, and M. Abdelrazek, "Advancing requirements engineering through generative ai: Assessing the role of llms," in *Generative AI for Effective Software Development*. Springer, 2024, pp. 129–148.

[55] A. Vogelsang and J. Fischbach, *Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline*. Cham: Springer Nature Switzerland, 2025, pp. 435–456.

[56] K. Huang, F. Wang, Y. Huang, and C. Arora, "Prompt engineering for requirements engineering: A literature review and roadmap," *arXiv preprint arXiv:2507.07682*, 2025.

[57] Y. Liang, J. Wang, H. Zhu, L. Wang, W. Qian, and Y. Lan, "Prompting large language models with chain-of-thought for few-shot knowledge base question generation," *ArXiv*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:263909537

[58] C. Jin, H. Peng, S. Zhao, Z. Wang, W. Xu, L. Han, J. Zhao, K. Zhong, S. Rajasekaran, and D. N. Metaxas, "Apeer: Automatic prompt engineering enhances large language model reranking," *arXiv preprint arXiv:2406.14449*, 2024.

[59] E. Cambria, L. Malandri, F. Mercorio, N. Nobani, and A. Seveso, "Xai meets llms: A survey of the relation between explainable ai and large language models," *arXiv preprint arXiv:2407.15248*, 2024.

[60] J. Jiao, S. Afroogh, Y. Xu, and C. Phillips, "Navigating llm ethics: Advancements, challenges, and future directions," *arXiv preprint arXiv:2406.18841*, 2024.

[61] F. Fang, Y. Bai, S. Ni, M. Yang, X. Chen, and R. Xu, "Enhancing noise robustness of retrieval-augmented language models with adaptive adversarial training," *arXiv preprint arXiv:2405.20978*, 2024.

[62] K. Zhu, X. Feng, X. Du, Y. Gu, W. Yu, H. Wang, Q. Chen, Z. Chu, J. Chen, and B. Qin, "An information bottleneck perspective for effective noise filtering on retrieval-augmented generation," *arXiv preprint arXiv:2406.01549*, 2024.

[63] Meta, "Llama-3.2-3B-Instruct," online; accessed 15 Jan 2025.

[64] abi, "Screenshot-to-code," online; accessed 15 Jan 2025.

[65] J. Tan, Z. Dou, W. Wang, M. Wang, W. Chen, and J.-R. Wen, "Htmlrag: Html is better than plain text for modeling retrieved knowledge in rag systems," *arXiv preprint arXiv:2411.02959*, 2024.

[66] Tevatron, "dse-phi3-docmatix-v2," online; accessed 15 Jan 2025.

[67] Sentence Transformers, "all-MiniLM-L12-v2 ," online; accessed 15 Jan 2025.

[68] Meta, "Llama-3.1-8B-Instruct," online; accessed 15 Jan 2025.

[69] Atlassian, "Jira," online; accessed 15 Jan 2025.

[70] R. Likert, "A technique for the measurement of attitudes." *Archives of psychology*, 1932.

[71] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "Retrieval augmented language model pre-training," in *International conference on machine learning*. PMLR, 2020, pp. 3929–3938.

[72] P. Zhao, H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, J. Jiang, and B. Cui, "Retrieval-augmented generation for ai-generated content: A survey," *arXiv preprint arXiv:2402.19473*, 2024.

[73] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

[74] H. Li, Y. Su, D. Cai, Y. Wang, and L. Liu, "A survey on retrieval-augmented text generation," *arXiv preprint arXiv:2202.01110*, 2022.

[75] H. Fang, G. Guo, D. Zhang, and Y. Shu, "Deep learning-based sequential recommender systems: Concepts, algorithms, and evaluations," in *Web Engineering: 19th International Conference, ICWE 2019, Daejeon, South Korea, June 11–14, 2019, Proceedings 19*. Springer, 2019, pp. 574–577.

[76] Y. Xie, J. Lin, H. Dong, L. Zhang, and Z. Wu, "Survey of code search based on deep learning," *ACM Transactions on Software Engineering and Methodology*, no. 2, pp. 1–42, 2023.

[77] J. Ou, J. Zhang, Y. Feng, and J. Zhou, "Counterfactual data augmentation via perspective transition for open-domain dialogues," *arXiv preprint arXiv:2210.16838*, 2022.

[78] Anthropic, "Claude," 2025, online; accessed 15 Jan 2025.

[79] DeepMind, "Gemini," online; accessed 15 Jan 2025.

[80] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford *et al.*, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.

[81] L. Borges, B. Martins, and P. Calado, "Combining similarity features and deep representation learning for stance detection in the context of checking fake news," *Journal of Data and Information Quality (JDIQ)*, no. 3, pp. 1–26, 2019.

[82] T. Hu and X.-H. Zhou, "Unveiling llm evaluation focused on metrics: Challenges and solutions," *arXiv preprint arXiv:2404.09135*, 2024.

[83] V. Hanke, T. Blanchard, F. Boenisch, I. Olatunji, M. Backes, and A. Dziedzic, "Open llms are necessary for current private adaptations and outperform their closed alternatives," *Advances in Neural Information Processing Systems*, pp. 1220–1250, 2024.

[84] S. Es, J. James, L. E. Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 2024, pp. 150–158.

[85] H. Wei, S. He, T. Xia, F. Liu, A. Wong, J. Lin, and M. Han, "Systematic evaluation of llm-as-a-judge in llm alignment tasks: Explainable metrics and diverse prompt templates," *arXiv preprint arXiv:2408.13006*, 2024.

[86] R. Falotico and P. Quatto, "Fleiss' kappa statistic without paradoxes," *Quality & Quantity*, no. 2, pp. 463–470, 2015.

[87] J. L. Fleiss, "Review papers: The statistical basis of meta-analysis," *Statistical methods in medical research*, no. 2, pp. 121–145, 1993.

[88] P. Grimm, "Social desirability bias," *Wiley international encyclopedia of marketing*, 2010.