SN Computer Science (2026) 7:320
https://doi.org/10.1007/s42979-026-04844-6

ORIGINAL RESEARCH

# Specializing LLMs to Low-Documented Domains with RAG: An Analysis Across Models and Retrieval Depths

**Jacopo Mereu[1] · Alessandro Carcangiu[1] · Valentino Artizzu[1] · Federico Maria Cau[1] · Lucio Davide Spano[1]**

Received: 19 August 2025 / Accepted: 18 February 2026


**Abstract**

Large Language Models (LLMs) are increasingly used to support technical tasks such as software development. However, they often struggle in low-documented or fast-evolving domains, where missing training data leads to inaccurate or incomplete responses. This paper presents a reproducible pipeline based on Retrieval-Augmented Generation to specialize LLMs for such domains by integrating curated external knowledge. We detail a systematic process to build a high-quality Q&A dataset from public instructional sources and developer forums and apply it to the Unity XR Interaction Toolkit (XRIv2) as a case study. We construct a domain-specific benchmark of 101 question-answer pairs based on real learning resources and evaluate five open and proprietary LLMs (GPT-3.5-Turbo, GPT-4o Mini, LLaMA2, LLaMA3, and Mistral) under varying retrieval settings. Results show that standard automatic metrics (e.g., METEOR) struggle to detect quality differences, while LLM-as-a-Judge evaluations reveal significant model-specific improvements as more documents are retrieved. Our findings offer practical guidance for tuning retrieval strategies and highlight the potential for generalizing this approach to other technical domains requiring targeted LLM specialization.

**Keywords** Large language model · Question answering · Retrieval augmented generation · Domain adaptation · Extended reality · Text generation metric evaluation

## Introduction

The fast improvement of Large Language Models (LLMs) [1–4] and their availability through relatively simple APIs has opened up new opportunities to automate, or at least speed up, a wide range of tasks. Some people use LLMs to translate texts from one language to another [5], others to summarize documents [6], while others exploit them to fill-in missing parts in a text [7]. Instead, developers increasingly use LLMs as assistants to address specific technical issues, taking advantage of these models' ability to answer questions [8].

According to a survey run in 2023[1] involving 3240 programmers, 36.6% (1186 developers) had 0–1 years of professional development experience and, among these, 34% (404 developers) used ChatGPT daily, while 61% (726 developers) used ChatGPT weekly. This shows early-career developers highly rely on LLMs to carry out their tasks.

General-purpose models have access to extensive documentation and numerous examples for established development technologies. However, they struggle with less documented topics, such as specialized toolkits or libraries for cutting-edge technologies. For instance, they are of little help when addressing questions about (1) private documents or tools internal to a company or (2) concepts and tools introduced after the model's training date. This can result in non-helpful or factually incorrect answers, including phenomena technically referred to as " hallucinations ".

✉ Jacopo Mereu
jacopo.mereu@unica.it

Alessandro Carcangiu
alessandro.carcangiu@unica.it

Valentino Artizzu
valentino.artizzu@unica.it

Federico Maria Cau
federicom.cau@unica.it

Lucio Davide Spano
davide.spano@unica.it

[1] Department of Mathematics and Computer Science, University of Cagliari, Via Ospedale 72, 09124 Cagliari, Italy

[1] https://zerotomastery.io/blog/the-state-of-ai-tools-and-coding-2023-edition/.

Published online: 28 March 2026

Retrieval Augmented Generation (RAG) [9] has been introduced in response to these challenges. RAG operates by enhancing the generator model (LLM) with a retriever model, which provides relevant context to the generator. This context includes both the original input and additional information sourced by the retriever from an external knowledge database. While RAG systems can improve the generator model's performance, they do not entirely eliminate the risk of hallucinations. Nevertheless, RAG presents an interesting approach to explore, as it requires *no* direct modification to the generator model.

This paper aims to address the following problem: if we are working with a topic whose documentation is scarce or not available during the model training, the LLM may be unable to provide adequate assistance. But if we can provide curated sources about this topic, we can use RAG to exploit the additional knowledge. However, there are many technical and performance issues to face during this process. This paper investigates the following questions:

1. How can we prepare the documentation available to support RAG?
2. To what extent does a retriever enhance the performance of a model through knowledge augmentation?
3. How can we balance the trade-off between the number of documents included in the LLM context and the answer quality?

We present a pipeline for constructing RAG systems through a case study in the domain of development frameworks. We assume the existence of a developer who is willing to transfer their knowledge through this pipeline to help themselves or other developers in the domain. An example would be a Senior Developer utilizing this pipeline to transfer their knowledge to junior developers within their organization, as usually happens in companies, big and small. To illustrate the process, we augment several LLMs (GPT-3.5-Turbo,[2] GPT-4o Mini,[3] Mistral OpenOrca 7B,[4] Llama3 8B,[5] and Llama2 7B[6]) to answer questions on an extended reality (XR) development toolkit: Unity XR Interaction Toolkit version 2[7] (XRIv2). We selected this toolkit because, even though the XR market is growing,[8] it remains relatively niche compared to more established fields in software development [10, 11]. Many domains—including software libraries, scientific toolkits, and industrial frameworks—suffer from the same knowledge access bottleneck: rich documentation and forum content exist, but are fragmented and not encoded into the LLMs. The proposed methodology is not limited to Unity or XRI; it offers a replicable solution for enabling LLMs to support real-world users in navigating poorly indexed or rapidly evolving technical domains.

We created a benchmark composed of 101 question-answer pairs focused on XRIv2 to evaluate our RAG-based conversational agent. Novice developers generated the questions, while expert developers provided the answers. This benchmark covers key topics such as tool setup, locomotion, interaction handling, user interface creation, accessibility, and performance optimization. We evaluated the five aforementioned LLMs using our benchmark, varying the level of augmentation by adjusting the number of retrieved documents (K). Traditional word-based (METEOR) and character-based (Normalized Edit Distance) metrics struggled to capture meaningful differences between generated and ground-truth answers. In contrast, we show that LLM-as-a-Judge better assesses the quality of the generated answers. Our analysis showed that increasing the number of retrieved documents generally improved judge scores, with significant gains occurring with the increase of low numbers of retrieved documents (e.g., passing from 0 to 4), which becomes stable after a model-dependent threshold. Such an optimal number varied across models, emphasizing the need for model-specific strategies when incorporating additional knowledge.

In this paper, we make the following contributions:

- We propose a reproducible RAG pipeline designed for low-documented or fast-evolving technical domains, combining web scraping, automatic Q&A generation, and integration with multiple LLMs.
- We introduce a domain-specific benchmark built from real-world learning content (video courses), annotated for conceptual coverage and difficulty.
- We perform a multi-model, multi-metric evaluation of retrieval augmentation results, revealing that LLM answer quality improves with additional retrieved content, but differently across models.
- We offer a model-specific analysis of Top-K retrieval tuning, highlighting how augmentation strategies should be adapted to different LLM architectures.

[2] https://platform.openai.com/docs/models#gpt-3-5-turbo.

[3] https://platform.openai.com/docs/models#gpt-4o-mini.

[4] https://huggingface.co/TheBloke/Mistral-7B-OpenOrca-GGUF.

[5] https://huggingface.co/QuantFactory/Llama-3-8B-16K-GGUF.

[6] https://huggingface.co/TheBloke/Llama-2-7B-32K-Instruct-GGUF.

[7] https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/.

[8] https://academyofanimatedart.com/virtual-reality-statistics/.

## Related Work

### Retrieving Information with Large Language Models

In recent years, considerable research has been conducted into the capabilities and limitations of Large Language Models (LLMs) in conversational AI [12, 13]. A prevalent challenge encountered in basic AI language models is the issue of " hallucination " [14], where generated text may be grammatically sound and plausible but lacks accuracy or truthfulness.

To mitigate the problem of hallucinations, the concept of Retrieval-Augmented Generation (RAG) [9] has been introduced by Facebook AI Research. In its basic form, RAG combines a Retriever model with a Generator model (represented by an LLM). When a user asks a question (query), the Retriever searches an external knowledge database containing both documents and their embeddings. It matches the user's query embedding with document embeddings using algorithms like K-Nearest Neighbors (kNN) or Approximate Nearest Neighbors (ANN) and similarity distances such as cosine distance. The Retriever then returns the most relevant documents to the Generator, which utilizes both the user question and the retrieved documents to generate the answer.

The RAG architecture has been explored in various domains, including religion [15], medicine [16], and FAQs [17], particularly within free-form question-answering systems. Research on advanced forms of RAGs adds processes before and after retrieving the documents to improve the retrieval quality. Operations like (user) query rewriting [18], expansion [19] or reranking [20] the retrieved results make the systems more complex, increasing both performance but also latency. Latency is an important factor to consider when building a RAG system: using advanced pre-processing or post-processing techniques or having a large knowledge database may exceed the latency constraints of your system. Recent research focuses on techniques to tune the quality of generated responses by RAG models. Akimoto et al. [21] indicate that the quality of context used during training directly impacts the performance of RAG models. Their findings suggest that models trained with high-quality contexts exhibit better coherence and relevance in their outputs. Another critical aspect of tuning RAG models is the selective augmentation of inputs. The architectural design of RAG systems also plays a crucial role in their tuning. The evolution of RAG from a simple " retrieve-then-read " model to a more modular and flexible framework has allowed for the incorporation of advanced components such as Query Rewriters, which enhance knowledge retrieval by generating search-friendly queries [22]. This modularity facilitates targeted tuning of individual components, enabling researchers to refine the retrieval process and improve the overall coherence and relevance of the generated text. Additionally, the introduction of mechanisms like the Pluggable Reward-Driven Contextual Adapter (PRCA) can refine the retrieved information, thereby improving the performance of retrieval question answering tasks [23]. In this paper, we investigate the effects of controlling the number of documents to retrieve for augmenting the context.

### Assessment of the Generated Answers

To assess the quality of the generated response, the literature exploits various techniques or metrics to measure the similarity between a generated text and a reference text. Classic metrics such as BLEU/ROUGE/METEOR [24–26] are based on the concept of shared N-grams, which are consecutive word sequences present in both the generated response and the reference response. The disadvantage is that if the model responds with words significant to the context different from those used in the correct response, the obtained sequence will differ. Thus, the metric would unfairly return a low value. Character-based metrics, such as the (Normalized) Edit Distance [27], measure the difference between two texts based on the number of edit operations required to transform one into another. Edit operations are insertions, deletions, and character replacements, ignoring semantic meaning, so even semantically distant texts can have a small edit distance. Another technique involves using unit testing [28], where functions are given the generated response and the reference and return true or false depending on whether the test is passed. Examples of unit testing could include checking the presence or absence of certain information. The problem with unit testing is that it does not cover all scenarios and is challenging to create automatically. A third approach is to use a " powerful " language model (LLM) as a judge [29]. The problem is that it introduces bias depending on the chosen model: is that model truly capable of judging the response as a human would? A fourth approach could be based on the semantic similarity between the text embeddings [30], while remaining aware that semantic proximity is not synonymous with factual accuracy.

In [31], the authors analyzed 28 RAG-related papers and proposed a taxonomy comprising 61 characteristics, organized into 5 meta-dimensions and 16 dimensions. One of these dimensions concerns evaluation metrics, distinguishing two characteristics: retrieval evaluation and generation evaluation. The latter focuses on the quality of the generated output and includes metrics such as those based on N-grams, which we also discuss. However, in contrast to us, their analysis does not address how the number of retrieved documents affects the generated output according to standard metrics. He et al. [32] offered an insightful analysis of the impact of using dense versus sparse retrievers in RAG applications for coding tasks. The authors also examine how the number of retrieved documents affects N-gram-based metrics and exact pattern matching. However, in contrast to this work, the evaluation does not include LLM-based metrics, such as LLM-as-a-Judge.

In this paper, we show the effectiveness of these techniques in the literature in assessing the differences between answers generated by different LLMs and varying the number of RAG-retrieved documents.

### Large Language Models for Software Engineering

Hou et al. [33] present the most comprehensive survey to date on the application of LLMs in Software Engineering (SE), reviewing 395 papers, $\approx 61\%$ of which were sourced from arXiv. The survey provides insights into three areas. The first is related to *model architectures and their applications in SE tasks.* The authors observe that decoder-only Transformer models are the most common architecture, appearing in roughly 49% of the reviewed works, while the remaining studies employ encoder-only or encoder–decoder architectures. Our models follow this trend, as they are also decoder-only. The second is about the *dataset categorization.* Four dataset types are identified in the review: open-source, collected, constructed, and industrial. While authors note the role of " constructed " datasets, neither the survey nor the cited research describes how to systematically build such datasets. In contrast, our methodology fills this gap by defining a repeatable pipeline for (i) identifying, (ii) scraping, and (iii) transforming raw web and forum content into a high-quality text-based Q&A dataset. The survey emphasizes the importance of ensuring high data quality through preprocessing, suggesting, for text-based data, a list of steps ranging from their extraction to their segmentation into smaller chunks. Our methodology wraps this principle holistically. By generating a question first and an answer second, our approach segments only the most relevant and coherent portions of a document. The last is dedicated to the different *Data types*: text-based, code-based, graph-based, software-repository-based, and combined datasets. Our work specifically targets text-based Q&A pairs. Among the 395 surveyed papers, only six adopt a similar format, and only three of them [34–36] have been published in peer-reviewed venues. ChatDev [34] is a multi-agent framework for software development assistance, leveraging automatic generation and human-guided post-processing. It is a more generalised approach than ours, focusing on multi-agent collaboration rather than dataset construction. Von der Mosel et al. [35] compares domain-specific BERT variants against general-domain transformers, using datasets mined from sources like GitHub issues and Stack Overflow. However, its focus is on contextual understanding rather than structured dataset creation. Moreover, the four data sources they use align with categories in our proposed source taxonomy (see Sect. 3.1).

Texeira de Lima et al. [37] shifts focus from RAG methodology to the data itself, addressing challenges in data labeling and dataset imbalances that can distort RAG system design. The authors introduce a taxonomy of query types to classify (context, query) pairs, showing that common public Q&A datasets are often skewed toward simpler factual questions. They also propose strategies for generating more balanced and grounded synthetic Q&A datasets. In contrast to us, they didn't explore how to practically retrieve the data, which was instead obtained from public datasets. Regarding how they generated Q&A from data, they adopted an inverted generation approach: instead of generating questions first, they extract factual, summary, or inferred statements from the context and then generate questions that can be unambiguously answered by those statements.

In summary, while prior work has explored RAG for coding tasks and domain adaptation, it mostly focuses on high-resource domains or assumes the availability of structured datasets. Our approach uniquely targets *low-documented technical domains* and provides a domain-agnostic pipeline from data acquisition to model evaluation. Moreover, we analyze how the number of retrieved documents affects LLM output quality using both statistical and qualitative evaluation, offering practical guidance for real-world deployment of RAG systems.

## RAG Pipeline and Benchmark Construction

This section describes a generalizable methodology for building RAG systems tailored to emerging or low-documented technical domains, exemplified through a case study on Unity's XR Interaction Toolkit (XRIv2).

To find an appropriate topic requiring augmentation for getting high-quality responses from an LLM, we considered eXtended Reality (XR) development in Unity. Such a domain, despite being a growing market, is still niche compared to more established fields in software development [10, 11], considering also that there are many toolkits available and no clear winner among them. In such a context, basic tasks such as installing the tool, enabling interactions, or deploying the project to a headset are not intuitive for beginners. The official XRIv2 documentation can also be inadequate, as it sometimes contains knowledge gaps that can hinder the understanding of the tool. Considering the lack of a benchmark providing ground-truth data for assessing the quality of generated answers through ground-truth data, we constructed our benchmark using two online course materials as described in Sect. 4.

## Retrieving and Structuring Domain Knowledge

To support domain-specific retrieval, our RAG system requires access to a curated external corpus. We focused on publicly available sources to ensure reproducibility, although private documentation—such as internal notes or presentations—could further enhance performance.

We categorized knowledge sources into four types: technical articles, supervised forums (e.g., Stack Overflow), unsupervised forums (e.g., Reddit), and instructional materials (e.g., MOOCs or YouTube courses). We excluded video content due to copyright concerns.

Figure 1 illustrates this processing pipeline, which involves three stages:

1. Link collection and filtering via keyword and tag-based queries.
2. HTML structure parsing to isolate question–answer pairs or content blocks.
3. Scraping and formatting using custom scripts.

In the final version, each scraped source is structured as a question–answer pair, $r_i = (q_i, a_i)$. This representation, together with associated metadata, is used to populate a vector store[9] for retrieval. Each record is keyed by the embedding of the answer, and the metadata includes the question, question ID, the answer itself, the source type (whether it's a document, a supervised source, or an unsupervised forum), and the source origin (the website domain). More details are available in Sect. 3.3.

Before reaching this final structure, the sources go through an intermediate standardized format that depends on the type of source. This preprocessing step is explained further in Sect. 3.2.

## Processing Information

Our goal is to produce a RAG-agent expert capable of answering free-form questions. For this reason, it is useful to represent the taxonomy category documents in the format of question-answer pairs.

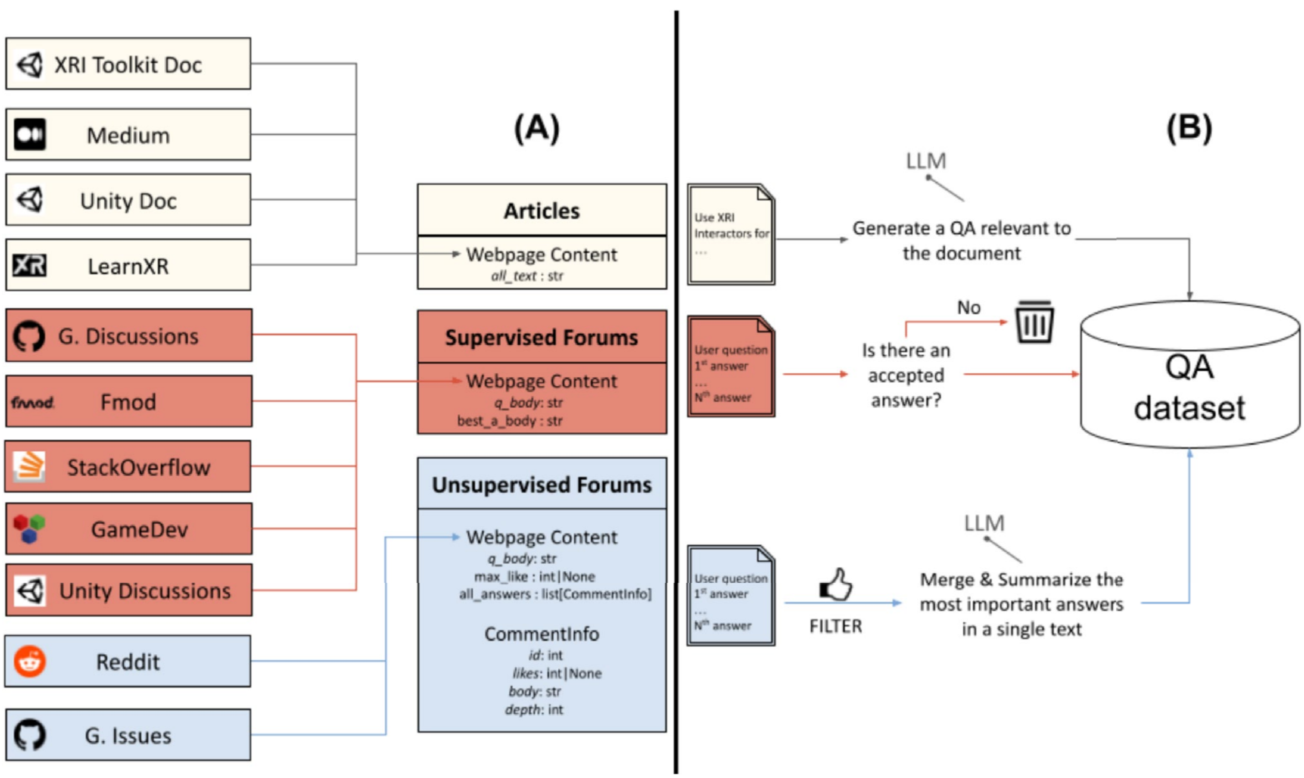


**Fig. 1** Overview of the pipeline used to construct the domain-specific retrieval dataset. **A** Examples of scraped sources for the Unity XRIv2 domain, including official documentation, forums, and tutorials. **B** Processing workflow: documents are scraped, parsed, and converted into standardized question–answer pairs for embedding and indexing. While the scraping process is domain-dependent, the remaining stages of the pipeline are domain-agnostic and reproducible

[9] In our implementation, we use Pinecone as the underlying vector database; however, the proposed methodology is independent of the specific vector store technology.

When a document already fits this structure, we can use it without changes; otherwise, we extract question-answer pairs using an LLM, GPT-4-Omni in our case. Figure 1B depicts this process for each source category.

For Articles, since there are neither questions nor answers, we have to generate both of them. We rely on prompt D.1 for this task. In this prompt, we ask the model to perform three sub-tasks: (1) provide a self-explanation of the article document, (2) generate a question based on the document, and (3) answer the question. The self-explanation (1) is intended only to enhance the accuracy of the question generation (2) and the response (3). We deliberately perform all three subtasks within a single generation to (a) reduce token usage and (b) latency, and (c) to preserve reasoning continuity between self-explanation, question, and answer generation, following chain-of-thought-inspired practices. We then use regular expressions (RegEx) to find and separate these three components in the model's output.

For Supervised Forums, we use the original question and, if available, the accepted answer; otherwise, the document is discarded.

For Unsupervised Forums, we have a question but lack a verified, correct answer, so we need to generate one. If no answers are present, the document is discarded. When the forum uses an upvote/like system where users rate responses, we filter out the least liked comments. We then use prompt D.2 to ask the model to generate a unified answer. The model's task is to read all remaining comments, judge which are useful, and create a unique answer. We don't apply post-processing with RegEx, as we ask the model to output only the unified answer.

### Creating the RAG

To enable retrieval-augmented generation, we represent each item in our curated dataset as a tuple $r_i = \{q_i, a_i\}$, where $q_i$ is a question and $a_i$ the corresponding answer. Using OpenAI's Ada v2 model, we generate an embedding of each answer:

$$\mathrm{emb}_{a_i} = \mathrm{ADA2}(a_i)$$

We store each embedding with its associated question–answer pair as metadata in a vector store.[10] Each resulting entry is represented as:

$$w_i = \{\mathrm{emb}_{a_i}, \mathrm{metadata} = \{q_i, a_i\}\}$$

[10] In our implementation, we use Pinecone as the underlying vector store and LangChain for integration; however, the proposed approach is independent of the specific vector store or orchestration framework.

At query time, the user question is embedded using the same embedding model:

$$\mathrm{emb}_q = \mathrm{ADA2}(q)$$

The retriever computes cosine similarities between this embedding and the stored answer embeddings,[11] selecting the top-$k$ closest entries:

$$\text{Top-}k \leftarrow \arg\max_j \ \mathrm{cosine\_similarity}(\mathrm{emb}_q, \mathrm{emb}_{a_j})$$

The retrieved answers are used as augmented context for the LLM, prepended to the user query. While questions serve the extraction stage and metadata, the retrieval itself relies only on answer embeddings. Additional implementation details are provided in Appendix B.

## Evaluation

We evaluate the performance of our retrieval-augmented QA system using a domain-specific benchmark derived from XRIv2 learning materials. This section describes the construction of the benchmark, the evaluation design, the experimental setup, the metrics used, and the results obtained.

To validate our RAG agent, we needed a dataset or benchmark containing questions and corresponding ground-truth answers related to the case study (XRIv2). Ideally, novice developers should post these questions, and expert developers should provide the answers. Even overlooking this constraint, no such publicly available benchmarks exist. We thus built a benchmark of 101 question-answer pairs to thoroughly evaluate the agent's performance in the domain of interest (see Sect. 4.1). Two students formulated the questions, while three authors (experts in XRIv2) provided the corresponding answers. The benchmark covers topics like tool setup, player locomotion, interaction handling, enhancing player immersivity, creating user interfaces, improving accessibility, and optimizing performance. Furthermore, we tested the benchmark with several LLMs and metrics, such as METEOR, Cosine similarity, and LLM-as-a-Judge. Finally, we conducted an additional quantitative analysis to evaluate how increasing document exposure affects judge scores across different language models (see Sect. 4.4). Specifically, we aimed to determine the minimum number of documents ($K > 0$) required for a significant improvement and the K that yielded the greatest impact, highlighting differences in how models benefit from additional documents.

[11] We omit question embeddings to avoid complexity and to isolate answer?based retrieval effects.

## The Question Benchmark

To assess the system's real-world utility, we designed a benchmark grounded in authentic learning trajectories for Unity's XR Interaction Toolkit v2 (XRIv2). The goal was to obtain domain-representative, difficulty-aware questions that reflect the challenges novice developers typically face, an aspect often lacking in synthetic benchmarks.

To avoid overlap with the knowledge base and prevent potential bias, we deliberately excluded any sources used in its construction. Instead, we selected two well-reviewed instructional video courses as the foundation for our benchmark: a comprehensive 19-hour course published on Unity Learn[12] (covering version 2.5.2), and a 3-h YouTube tutorial[13] (covering version 2.4.3). Both target beginners and introduce essential concepts in Unity's XR development workflow.

Two Master's students with basic Unity knowledge—gained through an undergraduate course in game design—were recruited to formulate the benchmark questions. The collection process unfolded in two phases. First, each student independently watched the two courses and produced questions based on the content, aiming to formulate inquiries that reflected either explicit learning goals (e.g., " How can I control a Unity object from afar with a Ray Interaction? ") or more exploratory doubts inspired by the demonstrations. Each student generated two question sets—one per course—without discussing their ideas with the other, to preserve diversity and avoid mutual influence.

In the second phase, the students collaboratively merged their four lists, removing duplicates and editing for clarity and consistency. This process resulted in a final set of 101 unique questions.

The answers were written by three of the authors, all of whom also watched the two reference courses. Each author responded independently to a subset of questions and reviewed answers authored by the others. All discrepancies were discussed and resolved collaboratively to ensure consistency, technical correctness, and coverage.

The resulting benchmark comprises 101 question–answer pairs and is provided as a JSON file, on a GitHub repository.[14]

***Benchmark Coverage Analysis*** To evaluate the representativeness of our benchmark with respect to the Unity XR Interaction Toolkit documentation, we performed a two-part qualitative analysis.

First, each question–answer pair was annotated with: (1) a *conceptual category* identifying the main topic it addresses, and (2) a *difficulty level* estimating how challenging the answer is for a learner to understand. It is worth pointing out that we are not considering the difficulty for an LLM to generate a correct answer here.

Figure 2 shows the distribution of the benchmark across twelve topic categories, which comprise:

- **Setup**: configuration and project preparation.
- **Interaction**: manipulation and user-object interaction.
- **Locomotion**: player movement strategies.
- **UI**: immersive user interface design and behavior.
- **Scripts**: implementing or customizing behavior through scripting.
- **Build & Deploy**: application packaging and deployment.
- **Theory**: conceptual foundations or design rationales.
- **Optimization**: performance tuning and resource management.
- **Event**: working with event-driven logic and callbacks.
- **Immersivity**: enhancing presence and realism.
- **Hands**: controlling and representing virtual hands.
- **Accessibility**: supporting diverse users and input modalities.

Second, to assess the overall benchmark's representativeness on the target knowledge domain, we assessed benchmark coverage against the library documentation hierarchy. We manually analyzed the structure published on the official website, pruning filler and redundant pages to retain only relevant content. Each leaf node (i.e., lowest-level topic) was marked as *covered* (green) if its content was addressed by at least one benchmark question; otherwise, it was marked *uncovered* (red). Parent nodes were recursively marked green if the majority of their children were green.

This procedure revealed a balanced distribution: 7 out of 14 top-level documentation categories were covered. While the benchmark addresses core areas such as interaction, input, and setup, it leaves out more specialized features such as advanced customization mechanisms and extended UI capabilities. These topics are typically covered in advanced scenarios or case-specific tutorials and are rarely part of introductory materials or early-stage developer needs. Their absence is therefore consistent with the learning context in which the benchmark was elicited. Appendix E includes the full documentation tree with coverage annotations.

## Experimental Setup

The goal of our evaluation is to determine whether augmenting LLMs with a specialized knowledge base improves their ability to answer questions about Unity's XR Interaction Toolkit. We evaluate this across different retrieval depths and LLMs using both automatic metrics and preference

[12] https://learn.unity.com/learn/pathway/vr-development.

[13] https://www.youtube.com/watch?v=YBQ_ps6e71k.

[14] https://github.com/cg3hci/unity-XRIv2-qa.

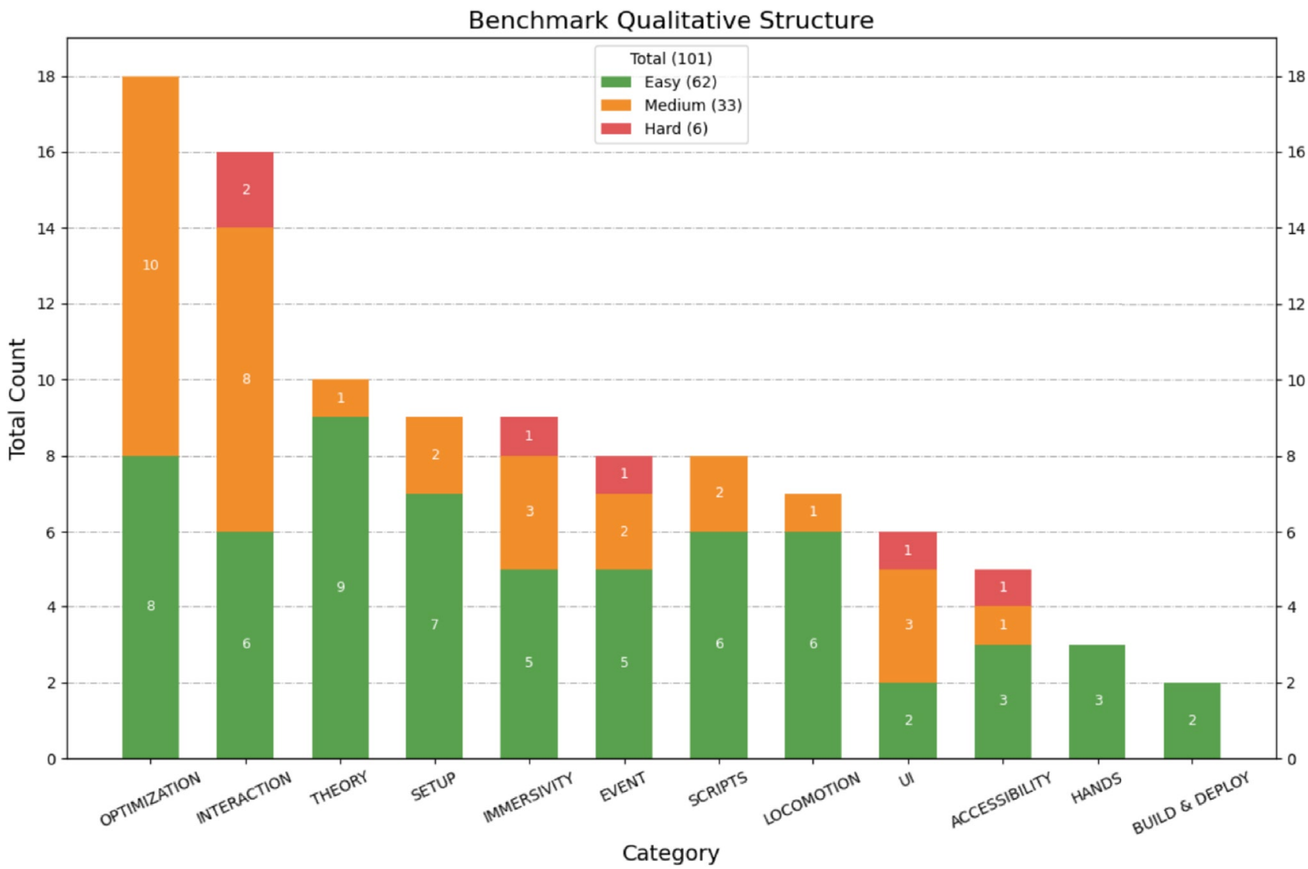


**Fig. 2** Distribution of benchmark questions by topic category and difficulty level. The benchmark spans a wide range of topics in XR development, with annotations reflecting both conceptual focus and estimated learning complexity

judgments from an LLM judge. The design allows for comparing baseline (no retrieval) with different RAG configurations ($k \in [1, 12]$), across five language models.

We evaluated the pipeline detailed in Sect. 3 by using the following LLMs: GPT-3.5-Turbo, GPT-4o Mini, Llama2 7B, Llama3 8B, and Mistral 7B. We refer to a model (LLM) *M*, augmented with a Retriever, as *RAG* (*M*, *k*), where *k* is the number of closest answers the Retriever picks from the database knowledge for each query. *RAG* (*M*, 0) represents the base LLM without augmentation. A *Test*(*M*, *k*) is defined as running all the benchmark question-answer pairs once through a specific *RAG* (*M*, *k*). For each question-answer record $r_i = (q_i, a_i)$, *RAG* (*M*, *k*) generates an answer $a_{M,k}$ for the input question $q_i$. It also pre-computes the embeddings $emb_{a_{M,k}}$ of such an answer and keeps track of the generation time in seconds.

All models were configured with a temperature of 0 to ensure determinism. However, for OpenAI's proprietary models, this setting does not fully eliminate variability. To account for this, we repeated the experiment three times (twice in October 2024 and once in February 2025) and measured the variability in their outputs. Figure 6 reports the median and interquartile range (IQR) of the per-question score differences across runs.

The generated answer is then compared to the reference answer $a_i$ using the following metrics: METEOR, Normalized Edit distance, and LLM-as-a-Judge. For the latter, we used GPT-4 Omni as a judge to assess the quality of the answers using a simple scoring system (see prompt D.3). The judge assigns a score between 1 (worst) and 5 (best) based on how many hallucinations the judge thinks are present in the generated answer. The judge model is unaware of which model (*M*) generated the answer or how many documents (*K*) the model had access to when it generated the answer. Finally, we also track the cosine similarity between the generated and reference answer embeddings.

## Results

Figure 3 shows the experiment results on the GPT models in 3 different runs, while Fig. 4 shows the experiment results on the open-source LLMs. We show each metric's mean and standard deviation at the change of the number of documents retrieved. Regarding the patterns visible in the charts, cosine similarity and METEOR tend to be constant

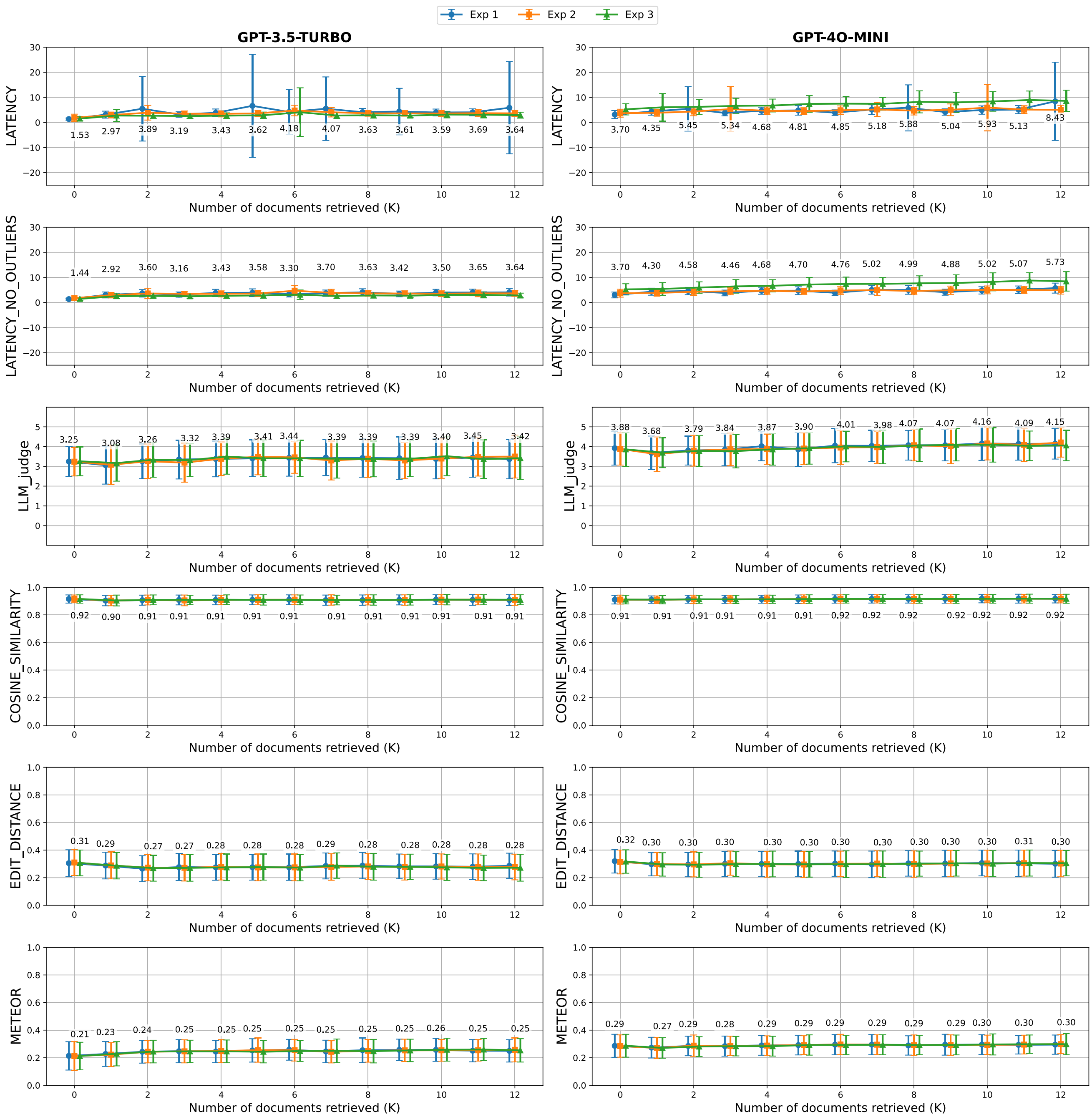


**Fig. 3** Benchmark results for the GPT models (GPT-3.5-Turbo and GPT-4o Mini) across three experimental runs, showing minimal metric fluctuation. Each column represents a different model, while within each plot the three curves represent individual runs conducted in October 2024 (Experiment 1 and 2) and February 2025 (Experiment 3). All plots report the mean (marker) with ± 1 standard deviation, where the marker height indicates the mean and the vertical lines denote the standard deviation. To improve visual readability, curves are slightly shifted horizontally; since the horizontal axis represents the integer-valued number of retrieved documents $K \in [0, 12]$, this offset does not affect interpretation. From top to bottom, the plots show: (i) latency (in seconds) for generating one answer (lower is better); (ii) latency (in seconds) without outliers beyond three standard deviations (lower is better); (iii) judge score assigned to the LLM (ranging from 1 to 5, higher is better); (iv) cosine similarity (0–1, higher is better); (v) normalized edit distance (0–1, lower is better); and (vi) METEOR score (0–1, higher is better)

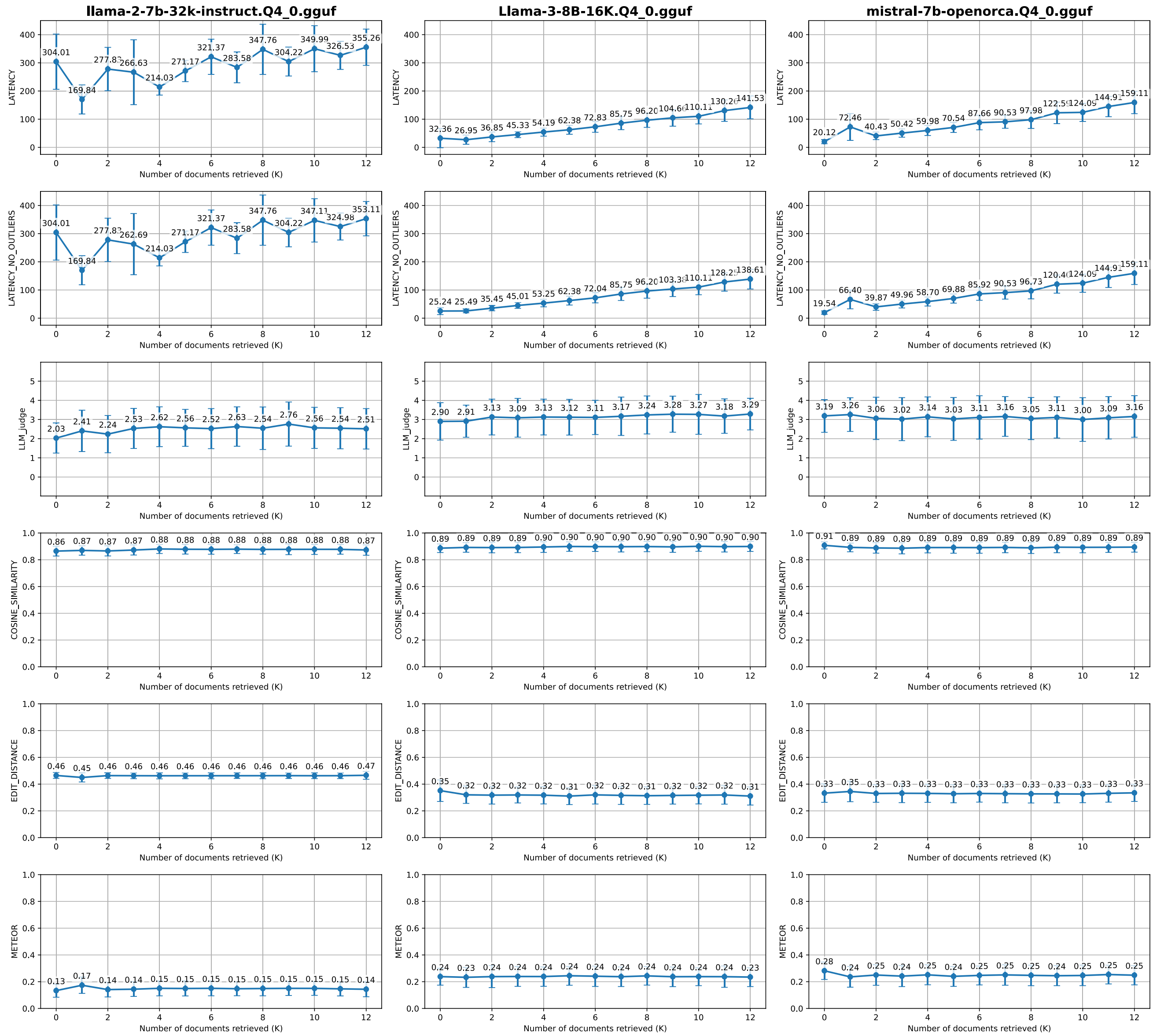


**Fig. 4** Benchmark results for the open-source models (LLama2, LLama3, Mistral). The arrangement follows the same format as the previous figure, only there is one single run for each model since they are deterministic. Each column represents a different model: leftmost is Llama2, middle is Llama3, and rightmost is Mistral. All graphics show the mean with ± 1 standard deviation. From top to bottom, the graphics depict: (i) latency (in seconds) for generating one answer (lower is better); (ii) latency (in seconds) without outliers beyond three standard deviations (lower is better); (iii) judge score assigned to the LLM (ranging from 1 to 5, higher is better); (iv) cosine similarity (0–1, higher is better); (v) normalized edit distance (0–1, lower is better); and (vi) METEOR score (0–1, higher is better)

as the change of K and of the model. Normalized edit distance tends to be constant too, making exceptions for the GPT models and Llama3, in which there is some positive change in the growth of the number of documents. These metrics don't seem to reflect any changes in how the models are performing, whether it's an improvement or a decline.

The Judge score shows an improvement in the growth for K for all models except Mistral, which remained overall constant. We recall that when the judge generated the score, it had access only to the human question, ground-truth answer, and the generated answer. In Fig. 5, we present the count of each judge's score for each model. The highest count of 5 (indicating the highest score) is recorded for $K = 12$ across all models, except for Llama2. Interestingly, the lowest count of 1 (worst score) is also recorded for $K = 12$, except for Mistral and GPT-3.5-Turbo. This indicates that the augmentation does not simply bring positive effects but also some negative ones. Still, overall, augmented models achieve a higher proportion of scores in the 4–5 range and a lower proportion in the 2–3 range than the non-augmented ones.

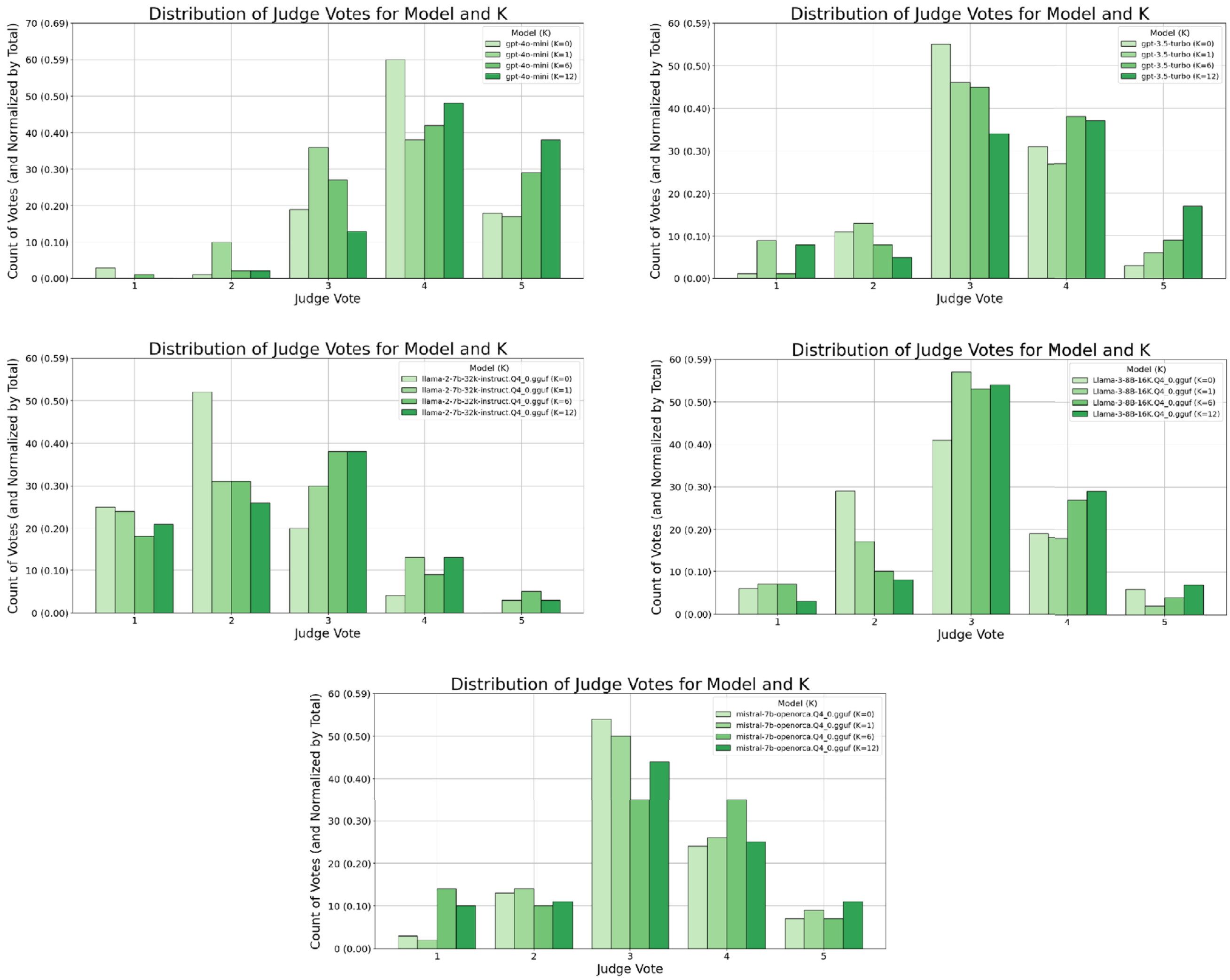


**Fig. 5** Grouped histograms of judge scores across varying retrieval levels (K) for each model. Within each subfigure, the horizontal axis represents the possible judge scores (1–5), where a higher score indicates that the LLM judge considered the answers to be of higher quality with fewer hallucinations. The vertical axis represents the frequency of each score, shown both in absolute terms and normalized. Bar colors indicate the level of retrieval augmentation (K), with lighter shades representing lower values of K. To reduce visual clutter, only four retrieval levels are shown (K=0, 1, 6, and 12) out of the full set of thirteen (K=0 through 12)

## Quantitative Post-Analysis of Top-K Document Influence on Judge Scores

To assess the impact of document exposure on judge scores, we fitted a Cumulative Link Mixed Model (CLMM), where the judge score (assigned by GPT-4 Omni) served as the ordinal dependent variable, representing the quality of answers generated by the five LLMs (i.e., GPT-3.5-Turbo, GPT-4o Mini, Llama2 7B, Llama3 8B, and Mistral 7B). The number of documents (K) was included as a fixed effect, while the five LLMs were modeled as a random intercept. For the CLMM analysis, we used only a single iteration for GPT models, given the minimal variation in the interquartile ranges and median values (see Fig. 6). Additionally, we conducted pairwise contrasts for each model to determine the point at which increasing document exposure produced the most pronounced and statistically significant changes in judge scores. To achieve this, we investigated two key thresholds: the first significant K increase, defined as the smallest number of documents for which the contrast between $K = 0$ and $K > 0$ was statistically significant, and the best K, defined as the number of documents ($K > 0$) needed to achieve the greatest improvement compared to $K = 0$. The ordinal regression analysis (CLMM) revealed that while lower values of documents (e.g., K = 1, 2) did not significantly improve judge scores compared to K = 0, a significant positive effect emerged starting at K = 4 (*Log-Odds* = 0.33, *Std. error* = 0.11, *z-value* = 2.96, $p$ =.003). This effect strengthened with increasing K, reaching its highest improvement at K = 12 (*Log-Odds* = 0.59, *Std. error* = 0.11,

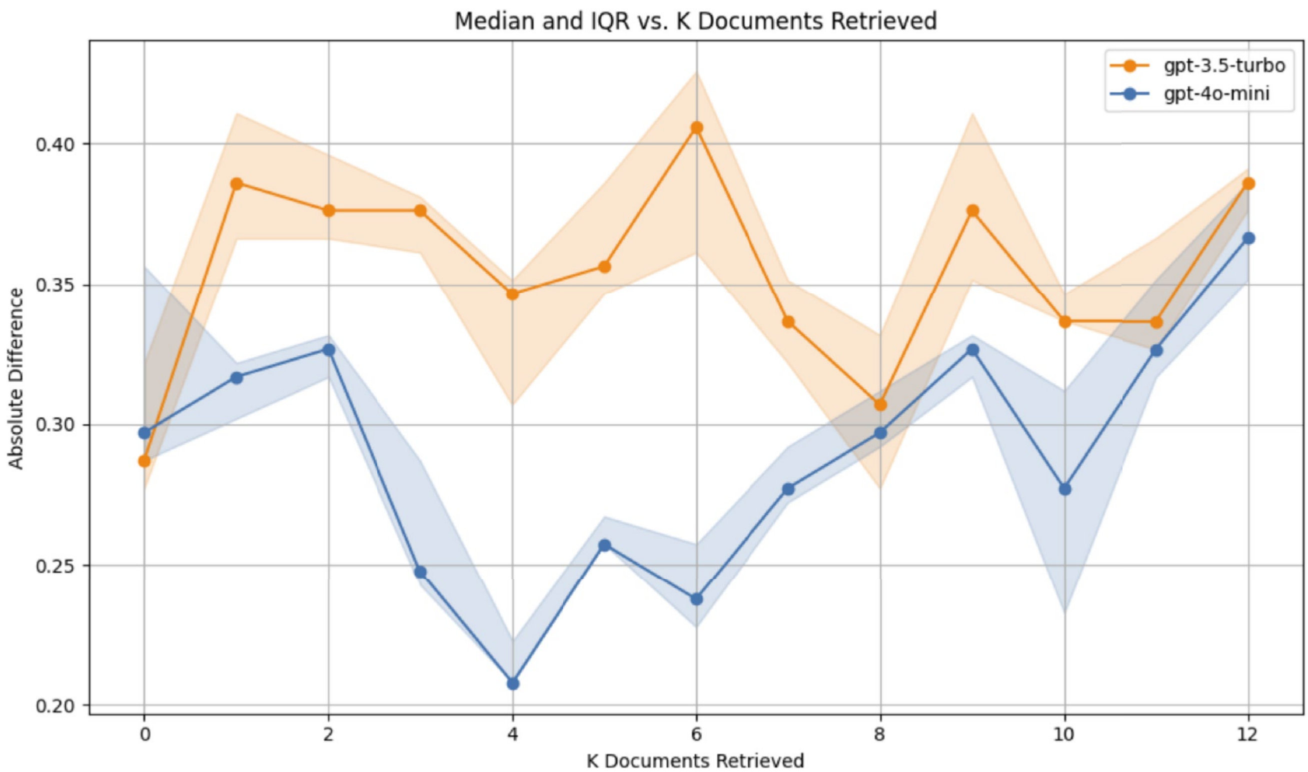


**Fig. 6** Median and interquartile range (IQR) of per-question absolute Judge Scores (JS) differences across three experimental runs. For each of the 101 question-answer pairs, we computed absolute differences across all possible run pairings ($|R1_{JS} - R2_{JS}|$, $|R1_{JS} - R3_{JS}|$, $|R2_{JS} - R3_{JS}|$), resulting in three vectors of size 101. The mean of each vector was then calculated, producing three summary values. The plot displays the median (Q2) of these values as a point chart, with the shaded area representing ±1 IQR (Q3–Q1). The LLM as judge assigned scores from 1 (minimum) to 5 (maximum)

*z-value* = 5.21, $p < .0001$), suggesting that providing more documents to the LLMs-answer generator generally leads to higher quality ratings by the judge.

To further examine the effect of document exposure considering each LLM, we conducted pairwise contrasts using single ordinal regression analyses for each model. Since the judge score represents the quality of answers generated by the five models, these analyses allowed us to assess how document exposure influenced answer quality across different LLMs. The results indicate that the number of documents $K > 0$ where we found significance varied across models. Llama2 7B exhibited an earlier improvement at K = 3 (*Log-Odds* = 0.83, *Std. error* = 0.25, *z-value* = 3.34, *p* =.009), with the best improvement occurring at K = 9 (*Log-Odds* = 1.23, *Std. error* = 0.25, *z-value* = 4.83, $p < .0001$). Instead, Llama3 8B showed its first significant improvement at K = 8 (*Log-Odds* = 0.76, *Std. error* = 0.27, *z-value* = 2.85, *p* =.042), with the strongest effect at K = 9 (smallest *p*-value; *Log-Odds* = 0.90, *Std. error* = 0.26, *z-value* = 3.39, *p* =.008). GPT-4o Mini required the highest number of documents, reaching significance only at K = 12 (*Log-Odds* = 0.73, *Std. error* = 0.26, *z-value* = 2.81, *p* =.0472). Also, GPT-3.5-Turbo and Mistral 7B did not show significant improvements for any K number of documents, suggesting their ratings remained stable regardless of document exposure. These findings underscore the importance of retrieval tuning as a model-specific hyperparameter, a dimension largely unexplored in existing RAG literature.

## Qualitative Observations

In this section, we present and analyze selected examples of model-generated responses showing the effects we registered during the experiments. More precisely, we are providing a positive and a negative example and a differences-in-style example. As stated in Sect. 4.2, each benchmark question was answered by five different LLMs across 13 levels of retrieval augmentation ($K \in [0, 12]$), but to limit the length of the text, we focus on the LLM and on the augmentation levels that better represent the effects we discuss. We report for reproducibility the references to the retrieved documents, along with their corresponding original sources. We recall that the retrieved documents may represent elaborations or reformulations of the original materials, rather than verbatim excerpts.

### Improved Specificity Through Retrieval

This example shows the expected effect that retrieving documents from the knowledge base should have on the LLM answer, which is the increase in its specificity. We discuss the answers to the following question ($ID = 40$ in the benchmark), as answered by Llama 3 8B.

" *I'm working on a Unity VR game. The scene contains an XR Ray Interactor, associated with the game objects representing the VR device's controllers. How can I associate a Direct and a Ray Interactor to each hand?* "

The analysis highlights a progression in the quality of the system's responses across different K values. At K=0 (Judge score=1), the output was vague and uninformative, referring to non-existent classes like XRDevices and using generic object references such as gameObj1, which were not grounded in the Unity XRI v2 architecture or its naming conventions. This initial response lacked specificity and practical value, making it difficult for users to derive any actionable insights.

At K=1 (Judge score=2) and K=2 (Judge score=3), the responses slightly improved as the system drew insights from retrieved documents F.1 and F.2. While K=1 introduced some relevant ideas within Unity's framework, it deviated from best practices by suggesting separate XRInteractionManagers for each hand and inaccurately naming properties. By K=2, the addition of the second document allowed for a more coherent incorporation of components like the Action Based Controller Manager. Despite this advancement, the guidance still lacked the operational specificity necessary for novice developers.

A significant increase in quality was observed at K=3 (Judge score=4), where the response exemplified much higher technical precision. It accurately referenced script and component names such as XR Ray Interactor (Action-based), XR Interaction Group, and XRI Default Right Controller, while outlining a configuration process aligned with established XR Toolkit patterns. Interestingly, although a teleport interactor was not mentioned in the original question, its contextual inclusion was appropriate, given typical implementations. The substantial improvement was largely driven by insights from the third retrieved document F.3, which provided a relevant overview, underscoring the importance of understanding prompt structure and retrieval conditioning in enhancing RAG system output.

See Listing 1 for the worst generated answer (Judge score = 1) related to this example and Listing 2 for the best (Judge score = 4).

```
You need two separate XRInteractors for left and right hands. You can
    create them in code using:

‘‘‘
var direct = new XRDirectInteractor();
direct.interactableObjects.Add(gameObj1);
direct.interactableObjects.Add(gameObj2);
var ray = new XRRayInteractor();
ray.interactableObjects.Add(gameObj3);
‘‘‘

Then you need to assign the interactors to each hand using:
‘‘‘
XRDevice leftHand = XRDevices.GetLocalizedName("Left Hand");
leftHand.SetInteractionSystem(direct, ray);

XRDevice rightHand = XRDevices.GetLocalizedName("Right Hand");
rightHand.SetInteractionSystem(ray);
‘‘‘

This will allow you to use the Direct Interactor for interactions with
     gameObj1 and 2 using left hand controller, while Ray Interactor
    is used for interaction with gameObj3.

Note that this assumes there are only two hands in your scene. If you
    have more than two hands, you’ll need separate interactors for
    each additional hand as well.
```

**Listing 1** Output of Llama3 8B for benchmark question $Q_{id} = 40$ with no retrieval(K=0), and Judge score = 1

```
To achieve this setup in your project:

1) Remove any unused components from both hands (e.g., XR Controller).

2) Add new Game Objects for each hand:
- For the left hand, add an XR Ray Interactor and a Direct Interactor
    by right-clicking > XR > Ray Interactor (Action-based) and XR >
    Direct Interactor (Action-based). Remove the XR Controller
    component from both of these interactions.
- Repeat this step for the right hand.

3) Set up the Teleport Interactor:
- Add another XR Ray Interactor and rename it to "Teleport Interactor
    ".
- Under the XR Controller (Action-based) component, click on the
    preset icon and select "XRI Default Right Controller".
- Uncheck "Enable Input Tracking" for Position Action, Rotation Action
    , and Tracking State Action.
- Replace the default reference of Select Action with "XRI LeftHand
    Locomotion/Teleport Select".

4) Add new components to each hand:
- For both hands, add an Action Based Controller Manager component and
     configure it accordingly.
- Also on both hands, add an XR Interaction Group component that
    contains the XR Interaction Manager.
- Create two elements for Starting Group Members list and drop in the
    Ray Interactor and Direct Interactor.

This setup will allow you to use the joystick on each hand controller
    separately for teleportation without interfering with object
    selection. You can then switch between different interaction modes
     seamlessly by activating/deactivating components as needed.
```

**Listing 2** Output of Llama3 8B for benchmark question $Q_{id}$ = 40 with retrieval(K=3), Judge score = 4

#### Retrieval-Induced Confusion

The second example shows instead the less frequent case where increasing the number of retrieved documents decreases the quality of the answer given by the same LLM. We consider the question with $ID = 60$ and GPT-4o Mini as target LLM: " *Is there an easy way to create a Dialog?* "

The origin of the problem is the lack of good sources related to this question in the knowledge base used for retrieval. Thus, the LLM is forced to consider information having a low relevance for the question and the overall quality decreases.

At K=0 (Judge score=3), the system produced a high-level, structurally sound response for creating a UI dialog in Unity but lacked specific XRIv2 details. At K=1 (Judge score=2, retrieved documents: F.4), the introduction of an irrelevant forum post caused a significant shift toward audio integration with FMOD, demonstrating how a single off-topic document can bias output. At K=2 (Judge score=3, retrieved documents: F.4, F.5), a second irrelevant document nullified the distortion, leading to a response similar to K=0. However, by K=3 (Judge score=1, retrieved documents: F.4, F.5, F.6), the accumulation of irrelevant documents overwhelmed the model, resulting in an uninformative reply. This degradation continued at K=4 (Judge score=1, retrieved documents: F.4, F.5, F.6, F.7), where additional FMOD-related content reinforced the misdirection. These findings highlight the critical need for relevance in document selection, as irrelevant documents can suppress useful knowledge in RAG systems.

See Listing 3 for the worst generated answer (Judge score = 1) related to this example, and Listing 4 for the best (Judge score = 3).

```
I don't have specific information on creating a Dialog in Unity XR
    Interaction Toolkit v2. However, you can refer to the FMOD
    documentation for guidance on dialogue and localization, which may
     provide insights into managing dialogues in Unity. Here are some
    links that could be helpful:

- [FMOD Dialogue and Localization Documentation](https://fmod.com/
    resources/documentation-studio?version=2.0&page=dialogue-and-
    localization.html)
- [FMOD Studio Guide on Dialogue and Localization](https://fmod.com/
    resources/documentation-api?version=2.0&page=studio-guide.html#
    dialogue-and-localization)

If you have any specific questions or need further assistance, feel
    free to ask!
```

**Listing 3** Output of GPT-4o Mini for benchmark question $Q_{id}$ = 60 with retrieval(K=4), Judge score = 1

```
Since you haven't provided specific knowledge about Unity XR
    Interaction Toolkit v2, I will rely on my base knowledge.

In Unity, creating a dialog system can be approached in various ways,
    but there isn't a built-in dialog system specifically within the
    XR Interaction Toolkit v2. However, you can create a dialog by
    using UI elements such as Canvas, Text, and Button components.

Here's a simple approach to create a dialog:

1. **Create a Canvas**: In your Unity scene, create a new Canvas (
    GameObject > UI > Canvas).

2. **Add a Panel**: Inside the Canvas, add a Panel (Right-click on
    Canvas > UI > Panel) to serve as the background for your dialog.

3. **Add Text**: Add a Text element (Right-click on the Panel > UI >
    Text) to display your dialog message.

4. **Add Buttons**: If you want to have options for the user to
    respond, you can add Button elements (Right-click on the Panel >
    UI > Button) for user interaction.

5. **Scripting**: Create a script to manage the dialog's visibility
    and the logic for what happens when buttons are clicked. You can
    use Unity's Event System to handle button clicks.

6. **XR Interaction**: If you want to interact with the dialog using
    XR controllers, you can use the XR Interaction Toolkit to set up
    interactions with the buttons.

This is a basic outline, and you can expand upon it based on your
    specific needs, such as adding animations, sound effects, or more
    complex dialog management. If you have any specific requirements
    or need further assistance, feel free to ask!
```

**Listing 4** Example 4: Output of GPT-4o Mini for benchmark question $Q_{id}$ = 60 withno retrieval (K=0), Judge score = 3

### LLMs Differences in Style

The different LLMs we used in this study had some relevant stylistic differences we summarise here.

**GPT-3.5-Turbo** adopts a notably concise style, producing succinct responses that become increasingly specific with additional context. This progression moves from general references to teleportation settings toward detailed mentions of components, such as the XR Interactor Line Visual.

**GPT-4o Mini** employs a more structured, verbose, and instructional approach, consistently organizing its outputs into clearly defined, step-by-step instructions. This level of organization remains consistent across different values of K, indicating a strong internal preference for formatting and instructional clarity.

**LLaMA 3** exhibits notable variability and occasional confusion, particularly at K=0, where it often introduces largely irrelevant content (e.g., ray modes) and inserts unrelated question-and-answer material. However, its performance improves with increased document retrieval, allowing it to focus more coherently on distinctions such as cone casting versus line casting.

**LLaMA 2** is the most problematic in this analysis. Its responses start with brief, sometimes inaccurate technical summaries at K=0 and progressively become longer and more disorganized. At higher K values, its outputs are characterized by redundancy, verbosity, and the inclusion of fabricated features and version numbers not present in the Unity XR Toolkit documentation.

**Mistral** maintains a more narrative and explanatory tone throughout. Its responses often provide contextual background—for example, justifying curved ray implementations as mimicking natural reaching behavior. It generally favors a user-centered perspective, emphasizing iterative testing and experiential usability.

## Discussion

***The Inadequacy of Word-based and Character-based metrics*** Our results indicate that METEOR and Normalized Edit Distance metrics are unsuitable for comparing LLM-generated text with ground-truth text. These metrics fail to detect any changes, whether for better or worse. This issue has already been highlighted in the state of the art: while we can enhance the quality of text generation, validating these improvements remains a significant challenge. LLM-as-a-Judge, on the contrary, seemed able to evaluate semantic changes in the answers as augmentation varied, suggesting it can be a precious way for assessing quality of generated text. Our results also call for more reliable and adaptive evaluation protocols for RAG-generated text, especially in applied contexts where factual grounding and task-specific relevance are critical.

***The Impact of Document Exposure on LLM Answer Quality*** Our analysis indicates that increasing the number of documents supplied to LLMs generally enhances answer quality, as assessed by judge scores. It is important to note that each model starts from a different baseline considering judge score levels at K = 0, so the improvements observed should be considered relative to these initial performance levels. Across all models, there is no significant difference between K = 0 and lower document counts (e.g., K = 1 or 2); however, a significant improvement arises at K = 4 and strengthens up to K = 12. Model-specific responses vary, however. In particular, Llama2 7B begins to show improvements at K = 3, peaking at K = 9, while Llama3 8B only shows a significant improvement starting at K = 8, with its best performance also at K = 9. In contrast, GPT-4o Mini only exhibits a significant positive effect at K = 12, and neither GPT-3.5-Turbo nor Mistral 7B shows any significant changes regardless of document exposure. These findings suggest that document exposure should be tuned according to the specific LLM in use. For models such as Llama2 7B and Llama3 8B, fewer documents may be enough to enhance quality, while GPT-4o Mini necessitates more context to achieve improved performance. Altering document counts may not influence output quality for applications using GPT-3.5-Turbo or Mistral 7B. Overall, our findings support a model-specific approach when integrating additional documents, ensuring a balance between quality improvement and computational efficiency.

***Generalizability of the Approach*** While this study focuses on Unity's XR Interaction Toolkit as a case study, the proposed RAG pipeline and evaluation framework are domain-agnostic and can be readily adapted to other poorly documented or emerging technical domains. Any area where developers face a steep learning curve due to fragmented resources, such as scientific software libraries, robotics SDKs, web frameworks, or enterprise APIs, can benefit from a similar pipeline for knowledge extraction, augmentation, and evaluation. Moreover, the benchmark construction methodology, which draws on publicly available instructional content and user-centered question formulation, provides a repeatable strategy for generating realistic evaluation sets in other domains. Finally, the analysis of retrieval depth across different LLMs offers empirical insights that remain relevant across contexts, guiding practitioners in tuning augmentation strategies based on model characteristics.

## Limitations

Our study relied partially on OpenAI models via the LangChain API (version 0.3.4). Setting the temperature to 0 is not sufficient to have a deterministic output given the same

**Table 1** Info sources scraped

| Source | # Documents | Filters applied |
|---|---|---|
| *Articles* | | |
| XRI Toolkit Doc | 28 | – |
| Medium | 18 | TAG: unity-xr xr-in-unity \|vr-unity |
| Unity Doc | 48 | KEYWORD:.xr \| -xr \| /xr \| .vr \| /vr \| .ar \| /ar |
| LearnXR | 20 | TAG: xr-development |
| *Supervised forums* | | |
| G. Discussion | 0 | – |
| Fmod | 1143 | TAG: unity |
| StackOverflow | 83 | TAG:unity-game-engine & (QUERY: XR Interaction toolkit \| XRI \| XR) |
| GameDev | 50 | TAG:unity & (QUERY: XR Interaction toolkit \| XRI \| XR) |
| Unity Discussion | 254 | TAG: XR |
| *Unsupervised forums* | | |
| Reddit | 208 | QUERY: XRI \| XR Interaction Toolkit SUBREDDIT: all \| unity \| unity3D \| oculus \| hololens \| virtualreality \| virtualrealitydev \| Microsoft \| unity3d_help |
| G. Issues | 10 | TAG: is:issue & is:closed & sort:relevance-desc |

input, yet, from empirical observations, generations to the same prompt tend to remain consistent across multiple runs. Additionally, using external services raises potential data privacy concerns. The usage of local " small " LLMs rather than larger LLMs was due to hardware limitations (NVIDIA RTX 3060 12GB). Nevertheless, the judge is LLM-based and its ranking is affected by the model quality and effectiveness in matching the ground-truth answer with the generated one. A more sound—yet impractical—approach would have been to ask to human judges to perform the assessment.

## Conclusion and Future Work

In this paper, we presented a methodology for building a RAG system specialized for a specific development framework. We instantiated this process using the Unity XR Interaction Toolkit version 2 as a case study. The numeric validation was conducted using a manually crafted question-answer benchmark based on comprehensive online courses. The benchmark consists of 101 human-crafted question–answer pairs, each annotated with a difficulty level for novice users and a topic category, with a imbalanced distribution across these labels. The benchmark was applied to five LLMs, iteratively changing the level of augmentation K (number of documents retrieved). Word-based (METEOR) and character-based metrics (Normalized edit distance) have shown difficulty tracking semantic differences between the generated and ground-truth answers. On the contrary, LLM-as-a-Judge seemed able to track such differences. Furthermore, our quantitative analysis revealed that increasing the number of retrieved documents generally improved judge scores, with an overall positive effect occurring at lower K values before stabilizing. However, the number of documents needed for improvement varied across models, highlighting the need for model-specific approaches when integrating additional documents. More broadly, our work demonstrates that RAG constitutes an effective means of specializing LLMs to niche domains. A carefully constructed RAG pipeline and benchmark allows for systematic adaptation using off-the-shelf models, paving the way for scalable domain adaptation across technical fields.

In future work, we plan to expand this study to other toolkits by experimenting with other RAG solutions and by expanding the benchmark. In particular, we aim to explore alternative retrieval strategies that actively incorporate both the question and the answer, rather than relying solely on answer-based retrieval. Additionally, we want to balance the benchmark based on the "difficulty" metadata. By doing this, we could further analyze whether the growth of the augmentation level influences the generation differently based on the difficulty of the question. We also aim to validate the quality of the pipeline by involving external experts rather than relying solely on the authors. Finally, to better understand the quality of the scores assigned by an LLM as a judge, we plan to sample benchmark answers and have them evaluated by human judges to analyze the differences between human scores and those given by the LLM.

## Appendix A: Implementation Details of the RAG Pipeline

### A.1 Source Categorization and Filtering

Public sources include:

- **Articles:** Technical documentation, developer blogs, and Unity's official manuals.
- **Supervised Forums:** Platforms like Stack Overflow, where accepted answers indicate content reliability.
- **Unsupervised Forums:** Reddit and GitHub Issues, where answers must be filtered or synthesized.
- **Instructional Materials:** Publicly accessible course websites; video content was excluded for legal reasons.Private sources, though not used in this study, may include:

- Developer-authored notes (structured or raw)
- Slide presentations (potentially parsable via image-to-text models)

### A.2 Scraping and Processing Pipeline

The scraping process followed three steps:

1. **Link Extraction:** Links were gathered from documentation indices and forum search pages via tag or keyword filters.
2. **DOM Parsing:** Using Selenium and BeautifulSoup, we extracted relevant blocks: main content (articles), question + accepted answer (supervised forums), or question + all replies (unsupervised forums).
3. **Text Extraction:** Cleaned HTML content was converted to Markdown or plain text using `html2text`. Unstructured replies were filtered using upvotes where available.Table 1 is a reference to the resources we scraped for building the dataset.

### A.3 Standardization

All extracted data were converted into question–answer pairs. For:

- **Articles:** GPT-4o was prompted to generate a question and answer from each passage.
- **Supervised Forums:** Original question and accepted answer were retained.
- **Unsupervised Forums:** GPT-4o synthesized an answer by reading all replies, weighted by upvotes if present.The final dataset was serialized as a JSONL file. Answer embeddings were generated using OpenAI's Ada v2 model and stored in a Pinecone vector database.

### A.4 Code and Tools

Key tools used in the implementation:

- `requests`, `Selenium`, `BeautifulSoup` - for scraping
- `html2text` - for text extraction
- OpenAI's Ada v2 - for embeddings
- Pinecone - for vector storage and retrieval
- Langchain - to orchestrate retriever-LLM calls

## Appendix B: RAG Construction Details

To construct the retrieval-augmented generation system, we followed these steps:

### B.1 Data Preparation

The final JSONL file produced by the processing pipeline contains question–answer pairs derived from documents across multiple categories. Each record takes the form:

$$r_i = \{q_i, a_i\}$$

### B.2 Embedding and Indexing

We used OpenAI's Ada v2 model to compute embeddings of the answers:

$$\text{emb}_{a_i} = \text{ADA2}(a_i)$$

These embeddings, along with the associated question and answer, were stored in Pinecone as:

$$w_i = \{\text{emb}_{a_i}, \text{metadata} = \{q_i, a_i\}\}$$

### B.3 Retriever Configuration

To enable fast and scalable retrieval:

- We used Pinecone as a vector database with cosine similarity as the distance metric.
- The retriever was implemented via LangChain's interface, enabling clean integration with different LLM backends.
- During query time, the user question is embedded, and the retriever selects the top-$k$ records with the smallest cosine distance to the query.
- Retrieved passages are concatenated and injected into the prompt window for the LLM call.

### B.4 Libraries and Tools

- `openai` Python SDK for embedding with Ada v2.
- `pinecone-client` SDK for vector storage and retrieval.
- `langchain` for prompt orchestration and retrieval-LLM integration.
- All code is available in our GitHub repository.

## Appendix C: Pseudocode

We write a simplified pseudocode representing the pipeline core and an example of what the developer should do to create a custom source scraper.

```
ABSTRACT CLASS GenericScraper
    ABSTRACT METHOD get_urls()→ list[str]
    ABSTRACT METHOD needs_browser_to_scrape()→ bool

    // This is the main function to call to start the scraping and
        conversion
    PUBLIC METHOD build_docs()
        FOR EACH url IN get_urls()
            page_html ← GET HTML of url // via HTTP or headless browser
            page_html ← CLEAN page_html // remove unwanted tags

            title ← get_title_from_webpage(page_html, url)
            content ← get_content_from_webpage(page_html, url)
            {question, answer} ← content_to_qa(url, page_html, content)
            SAVE {question, answer} as title.json

    STRUCT QA
        FIELD question: str
        FIELD answer : str

    ABSTRACT STRUCT WebpageContent
    ABSTRACT METHOD get_content_from_webpage(page_html: HTML, url: str
        ) → WebpageContent
    ABSTRACT METHOD get_title_from_webpage(page_html: HTML, url: str)
        → str
    ABSTRACT METHOD content_to_qa(url: str, title: str, content:
        WebpageContent) → QA


ABSTRACT CLASS GenericType1_WebArticle EXTENDS GenericScraper
    OVERRIDE STRUCT WebpageContent
        FIELD main_content : str

    OVERRIDE METHOD content_to_qa(url:str, title:str, content:
        WebpageContent) → QA
        prompt ← Prompt D.1(content["main_content"])
        llm_output ← SEND prompt to LLM
        {context, question, answer} ← PARSE llm_output (via RegEx)
        RETURN {question, answer}


ABSTRACT CLASS GenericType2_Forums_MarkedAnswer EXTENDS GenericScraper
    OVERRIDE STRUCT WebpageContent
        FIELD question_body: str
        FIELD best_answer_body: str

    OVERRIDE METHOD content_to_qa(url: str, title: str, content:
        WebpageContent) → QA
        RETURN {content["question_body"], content[best_answer_body]}
```

```
ABSTRACT CLASS GenericType3_Forums_NO_MarkedAnswer EXTENDS
    GenericScraper
    PRIVATE STRUCT CommentInfo
        FIELD positional_id: str
        FIELD likes: int OR None
        FIELD content: str
        FIELD depth: int

    OVERRIDE STRUCT WebpageContent
        question_body: str
        replies: list[CommentInfo]
        max_like: int OR None

    OVERRIDE METHOD content_to_qa(url:str, title:str, content:
        WebpageContent) → QA
        IF content["max_like"] THEN
            threshold ← content["max_like"] * 0.25
            FILTER content["replies"] WHERE likes ≥ threshold

        merged_replies ← CONCATENATE all reply info as a single string
        prompt ← Prompt D.2()
        LLM_output ← Send the prompt to the model
        RETURN {content["question_body"], LLM_output}

CLASS Example EXTENDS GenericType1_WebArticle
// This shows how a Developer may create a custom scraper for specific
     web articles

    OVERRIDE METHOD needs_browser_to_scrape()→ bool
        flag ← Developers decides if they need to use a browser to
            scrape the web pages
        RETURN flag

    OVERRIDE METHOD get_urls()→ list[str]
        urls ← Developers defines list of urls to scrape
        RETURN urls


    OVERRIDE METHOD get_title_from_webpage(page_html: HTML, original_
        url:str) → str
        title_html ← page_html.find("h1")
        RETURN html2text(title_html) // html2text is a Python module
            able to extract the inner text from HTML


    OVERRIDE METHOD get_content_from_webpage(page_html: HTML, original
        _url:str) → WebpageContent
        section_html ← page_html.find("section")
        text_content ← html2text(section_html )
        RETURN {"main_content":text_content }
```

## Appendxi D: Prompt Templates

We list the main prompts used in this project for LLM. Please refer to our GitHub repository for full details. We have put some common text inside variables to reduce the verbosity. We grouped these variables under the name of *DomainInfo*, meaning that they should be changed by the developer based on the domain of interest.

```
DOMAIN_FULL_NOME := Unity XR Interaction Toolkit 2
DOMAIN_SHORT_NOME := Unity XRI v2

REAL_QUESTION_EXAMPLES := [N.B. explicit line break]
- "I want to create a black fade-in effect when teleporting around.
    Are there any tools or features, especially in XRI, that can help
    me achieve this, and what steps should I follow to implement it?"
- "What's the structure of XRI? What kinds of scripts should I be
    aware of?"
- "How does Audio work when using XRI?"
- "How can I customize my UI Canvas buttons, considering that I'm
    working on a Unity XRI Project?"
- "How to set up XRI in my Unity scene?"
- "How do I intercept when a Controller Button is pressed?"

ROLE := You are a Human Developer expert in {DOMAIN_FULL_NOME}.

TARGET := Users are Unity developers who have never used {DOMAIN_SHORT
    _NOME} and want to learn it with your help.
GENERIC_JOB_ANSWER_USER_QUESTIONS := Your goal is to answer user
    questions.
GENERIC_JOB_GENERATE_SYNTHETIC_DATASET := Your goal is to generate a
    synthetic dataset about {DOMAIN_SHORT_NOME}.
```

### D.1 Article to Question-Answer Pair

```
{DomainInfo.ROLE} {DomainInfo.GENERIC_JOB_GENERATE_SYNTHETIC_DATASET}
    {DomainInfo.TARGET}

I'm interested in questions that are related to {DomainInfo.DOMAIN_
    SHORT_NOME} and can be answered with the documentation, but the
    user doesn't know the existence of the documentation and doesn't
    have access to it. This means that:
1) You should not generate the question if the answer is not covered
    in the documentation.
2) You should not answer the question with a reference to the
    documentation.

Here are some examples of real user questions, you will be judged on
    how well you match this distribution:
{DomainInfo.REAL_QUESTION_EXAMPLES}

<DOCUMENT>
{document}
</DOCUMENT>

You will now generate a user question and the corresponding answer
    based on the above document.
1) Explain the user context and what problems they might be trying to
    solve.
2) Generate the user question.
3) Provide the accurate answer in markdown format to the user question
     using the documentation.

You'll be evaluated on:
- How realistic is that this question could be asked by a human
    developer?
- Can the question be answered with the information in the document?"
- How accurate is the answer? Remember that the user is a human
    developer, so the answer should be written in a way that a human
    developer can understand.

Your output must be in the format below:
User Context: <Generate the context>
User Question: <Generate the question>
Answer: <Generate the answer>
```

## D.2 Unsupervised Forums to Question-Answer Pair

```
{DomainInfo.ROLE} {DomainInfo.GENERIC_JOB_GENERATE_SYNTHETIC_DATASET}
    {DomainInfo.TARGET}

I'm going to pass you a question made in a QA forum.
There are {n_replies} replies.

Your job is to generate a unified answer based on the question and the
     comments.
Judge yourself if the comments are useful or not. Try to keep the
    answer as concise as possible.

You have to write only the answer, not the question or the comments.
    Do not write any preliminary text, just your answer.
Format your answer as if you were writing a Markdown document.

DO NOT:
- Insert any names or references to the comments.
- Write any offensive or inappropriate content, even if they were
    present in the original comments.
- Write any irrelevant content.

<QUESTION>
{question}
</QUESTION>

<HUMAN ANSWERS>
{replies_stringified}
</HUMAN ANSWERS>
```

### D.3 LLM as Judge

```
Please act as an impartial judge and evaluate the quality of a
    response provided by an AI assistant to the user question
    displayed below, considering the real human answer as the
    reference point. Your goal is to give a score from 1 to 5 to the
    assistant answer considering how well it answers the user's
    question, while also comparing the response against the human
    answer. Your evaluation should consider factors such as
    helpfulness, relevance, accuracy, depth, creativity, and level of
    detail of their responses.

Begin your evaluation by comparing the AI response to the human answer
     to determine how much the assistant response was helpful and
    accurate overall. Avoid any style biases. Do not allow the length
    of the response to influence your evaluation. Be as objective as
    possible.

It's mandatory to end with a final verdict that clearly states which
    assistant's response you believe is better based on the factors
    above. The last line of your response should be the output
    followed by a number between 1 to 5, with 1 the worst and 5 the
    best.

After providing your explanation, output your final verdict by
    strictly following this format:
- Score "5": No hallucination, all information is accurate and
    supported by the context.
- Score "4": Minor hallucination, mostly accurate with slight
    embellishments.
- Score "3": Moderate hallucination, mix of accurate and false
    information.
- Score "2": Significant hallucination, mostly false or unsupported
    information.
- Score "1": Complete hallucination, entirely false or unrelated to
    the context.

<=== Example #1 ===>
I think that Assistant's answer is perfect because blablabla.

Score "5".
<=== End of Example #1 ===>

<=== Example #2 ===>
I think that Assistant's answer is completely wrong because blablabla.

Score "1".
<=== End of Example #2 ===>

<=== Human ===>
[The Start of User Question] {real_question} [The End of User Question
    ]

[The Start of Human's Answer] {real_answer} [The End of Human's Answer
    ]

[The Start of Assistant's Answer] {gen_answer} [The End of Assistant's
     Answer]
<=== End of Human ===>

How much is the assistant score?
```

## Appendix E: Benchmark Content Coverage

The domain is composed of 7 red and 7 green macro-nodes.

- Introduction [R]
- Installation [R:2, G:2 ⇒ G]
  - `com.unity.xr.interaction.toolkit` [G]
  - New Input System [R]
  - Interaction Layer Mask Updater [R]
  - Installing samples [G]
- Architecture [R:2, G:1 ⇒ R]
  - States (Theory) [R]
    - Hover, Select, Focus, Activate [R]
  - Components
    - Interactors [G]
    - Interactables [G]
    - Interaction Manager [G]
    - Interaction Groups [G]
    - Controllers [R]

Controller Recorder [R]

  - Update loop [R]
    - Interaction strength [R]
- General setup [G]
  - Action-based vs Device-based behaviors [G]
  - Create the XR Interaction Manager [G]
  - Create the XR Origin camera rig for tracked devices [G]
  - Configure XR Controller and Interactor [R]
  - Enable actions for action-based behaviors [G]
  - Create an Interactable for the player to grab [G]
  - Enable an XR provider [G]
- UI interaction setup [R:2, G:4]
  - Using the GameObject menu [G]
  - Event System [G]
  - XR UI Input Module [G]
  - Canvas optimizer [R]
  - Canvas [G]
  - Checking for UI Occlusion [R]
- Locomotion [R:0, G:3 ⇒ G]
  - Glossary [G]
    - XR Origin [G]
    - Origin [G]
    - Camera Floor Offset GameObject [R]
    - Camera [G]
    - Floor mode [G]
    - Device mode [G]
    - Locomotion System [G]
    - Locomotion Provider [G]
    - Teleportation [G]
    - Snap Turn [G]
    - Continuous Turn [G]
    - Continuous Move [G]
    - Grab Move [G]
    - Climb [G]
    - Action-based [G]
    - Device-based [G]
  - Set up a basic scene for snap turn and teleportation [G]
    - 1. Set up the XR Origin and input actions [G]
    - 2. Add snap turn and teleportation capabilities [G]
    - 3. Create teleportation interactables [G]
    - 4. Configure line type [G]
    - 5. Set line visual [R]
- Architecture [R:2, G:8 ⇒ 8]
  - XR Origin [G]
  - Locomotion System [G]
  - Locomotion Providers [G]
  - Teleportation [G]
    - Teleportation Provider [G]
    - Teleportation Area Interactable [G]

* Teleportation Anchor Interactable [G]
* Teleportation Multi-Anchor Volume Interactable [R]

– Snap Turn Provider [G]
– Continuous Turn Provider [G]
– Continuous Move Provider [G]
– Grab Move Providers [R]

* Grab Move Provider [R]
* Two Handed Grab Move Provider [R]

– Climb Locomotion [R]

* Climb Provider [R]
* Climb Interactable [R]

– Character Controller Driver [G]

- Interaction layers [G]
  – Interaction Layers Settings [G]
- Target filters [R]
  – XR Target Filter [R]

* Setup [R]
* Evaluator [R

– Evaluators list order [R]

- AR Interaction Overview [R]
  – Touchscreen gestures [R]
  – Setting up for Touchscreen AR [R]
  – AR Transformer [R]
  – AR Starter Assets [R]
- Affordance system [R]
  – Affordance state providers [R]
  – Affordance receivers [R]
  – Affordance themes [R]
  – How to set up your project using the affordance system [R]

* Add an XR Interactable Affordance State Provider [R]
* Add an Affordance Receiver [R]
* Audio Affordance Receiver [R]
* Adding a theme [R]

– XR Device Simulator overview [G]

  – Installing the XR Device Simulator [G]
  – Using the XR Device Simulator [G]

* Setting up a scene for the XR Device Simulator [G]
* Testing with the XR Device Simulator [G]
* Making the XR Device Simulator work automatically in your project [R]
* Using the XR Device Simulator prefab [G]

– Setting the XR Device simulator to work with different input bindings [R]

  – Debugger window [R]

– Input devices [R]
– Target filters [R]

- Components [R:6, G:6 ⇒ G]
  – Controller Components [R]

* XR Controller [G]
* XR Controller Recorder [R]
* XR Screen Space Controller [R]

– Interactables [G]

* XR Grab Interactable [G]
* XR Simple Interactable [G]

– Locomotion Interactables [G]

  * Climb Interactable [G]
  * Teleportation Anchor [G]
  * Teleportation Area [G]
  * Teleportation Multi-Anchor Volume [R]

– AR Interactables [R]

  * AR Annotation Interactable [R]
  * AR Placement Interactable [R]
  * AR Rotation Interactable [R]
  * AR Scale Interactable [R]
  * AR Selection Interactable [R]
  * AR Translation Interactable [R]

– Interactors [G]

  * AR Gesture Interactor [R]
  * XR Direct Interactor [G]
  * XR Poke Interactor [G]
  * XR Ray Interactor [G]
  * XR Socket Interactor [G]

– Locomotion Interactors [R]

  * Climb Teleport Interactor [R]

– Filters [R:1, G:1 ⇒ G]

  * XR Poke Filter [G]
  * XR Target Filter [R]

– Locomotion [R:3, G:7 ⇒ G]

  * Character Controller Driver [G]
  * Climb Provider [R]
  * Continuous Move Provider [G]
  * Continuous Turn Provider [G]
  * Grab Move Provider [R]
  * Locomotion System [G]
  * Snap Turn Provider [G]
  * Teleportation Provider [G]
  * Tunneling Vignette Controller [G]
  * Two Handed Grab Move Provider [R]

– Visuals [R]

  * XR Interactor Line Visual [G]
  * XR Interactor Reticle Visual [R]
  * XR Tint Interactable Visual [R]

– Affordance system [R]

  * XR Interactable Affordance State Provider [R]
  * XR Interactor Affordance State Provider [R]

– UI [R]

  * Canvas Optimizer [R]
  * Hand Menu [R]
  * Lazy Follow [R]

– Other Components [R:4, G:4 ⇒ G]

  * Input Action Manager [G]
  * XR Device Simulator [G]
  * XR Hand Skeleton Poke Displacer [R]
  * XR Input Modality Manager [R]
  * XR Interactable Snap Volume [R]
  * XR Interaction Group [G]
  * XR Interaction Manager [G]
  * XR Transform Stabilizer [R]

● Extending the XR Interaction Toolkit [R:3, G:2 ⇒ R]

  – Interactor and Interactable event callbacks [G]
  – Extending Interactors [G]
  – Extending Interactables [R]
  – Extending target filters [R]
  – Inspectors [R]

## Appendix F: Retrieved Documents

### F.1 Retriever Document ID 60

Original Document at Medium.[15]

```
To make an object in your Unity XR project attach to the hand at the
    point where it is grabbed, you can use the \"Use Dynamic Attach\"
    feature in the XR Grab Interactable component. Here are the steps
    to achieve this:
1. Select the object you want to grab (e.g., a ball).
2. In the Inspector window, find the \"XR Grab Interactable\"
    component attached to the object.
3. Scroll down to find the \"Use Dynamic Attach\" option.
4. Check the \"Use Dynamic Attach\" checkbox.
By enabling \"Use Dynamic Attach,\" the object will now attach to the
    hand at the point where it is grabbed, rather than at its origin
    point. This should make the grab interaction feel more natural and
     intuitive.
```

### F.2 Retriever Document ID 63

Original Document at Medium.[16]

[15] https://medium.com/@codingcoremd/setting-a-dynamic-grab-attach-point-in-unity-xr-b155c6afd2f9.

[16] https://medium.com/@felix.m.quinones777/so-many-controls-not-enough-hands-343bb1b20e70.

```
To set up your right-hand controller to handle both teleportation and
    object selection without conflicts in Unity XR Interaction Toolkit
     v2, you can follow these steps:
1. **Remove Unused Components**:
 - Start by removing any unused components from the right-hand
     controller, such as the **XR Ray Interactor**, **Line Renderer**,
      and **XR Interactor Line Visual**.
2. **Add New Game Objects**:
 - Under the right-hand controller, add a **Ray Interactor** and a **
     Direct Interactor** by right-clicking and selecting **XR > Ray
     Interactor (Action-based)** and **XR > Direct Interactor (Action-
     based)**. Remove the **XR Controller** component from both of
     these interactions.
3. **Set Up Teleport Interactor**:
 - Add another **Ray Interactor** and rename it to something like \"
     Teleport Interactor\".
 - Under the **XR Controller (Action-based)** component of the
     Teleport Interactor, click on the preset icon and select "**XRI
     Default Right Controller**".
 - Uncheck "**Enable Input Tracking**" and "**Use Reference**" for
     Position Action, Rotation Action, and Tracking State Action.
 - Replace the default reference of the Select Action and Selection
     Action Value with "**XRI RightHand Locomotion/Teleport Select**".
4. **Add New Components to the Right-Hand Controller**:
 - Add the **Action Based Controller Manager** component to the right-
     hand controller and configure the corresponding fields.
 - Add the **XR Interaction Group** component, drop in the **XR
     Interaction Manager**, create two elements for the **Starting
     Group Members** list, and drop in the **Ray Interactor** and **
     Direct Interactor**.
After completing these steps, you should be able to use the joystick
    on the right-hand controller for teleportation without interfering
     with object selection. This setup allows you to switch between
    different interaction modes seamlessly.
```

### F.3 Retriever Document ID 1830

Original Document at Reddit.[17]

[17] https://www.reddit.com/r/Unity3D/comments/15ofnyx/worldspace_canvas_interaction_with_rays/.

```
To achieve interaction with world-space UI elements in both 3D and VR
    using rays, you can leverage Unity's XR Interaction Toolkit. This
    toolkit provides built-in support for interacting with UI elements
     using raycasting, which can simplify your implementation and
    ensure scalability.

Here's a step-by-step guide to set it up:
1. **Install XR Interaction Toolkit**: Ensure you have the XR
    Interaction Toolkit package installed in your Unity project. You
    can do this via the Package Manager.
2. **Setup XR Rig**: Configure your XR Rig to include the necessary
    components for ray-based interaction. This typically involves
    setting up the XR Controller components for both left and right
    hands.
3. **Add XR Ray Interactor**: Attach the 'XR Ray Interactor' component
     to your VR controllers. This component will handle the raycasting
     from the controllers to the UI elements.
4. **Configure Event System**: Ensure your scene has an 'EventSystem'
    and an 'XRUIInputModule'. The 'XRUIInputModule' will handle the
    input events for the UI elements.
5. **UI Canvas Setup**: For each world-space UI canvas, ensure it has
    a 'Canvas' component set to \"World\" render mode and an 'XR Ray
    Interactor' component to handle the ray interactions.
6. **UI Elements**: Add your UI elements (Buttons, InputFields,
    Sliders, Dropdowns, Toggles) to the canvas. The XR Interaction
    Toolkit will handle the interaction logic, making them behave as
    if they were being interacted with using a mouse.

By following these steps, you can achieve a consistent and scalable
    solution for interacting with world-space UI elements in both 3D
    and VR environments without having to reinvent the wheel. For more
     detailed instructions, refer to the [XR Interaction Toolkit
    documentation](https://docs.unity3d.com/Packages/com.unity.xr.
    interaction.toolkit@2.4/manual/ui-setup.html).
```

### F.4 Retriever Document ID 434

Original Document at Fmod.[18]

[18] https://qa.fmod.com/t/dialogue-and-localization-system/14679.

```
We don't have any examples of this yet, but there is some information
    in our docs:

[https://fmod.com/resources/documentation-studio?version=2.0&page=
    dialogue-and-localization.html 16](https://fmod.com/resources/
    documentation-studio?version=2.0&page=dialogue-and-localization.
    html)
[https://fmod.com/resources/documentation-api?version=2.0&page=studio-
    guide.html#dialogue-and-localization 17](https://fmod.com/
    resources/documentation-api?version=2.0&page=studio-guide.html#
    dialogue-and-localization)

In Unity you will need to control which of the localized banks are
    loaded, only the first loaded (locale) bank will be used, you will
     need to make sure that only the correct bank is loaded.
```

### F.5 Retriever Document ID 1674

Original Document at Reddit.[19]

[19] https://www.reddit.com/r/Unity3D/comments/1eajum/designer_here_id_like_to_try_using_unity_to/.

```
If you want to use Unity to prototype non-game apps with interactive
    elements like scroll views and pagination effects, here are some
    steps and tools you can use:

1. **Scroll View Implementation**:
 - **Unity's Built-in UI System**: Unity's built-in UI system can be
     used to create scroll views. You can use the `ScrollRect`
     component to achieve this. Add a `ScrollRect` to a UI panel and
     set up the content to be scrollable.
 - **NGUI or EZGUI**: These are third-party plugins that offer more
     advanced UI features, including scroll views. NGUI is highly
     recommended for its comprehensive features and ease of use,
     though it is a paid asset.

2. **Pagination Effect**:
 - **Lerping Movements**: You can use linear interpolation (lerping)
     to create smooth transitions between pages. This can be done by
     adjusting the position of the content within the scroll view.
 - **iTween**: For more complex animations, you can use the iTween
     library to animate the transitions. This can help you mimic the
     iOS-style pagination effect.

3. **Using Playmaker**:
 - **Playmaker**: Since you are familiar with Playmaker, you can use
     it to handle the logic for your UI interactions. Playmaker allows
      you to create state machines that can manage the transitions and
      interactions without writing code.

4. **2D Toolkit**:
 - **2D Toolkit**: This tool can be used to manage your 2D sprites and
      UI elements. It provides features like 9-slice scaling, which
     can be useful for creating scalable UI elements.

5. **Alternative Tools**:
 - **Pencil Project**: For quick UI prototyping without coding, you
     can use Pencil Project. It's a free tool designed for UI/UX
     design.
 - **Briefs.app**: Another tool for creating interactive prototypes,
     though it might be limited for highly custom UI designs.

6. **Considerations**:
 - **Coding Skills**: While tools like Playmaker can reduce the need
     for coding, some basic coding knowledge will be beneficial,
     especially when using plugins like NGUI or EZGUI.
 - **Prototyping vs. Production**: Remember that Unity might not be
     the most efficient tool for rapid prototyping of non-game apps.
     Tools like Xcode with Storyboarding or dedicated prototyping
     tools might be more suitable for quick iterations.

By leveraging these tools and techniques, you can create interactive
    prototypes in Unity that mimic the behavior of common app
    interfaces, allowing you to present your ideas effectively to
    clients and developers.
```

### F.6 Retriever Document ID 1651

Original Document at Reddit.[20]

```
To use the new UI Builder and Toolkit for World Space UI in Virtual
    Reality with Unity XR Interaction Toolkit v2, you can refer to a
    script shared on Reddit that demonstrates how to use UI Toolkit in
     runtime: [Reddit Script Link](https://www.reddit.com/r/Unity3D/
    comments/qh4fe4/here_is_a_script_to_use_uitoolkit_in_runtime/).

While this script provides a starting point, integrating it with VR
    might require additional adjustments. If you encounter issues,
    consider using a more traditional approach initially to get a
    working prototype. Keep an eye on updates from Unity, as they may
    provide an official solution in the future.
```

### F.7 Retriever Document ID 1029

Original Document at Fmod.[21]

20 https://www.reddit.com/r/gamedev/comments/rheo66/alright_is_there_even_a_way_of_doing_this_new_ui/.

21 https://qa.fmod.com/t/programmer-sound-on-nintendo-switch/15151.

```
So I managed to fix my first problem by changing my code to this:

 > GCHandle stringHandle = GCHandle.Alloc(key, GCHandleType.Pinned);
 > DialogUserData d = new DialogUserData(GCHandle.ToIntPtr(
     stringHandle), speakerType, notify);
 > GCHandle dataHandle = GCHandle.Alloc(d, GCHandleType.Pinned);
 > dialogueInstance.setUserData(GCHandle.ToIntPtr(dataHandle));

where the struct looks like this:
 > [StructLayout(LayoutKind.Sequential)]
 > struct DialogUserData {
 > public IntPtr Key;
 >
 >
 > public int Speaker;
 > public uint NotifyIndex;
 >
 > public DialogUserData(IntPtr key, int speaker, uint onNotify) {
 > Key=key;
 > NotifyIndex = onNotify;
 > Speaker = speaker;
 > }
 > }

 I'm still curious how it's possible to pin a single string, but not a
      struct containing a string though.

 I found a solution to the other problem in another thread here on the
      forum. Adding the attribute [AOT.MonoPInvokeCallback (typeof (
     FMOD.Studio.EVENT_CALLBACK))] above my callback function was all
     I needed to do.
 I think this should be included in the scripting examples for future
     reference.
```

**Acknowledgments** Jacopo Mereu participated in this research while attending the PhD program in Mathematics and Computer Science at the University of Cagliari (39th cycle), supported by a scholarship funded under D.M. n. 118 (2.3.2023), within the Italian National Recovery and Resilience Plan (PNRR) – funded by the European Union – NextGenerationEU – Mission 4, Component 1, Investment 4.1.

**Funding** Open access funding provided by Università degli Studi di Cagliari within the CRUI-CARE Agreement. Jacopo Mereu participated in this research while attending the PhD program in Mathematics and Computer Science at the University of Cagliari (39th cycle), supported by a scholarship funded under D.M. n. 118 (2.3.2023), within the Italian National Recovery and Resilience Plan (PNRR) - funded by the European Union - NextGenerationEU - Mission 4, Component 1, Investment 4.1.

**Data Availability** https://github.com/cg3hci/unity-XRIv2-qa.

## Declarations

**Conflict of interest** The authors have no relevant financial interests to disclose. Prof. Lucio Davide Spano mentored Jacopo Mereu during his PhD. No other non-financial interests to disclose.

**Ethics Considerations** Our work follows ethical standards by only scraping publicly accessible documents, like online documentation, for research purposes. We ensured no copyright laws were broken, assuming these documents were open for public use. Our approach avoids any illegal activities and complies with data use regulations.

## References

1. Thirunavukarasu AJ, Ting DSJ, Elangovan K, Gutierrez L, Tan TF, Ting DSW. Large language models in medicine. Nat Med. 2023;29(8):1930–40. https://doi.org/10.1038/s41591-023-02448-8.
2. Martin L, Whitehouse N, Yiu S, Catterson L, Perera R. Better call GPT, comparing large language models against lawyers. Preprint at 2024; arxiv:2401.16212.
3. Rane N. ChatGPT and similar Generative Artificial Intelligence (AI) for building and construction industry: contribution, opportunities and challenges of large language models for industry 4.0, industry 5.0, and society 5.0. Opportunities and Challenges of Large Language Models for Industry 2023; 4. https://doi.org/10.2139/ssrn.4603221.
4. Kasneci E, Seßler K, Küchemann S, Bannert M, Dementieva D, Fischer F, et al. ChatGPT for good? On opportunities and challenges of large language models for education. Learn Individ Differ. 2023;103:102274. https://doi.org/10.1016/j.lindif.2023.102274.
5. Kocmi T, Federmann C. Large language models are state-of-the-art evaluators of translation quality. European Association for Machine Translation, Tampere, Finland 2023. https://aclanthology.org/2023.eamt-1.19/.
6. Zhang T, Ladhak F, Durmus E, Liang P, McKeown K, Hashimoto TB. Benchmarking large language models for news summarization. Trans Assoc Comput Linguist. 2024;12:39–57. https://doi.org/10.1162/tacl_a_00632.
7. Donahue C, Lee M, Liang P. Enabling language models to fill in the blanks. In: Proceedings of the 58th annual meeting of the association for computational linguistics, pp. 2492–2501. Association for Computational Linguistics, Online 2020. https://doi.org/10.18653/v1/2020.acl-main.225.
8. Barke S, James MB, Polikarpova N. Grounded copilot: how programmers interact with code-generating models. Proc ACM Program Lang. 2023;7(OOPSLA1):85–111. https://doi.org/10.1145/3586030.
9. Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in neural information processing systems. 2020;33:9459–74. https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf
10. Ashtari N, Bunt A, McGrenere J, Nebeling M, Chilana PK. Creating augmented and virtual reality applications: current practices, challenges, and opportunities. In: Proceedings of the 2020 CHI conference on human factors in computing systems. CHI '20, pp. 1–13. Association for Computing Machinery, New York, NY 2020. https://doi.org/10.1145/3313831.3376722.
11. Frau V, Spano LD, Artizzu V, Nebeling M. XRSpotlight: example-based programming of XR interactions using a rule-based approach. Proc ACM Hum-Comput Interact. 2023; 7(EICS). https://doi.org/10.1145/3593237.
12. Zong M, Krishnamachari B. A survey on GPT-3. Preprint at arxiv:2212.00857 2022.
13. Alawida M, Mejri S, Mehmood A, Chikhaoui B, Isaac Abiodun O. A comprehensive study of ChatGPT: advancements, limitations, and ethical considerations in natural language processing and cybersecurity. Information. 2023;14(8):462. https://doi.org/10.3390/info14080462.
14. Ye H, Liu T, Zhang A, Hua W, Jia W. Cognitive mirage: a review of hallucinations in large language models. In: Proceedings of the first international OpenKG workshop: large knowledge-enhanced models co-locacted with the international joint conference on artificial intelligence (IJCAI 2024) 2024. https://ceur-ws.org/Vol-3818/paper2.pdf.
15. Alan AY, Karaarslan E, Aydın Ö. Improving LLM reliability with RAG in religious question-answering: MufassirQAS. Turkish J Eng. 2025;9(3):544–59. https://doi.org/10.31127/tuje.1624773.
16. Ranjit M, Ganapathy G, Manuel R, Ganu T. Retrieval augmented Chest X-ray report generation using OpenAI GPT models. In: Proceedings of the 8th machine learning for healthcare conference. Proceedings of machine learning research, 2023; vol. 219, pp. 650–666. https://proceedings.mlr.press/v219/ranjit23a/ranjit23a.pdf.
17. Pathiyan Cherumanal S, Tian L, Abushaqra FM, Paula AF, Ji K, Ali H, Hettiachchi D, Trippas JR, Scholer F, Spina D. Walert: putting conversational information seeking knowledge into action by building and evaluating a large language model-powered chatbot. In: Proceedings of the 2024 conference on human information interaction and retrieval. CHIIR '24, pp. 401–405. Association for Computing Machinery, New York, NY 2024. https://doi.org/10.1145/3627508.3638309.
18. Ye F, Fang M, Li S, Yilmaz E. Enhancing conversational search: large language model-aided informative query rewriting. In: Findings of the association for computational linguistics: EMNLP 2023, pp. 5985–6006. Association for Computational Linguistics, Singapore 2023. https://doi.org/10.18653/v1/2023.findings-emnlp.398.
19. Chen X, Chen X, He B, Wen T, Sun L. Analyze, generate and refine: query expansion with LLMs for zero-shot open-domain QA. In: Findings of the association for computational linguistics ACL 2024, pp. 11908–11922. Association for Computational Linguistics, Bangkok, Thailand and virtual meeting 2024. https://doi.org/10.18653/v1/2024.findings-acl.708.
20. Yu Y, Ping W, Liu Z, Wang B, You J, Zhang C, et al. RankRAG: unifying context ranking with retrieval-augmented generation in LLMs. In: Advances in neural information processing systems. 2024;37:121156–84. https://doi.org/10.52202/079017-3850.
21. Akimoto K, Takeoka K, Oyamada M. Context quality matters in training fusion-in-decoder for extractive open-domain question answering. Findings of the Association for Computational Linguistics EMNLP. 2023;2023:11711–29. https://doi.org/10.18653/v1/2023.findings-emnlp.784.
22. Shi Y, Zi X, Shi Z, Zhang H, Wu Q, Xu M. Enhancing retrieval and managing retrieval: a four-module synergy for improved quality and efficiency in RAG systems. ECAI. 2024;2024:2258–65. https://doi.org/10.3233/FAIA240748.
23. Yang H, Li Z, Zhang Y, Wang J, Cheng N, Li M, Xiao J. PRCA: fitting black-box large language models for retrieval question answering via pluggable reward-driven contextual adapter. In: Proceedings of the 2023 conference on empirical methods in natural language processing, pp. 5364–5375. Association for Computational Linguistics, Singapore 2023. https://doi.org/10.18653/v1/2023.emnlp-main.326.
24. Papineni K, Roukos S, Ward T, Zhu W-J. BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics. ACL '02, pp. 311–318. Association for Computational Linguistics, USA 2002. https://doi.org/10.3115/1073083.1073135.
25. Lin C-Y, Och F. Looking for a few good metrics: ROUGE and its evaluation. In: Ntcir Workshop 2004. https://research.nii.ac.jp/ntcir/ntcir-ws4/NTCIR4-WN/OPEN/OPENSUB_Chin-Yew_Lin.pdf.

26. Banerjee S, Lavie A. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In: Proceedings of the Acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, 2005; pp. 65–72. https://aclanthology.org/W05-0909/.
27. Yujian L, Bo L. A normalized Levenshtein distance metric. IEEE Trans Pattern Anal Mach Intell. 2007;29(6):1091–5. https://doi.org/10.1109/TPAMI.2007.1078.
28. Min S, Krishna K, Lyu X, Lewis M, Yih W-T, Koh P, Iyyer M, Zettlemoyer L, Hajishirzi H. FActScore: fine-grained atomic evaluation of factual precision in long form text generation. In: Proceedings of the 2023 conference on empirical methods in natural language processing, pp. 12076–12100. Association for Computational Linguistics, Singapore 2023. https://doi.org/10.18653/v1/2023.emnlp-main.741.
29. Zheng L, Chiang W-L, Sheng Y, Zhuang S, Wu Z, Zhuang Y, Lin Z, Li Z, Li D, Xing EP, Zhang H, Gonzalez JE, Stoica I. Judging LLM-as-a-judge with MT-bench and Chatbot Arena. In: Proceedings of the 37th international conference on neural information processing systems. NIPS '23. Curran Associates Inc., Red Hook, NY, USA 2024. https://openreview.net/pdf?id=uccHPGDlao.
30. Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Inf Process Manag. 1988;24(5):513–23. https://doi.org/10.1016/0306-4573(88)90021-0.
31. Nikishina I, Sevgili, Li, MM, Biemann C, Semmann M. Creating a taxonomy for retrieval augmented generation applications. Preprint at arxiv:2408.02854 2025.
32. He P, Wang S, Chowdhury S, Chen T-H. Evaluating the effectiveness and efficiency of demonstration retrievers in RAG for coding tasks. In: 2025 IEEE international conference on software analysis, evolution and reengineering (SANER), 2025; pp. 500–510. https://doi.org/10.1109/SANER64311.2025.00053.
33. Hou X, Zhao Y, Liu Y, Yang Z, Wang K, Li L, et al. Large language models for software engineering: a systematic literature review. ACM Trans Softw Eng Methodol. 2024;33(8). https://doi.org/10.1145/3695988.
34. Qian C, Liu W, Liu H, Chen N, Dang Y, Li J, Yang C, Chen W, Su Y, Cong X, Xu J, Li D, Liu Z, Sun M. ChatDev: communicative agents for software development. In: Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: long papers), 2024; pp. 15174–15186. https://doi.org/10.18653/v1/2024.acl-long.810.
35. Mosel J, Trautsch A, Herbold S. On the validity of pre-trained transformers for natural language processing in the software engineering domain. IEEE Trans Software Eng. 2023;49(4):1487–507. https://doi.org/10.1109/TSE.2022.3178469.
36. Zhou Y, Muresanu AI, Han Z, Paster K, Pitis S, Chan H, Ba J. Large language models are human-level prompt engineers. In: The eleventh international conference on learning representations 2022. https://openreview.net/pdf?id=92gvk82DE-.
37. Lima R, Gupta S, Berrospi Ramis C, Mishra L, Dolfi M, Staar P, Vagenas P. Know your RAG: dataset taxonomy and generation strategies for evaluating RAG systems. In: Proceedings of the 31st international conference on computational linguistics: industry track, pp. 39–57. Association for Computational Linguistics, Abu Dhabi, UAE 2025. https://aclanthology.org/2025.coling-industry.4/.